Finding the Right Answer: An Information Retrieval Approach Supporting Knowledge Sharing

Pablo Gomes Ludermir[1], Renata S. S. Guizzardi[1], and Diego Sona[2]

[1] Computer Science Department, University of Twente (UT)
P.O. Box 217, 7500 AE Enschede, The Netherlands
{gomesp, souza}@cs.utwente.nl
[2] ITC-IRST,
Via Sommarive, 18 - 38050 - Povo - Trento - Italy
sona@itc.it

Abstract. Knowledge Management can be defined as the effective strategies to get the right piece of knowledge to the right person in the right time. Having the main purpose of providing users with information items of their interest, recommender systems seem to be quite valuable for organizational knowledge management environments. Here we present KARe (Knowledgeable Agent for Recommendations), a multi-agent recommender system that supports users sharing knowledge in a peer-to-peer environment. Central to this work is the assumption that social interaction is essential for the creation and dissemination of new knowledge. Supporting social interaction, KARe allows users to share knowledge through questions and answers. This paper describes KARe's agent-oriented architecture and presents its recommendation algorithm.

1 Introduction

As an organization develops, its knowledge and expertise becomes increasingly distributed. While this process promotes the growth of specialized knowledge communities, it also makes discovering relevant knowledge from these communities more difficult.

A common way to target this problem is to create a central repository to which all organizational members are asked to contribute with their own personal knowledge. These repositories may contain different documents used in the work routine, but also specialized records aimed at capturing the more tacit kind of knowledge, such as reports on lessons learned and best practices gathered by workers' experience in different projects [1]. One problem with this approach is to get users' acceptance to spend extra hours on feeding the system, without knowing who will use his/her knowledge and for which purpose. Besides, employees have no guarantee of finding useful information in the repositories when they need it [2].

Workers dissatisfaction many times leads the Knowledge Management (KM) system to be abandoned, while people continue relying on their natural ways of finding knowledge, such as asking for the help of colleagues that are part of their circle of trust. The work described in this paper aims at improving these natural processes by imitating in a virtual environment, the social processes that are involved in knowledge sharing. Instead of taking a centralized view, we rely on the distributed KM paradigm [3], providing autonomous and locally managed knowledge sources organized in a peer-to-peer community. Peer-to-peer technology supports the horizontal relationship between people, seeing them as both consumers and providers of knowledge [4]. Each peer controls his own personal knowledge artifacts and exchange knowledge with other peers based on, for example, their common interests, roles, expertise, and trust. These same factors are the main motivators for the formation of communities of practices [5], which may strengthen the relationship between peers, leading to more effective knowledge creation and sharing.

Central to this work is the recognition that social interaction is the driving force behind the creation of knowledge. Nonaka and Takeuchi [6] claim that knowledge is created as a result of a transformation cycle between explicit and tacit knowledge and, for that, social interaction is not only necessary, but essential. According to Paulo Freire [7], a question is the first knowledge sparkle, as questioning is a means to explicitate one's personal knowledge, starting with a reflection on what one knows and what one does not know. In addition to that, questioning provides an opportunity for others to express their points of view, many times tacit. In other words, such process allows the explicitation of tacit knowledge, allowing it to be shared.

In this work, we present KARe(Knowledgeable Agent for Recommendations), a socially aware recommender system that recommends artifacts to organizational members based on their natural language questions. KARe is designed and implemented as a multi-agent system, where agents cooperate to organize and search knowledge artifacts on behalf of their users, interacting in a peer-to-peer network. In order to look for the answers to the users' questions, we propose an algorithm based on the information retrieval vector model [8] that provides semantic information to the queries and to the artifacts being searched. This semantic information allows us to reduce the search space by adding a step of query scope reduction in the searching process, aimed at reducing the noise of KARe's recommentations. This extension is the main focus of this paper. The remaining of this work is organized as follows: section 2 presents recommender systems and their connection to KM; section 3 describes the proposed system; section 4 gives a description of our algorithm; section 5 brings the system's agent-oriented design; section 6 describes some related work; and finally, on section 7, some conclusions and future work ideas are provided.

2 Recommender Systems Supporting Knowledge Management

Recommender systems support users in selecting items of their interest or need from a big set of items, helping users to overcome the overwhelming feeling when facing a vast information source, such as the web, an organizational repository or the like. This kind of systems has become very popular in the late 1990s, especially due to the popularization of the Internet and the consequent danger of information overload. Today, as the number of users of the information society grows rapidly, recommender systems are not less needed.

Having in mind that KM refers to "providing the right people with the right piece of knowledge, at the right time", it becomes evident that recommender systems can be of much value in organizational settings. Workers can rely on recommender systems to find out specific information, or to look for people who would know what they need.

Recommendations may be based on similar items to those a given user has liked in the past (content-based recommendation); or on items owned by users whose taste is similar to those of the given user (collaborative recommendation) [9]. Both types of recommendation may be quite beneficial in Knowledge Management communities, where knowledge is distributed and, thus, the knowledge one needs may be hard to find and sort out. Here, we explore a hybrid approach, as artifacts are recommended based on their content, but also taking into account cognitive characteristics of the system users.

Besides the difference given by the recommendation approaches described above, recommender systems can also be differentiated by [10]: the items they recommend (systems have been developed to recommend web pages [9], movies [11], etc.); the nature of the user models they use to guide the recommendations (e.g. history of items accessed by the user, topics indicating user interest, etc.); the recommendation techniques (mainly, how the user model is represented, what kinds of relevance mechanisms are used to update the user model, and which algorithm is used to generate recommendations); and the recommendation trigger, i.e. whether the recommendation is started by the user or by the proactive behavior of the system.

3 KARe: Knowledgeable Agent for Recommendations

KARe is a multi-agent system that recommends artifacts to meet the user needs, simulating the natural social processes involved in knowledge sharing. In real life, asking and answering to questions is an interactive process. The questioner finds a suitable colleague and poses his doubt. Usually, this choice is based on the questioner's assumption that his colleague knows about the targeted subject, besides feelings of trust and comfort towards the responder (e.g. "he is not going to judge me for my question"). The responder, on his turn, is likely to provide the questioner's with some help, provided that the trust between them is mutual. He will then use his own language and knowledge to provide the answer to the questioner. Besides solving the problem at hand, having the answer gives the questioner the ability to share this new knowledge with other colleagues.

In KARe, we simulate this process using a peer-to-peer infrastructure. Each user (a peer) is able to organize his knowledge assets (typically, working documents) according to his own domain conceptualization, using a taxonomy or context. A taxonomy or context is a concept hierarchy that describes the user's view of a specific domain [12] (see Figure 4). After defining meaningful concepts and their inter-relationships the user distributes the artifacts according to the "matching" concepts in the hierarchy.

The system allows the user to pose natural language questions[3], searching in other peers' collection for answers among their stored artifacts. These artifacts can be documents or messages sent by other peers responding previous similar questions. Having received a suitable answer from another peer, the questioner now has this answer stored in his own context, and can be consulted by others regarding the same subject. We chose to replicate the same knowledge in several peers to increase the possibility that this knowledge will remain in the organization even when some members are not available anymore. Thus, again we simulate the natural processes of knowledge sharing, i.e. when a colleague provides us with some explanation, we keep it and are now able to give it to someone else.

On the other hand, if the questioner is not satisfied with the answer, the system provides him with a list of possible responders. Responders are chosen based on their knowledge expertise (given by their context) but also by cognitive and personal features, such as: a) trust: each peer maintains a "list of friends", indicating who they trust; b)the organizational role of the responder: similar roles to that of the questioner are preferable, hoping that feelings of comfort and similar vocabularies may be favorable to questioner/responder interaction; and c)the responder's collaborative level: calculated by the system based on how often a given peer answers to questions that are sent to him/her. The system peers can group themselves into communities. KARe encourages community building by recommending the interaction of the user with other peers that share interests with him/her, have the same role, or are friends with the same people.

Figure 1 shows a screenshot of KARe. On the left part of the window there is the user context, showing in a tree of concepts how the user has structured his knowledge.

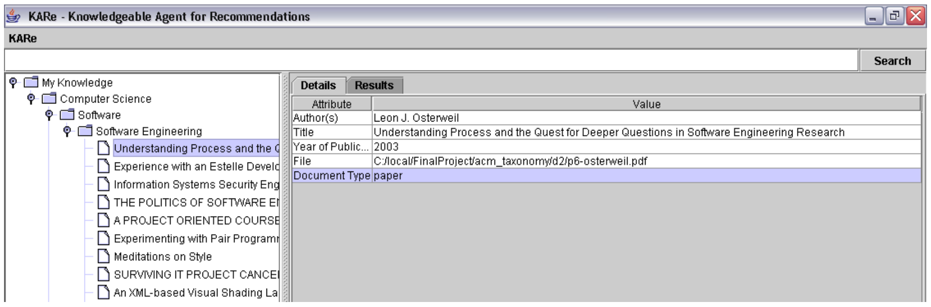
On the right side we divided the screen in two parts. The "Details" screen shows information (metadata) of the selected item (concept or artifact) on the concepts tree. The "Results" screen shows the answers to the questions performed, classified by peer and similarity with the query as we can see in Figure 2. Here are the main differences of KARe compared to other recommender systems:

(a) We provide a question/answer function, instead of a simple keyword search. The user needs are indicated through an imitation of the human social behavior of asking and answering questions to colleagues, i.e. stimulating the user's interaction by supporting the question pedagogy [7];

[3] Note that this does not mean that our recommendation algorithm performs natural language processing. The question is made in natural language by the user, but then our algorithm syntactically represents it as a set of keywords

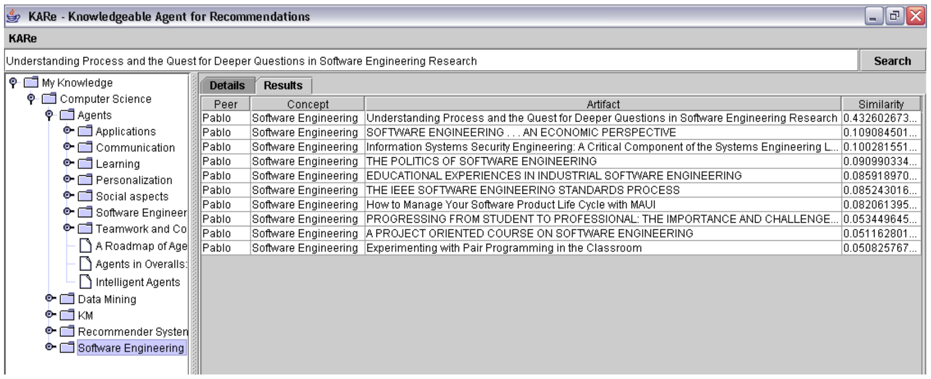
(b) The users structure their knowledge according to their own conceptualization of a domain. This gives autonomy to the users to organize knowledge following their own points of view and language.

(c) The users share documents and messages on their will. This kind of knowledge is decentralized, i.e. it is present in each user's computer, not on a central server host. Furthermore, this results in the artifacts being available for the system, rather than captured by a crawler.


KARe - Knowledgeable Agent for Recommendations
KARe
Search
My Knowledge
Computer Science
Software
Software Engineering
Understanding Process and the Q
Experience with an Estelle Develo
Information Systems Security Eng
THE POLITICS OF SOFTWARE EN
A PROJECT ORIENTED COURSE
Experimenting with Pair Programm
Meditations on Style
SURVIVING IT PROJECT CANCEL
An XML-based Visual Shading La
Details
Results

Attribute	Value
Author(s)	Leon J. Osterweil
Title	Understanding Process and the Quest for Deeper Questions in Software Engineering Research
Year of Public...	2003
File	C:/local/FinalProject/acm_taxonomy/d2/p6-osterweil.pdf
Document Type	paper

Fig. 1. Screenshot of KARe's main screen



Peer	Concept	Artifact	Similarity
Pablo	Software Engineering	Understanding Process and the Quest for Deeper Questions in Software Engineering Research	0.432602673...
Pablo	Software Engineering	SOFTWARE ENGINEERING . . . AN ECONOMIC PERSPECTIVE	0.109084501...
Pablo	Software Engineering	Information Systems Security Engineering: A Critical Component of the Systems Engineering L...	0.100281551...
Pablo	Software Engineering	THE POLITICS OF SOFTWARE ENGINEERING	0.090990334...
Pablo	Software Engineering	EDUCATIONAL EXPERIENCES IN INDUSTRIAL SOFTWARE ENGINEERING	0.085918970...
Pablo	Software Engineering	THE IEEE SOFTWARE ENGINEERING STANDARDS PROCESS	0.085243016...
Pablo	Software Engineering	How to Manage Your Software Product Life Cycle with MAUI	0.082061395...
Pablo	Software Engineering	PROGRESSING FROM STUDENT TO PROFESSIONAL: THE IMPORTANCE AND CHALLENGE...	0.053449645...
Pablo	Software Engineering	A PROJECT ORIENTED COURSE ON SOFTWARE ENGINEERING	0.051162801...
Pablo	Software Engineering	Experimenting with Pair Programming in the Classroom	0.050825767...

Fig. 2. KARe showing the search results when using the proposed approach.

The "questioning" function takes place in four steps: (1) The user chooses a concept related to his question followed by (2) the question itself. At this point, the question is broadcasted among the peers, and their local agents (3) find appropriate concepts within their knowledge context that matches the question's concept, and then finally (4) search for answers within the chosen concepts.

Note that we do not make any semantic analysis of the concepts, such as those applied on ontology mapping. Instead, the process of finding the appropriate concept that matches the question is solved by calculating the similarity between the artifacts associated with the question's concept (on the questioner's side), and the artifacts organized according to the other possible responders' context. Therefore, we assume that if the artifacts classified in two concepts are similar (one from the questioner and another from the responder), then, those two concepts are similar. This similarity function is further explained in section 4. Since the artifacts are grouped within similar concepts (in the questioner's and responder's contexts), the retrieval system will decrease the level of noise (i.e. the number of artifacts that are not relevant to the user) of the search results.

4 KARe's Recommendation Algorithm

This section presents the information retrieval model based on reference vectors that, taking into account the user context where knowledge is previously structured, helps retrieving the best matching documents in a remote structured knowledge archive.

In the vector model for information retrieval [8], documents and queries are treated as real algebraic vectors where the dimensions of the vectors is determined by the dimension of the vocabulary (i.e., made of all terms for the given domain). Therefore, once the vocabulary has been determined, all documents are represented by vectors, each element of which identify the frequency of the corresponding term in the document itself. Having this vectorial representation, it is quite easy to determine the similarity between couples of documents or between a document an a query.

In figure 3, the depicted vectors are the abstraction of a query (Q) and any particular document (d from a set of documents D), and the angle θ indicate how close these vectors are, thus indicating the similarity between the document and the query. Our algorithm calculates this similarity using a very common approach, i.e. the cosine of the angle θ. The result of the similarity function will, as a consequence of being a cosine, vary from 0 to 1 in an ascending order of vectors similarity. Equation 1 describes a *cosine similarity* between vectors which is a measure commonly used in information retrieval.

$$similarity(\vec{d_j}, \vec{Q}) = \frac{\sum_{i=1}^{t} w_{i,j} \cdot w_{i,Q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{i,Q}^2}} \qquad (1)$$

Here, $w_{i,j}$ is the weight of the index terms of document $\vec{dj}$ and $w_{i,Q}$ is the weight of the index terms of the query $\vec{Q}$.

To represent the documents as vectors, we assume an t-dimensional space where "t" is determined by the number of index terms. Thus, each index term corresponds to one dimension of the vector whose value is computed based on the frequency of the index term in question on the document collection. There

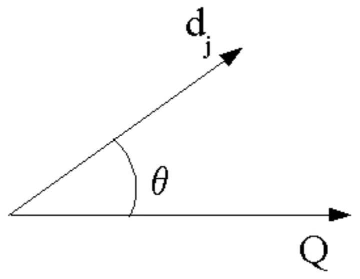


Fig. 3. Similarity is given by the cosine between Q and d_j

are several ways to compute the dimension's weight and this process is called index term weighting. For this work, we chose the $TF * IDF$ method [8].

The standard vector model approach for the retrieval of information in a given knowledge base is to compute the similarity between the query and all the documents in the given base and then select the most similar vector as the winner document. However, this approach [8, p.27] does not consider any knowledge that users may have about the structure and concepts involved with the artifacts being searched. This can lead to increased noise on the search results. For instance, trying to Google the word "agents" would result in several different kinds of agents (e.g. chemical, real state and travel agents).

The solution we propose is to include a user context in the searching mechanism, providing semantics to the artifact (i.e. relating it to the concept it is associated). Similar documents are grouped by the user under the same concept in the context tree, which facilitates artifacts' retrieval. Besides, before submitting the question, the user contextualizes the query, assigning it to a specific concept. Doing this, the user is giving to the system an extra hint on the query's content (besides the keywords contained in the query itself), leading to more accurate results.

As an example, consider two users A and B willing to share their contextualized knowledge (see Figure 4). User A submits the following question: "What is an agent?", assigning it to the "Agents" concept in the context in the left side of Fig. 4. Taking the context of the user B in the right side of the Fig. 4, it is most likely that the concept being searched is within "Computer Science → Software Engineering → Agent Oriented". But how can the algorithm be aware of that?

Essentially, each of the concepts of a user context has a vectorial representation of the words associated to it, which measures the relevancy of the words according to the documents under that given concept. The relational information among concepts (i.e., parent and/or children relationship) could help to determine reference vectors that also exploit this relational knowledge [13]. The determination of the concept reference vectors follows the equation 2.

$$w(term_i, concept_j) = \frac{f_d(term_i, concept_j)}{N_c(j)} * \log \frac{N_c}{N_c(term_i)} \tag{2}$$

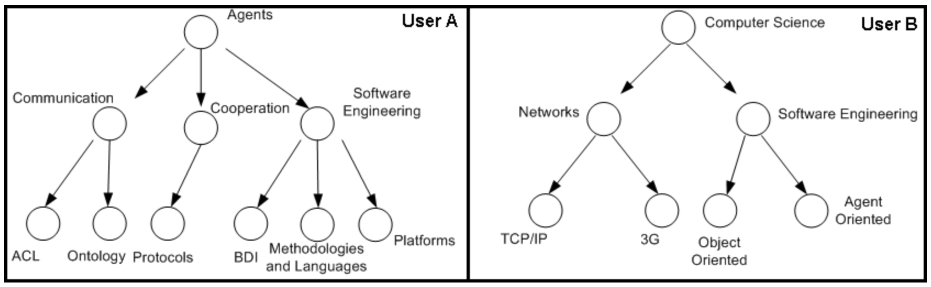


Fig. 4. The knowledge of users A and B structured in their contexts

Here, $w(term_i, concept_j)$ stands for the weight of the term "i" on the concept "j". Such approach was based on the $TF*IDF$ measure, however the parameters were chosen in a way that differentiates the concepts from each other.

In the first part of the equation, we aim at increasing the weight value as the number of documents, within the concept "j", containing the word increases. Thus, $f_d(term_i, concept_j)$ stands for the document count on the concept "j" that contains the term "i" and $N_c(j)$ is the total number of documents on the current concept (j) of the user's context. On the other hand, with the second part of the equation, we aim at decreasing the weight value as the number of concepts containing the term "i" increases. If a word is present in every concept then it is not a good candidate to differentiate the concepts from each other.

Once the concept reference vectors are determined, the searching can take place. The first step of the search is to find the concept in the taxonomy of user B that best matches the concept selected in context A for the given query. This again is simply determined by computing the cosine distance between the selected concept of A and all concepts of B. Then the most similar concept of B (i.e., with the greatest cosine value) is selected. We call this process a *query scope reduction*, the actual novelty of our approach. In summary, the query scope reduction can be seen as a reduction in the searched document set before we retrieve information from it, based on the fact that the required information is more likely to be found in a specific region of this set (in our case, within the concept that is more similar to the one selected by user A to contextualize his query). We hope that adding this process prior to the execution of the query, we will increase the quality of our search, resulting in a result of less noise, thus recommending only pertinent documents to our users.

In our example, we would calculate the similarity function among the vectorial representation of the concept "Agents" of user A and all concepts of the user B. Each concept is represented by a vector created with the basis on the documents stored under that particular concept. User A and user B have different global indexes, i.e. the vectors of each taxonomy are created based on different sets of keywords (index terms). Consequently, the first step on the query scope reduction is to project the concept vector coming from A in the new space of B. This is made with an intersection between the vector coming from A and the index of B. Then, the projected vector is compared to each of the vectors representing the concepts of taxonomy B. Finally, the concept in user B's context that has the highest similarity will be chosen for performing the query.

Then, in order to retrieve the answer to user A's question, the system searches all artifacts within the selected concept of user B. In this phase, all keywords of the user's query are taken into account to select the artifacts of the given concept. The documents are then ranked in a result set that is finally sent to user A.

Listing 1.1. An excerpt of KARe's recommendation algorithm

```
procedure answer(conceptVectorA, peerQuestion, questioner) {

  //step 1: search the best matching concept for the scope reduction
  projConceptVectorA := intersect(conceptVectorA, indexB)
  for each (concept on the user B context) {
    s := cosine(currentConceptVectorB, projConceptVectorA)
    if (s > maxSimilarity) {
      bestConcept := currentConceptB
      maxSimilarity := s
    }
  }

  //step 2: search among the documents in the bestConcept
  queryVector := createQueryVector(peerQuestion, indexB)
  for each (document in bestConcept)
    documentList.add(document, cosine(queryVector, documentVector))
  documentList.sortBySimilarity()

  //step 3: send the answer back to the questioner
  sendAnswer(documentList, questioner)
}
```

5 Agent-oriented Design

Agents have been acknowledged as adequate metaphors to represent the actors involved in Knowledge Management settings [14]. Besides, this approach has been chosen because agents can be viewed as autonomous and proactive technological building blocks, suitable for modeling peer-to-peer architectures.

In order to design our system, we applied Agent-Object-Relationship Modeling Language [15], specifically tailored for designing agents. Figure 5 presents the AOR Agent Diagram, that shows all agents and objects involved in our system design.

The main agents (represented by the rounded boxes) in KARe are the *Peer*, the *Peer Assistant* and the *Artifact Manager*. The *Peer* human agent is the user himself, who is represented in the system by the *Peer Assistant* that is responsible for: a) submitting questions from his associated human peer, and b) searching for the answers to incoming requests sent by other peers. The *Artifact Manager* collaborates with the Peer Assistant on the process of searching for answers aforementioned, manipulating for this a number of objects that are used in the generation of recommendations.

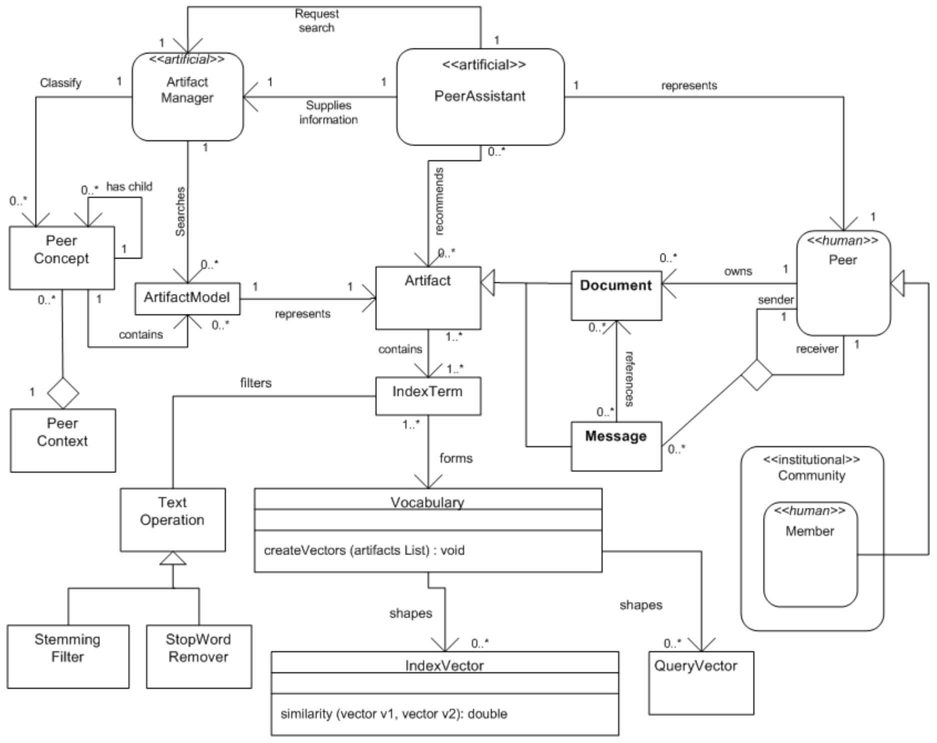


Fig. 5. KARe AOR agent diagram

The *Artifact*, *IndexTerm*, *Vocabulary*, *IndexVector* and *QueryVector* classes compose the core of the vector model. Artifact's class is the abstraction of either a document or a message (specialized in two classes). This document or message is the knowledge asset owned and shared by the user. From each *Artifact* we extract the words that will compose the dimensions of the vocabulary vector. The *Vocabulary* is the normalized vector itself and will shape the vectors of the *IndexVector* and *QueryVector* classes. The last ones are the concrete vectors used in the similarity (cosine) measure during the search, the first representing an artifact and the second representing the user's question.

The classes to perform operations over the index terms are: *Stemming Filter* and *Stopword Remover*, generalized by *Text Operations*. A large number of operations could be done to reduce and compress the index terms. We implemented the stemming and stopword remover mechanisms. Stemming is a process of reducing terms to a common root (e.g. 'swimmer' and 'swimming' are both represented by the term 'swim'), thus decreasing the number of terms on the vocabulary. Stopword remover is a way of selecting all unimportant terms and removing them from the index. These terms are usually conjunctions, articles, prepositions, etc. that do not say anything about the content of the artifact.

Finally, the *PeerContext*, *PeerConcept* and *ArtifactModel* classes describe the user contextualization. The abstraction of the user's knowledge collection is represented by *PeerContext*. This is composed by several concepts that are represented by each *PeerConcept* instance. Each *PeerConcept* may have others as their children, thus structuring the concepts relations. Finally, the *ArtifactModel* class is a representation of the artifact's metadata.

6 Experiment

The theory behind our algorithm seems to be consistent, however in order to prove that it actually brings any gains in the efficiency and accuracy of our search, we have experimented it with real data.

Since we do not have a dataset composed of contextualized questions and answers, our idea is to simulate these questions and answers using the abstract and the body of scientific papers respectively. This simulation is appropriate because, in general, a question will have only a few keywords (as a paper abstract) while the answer (which in our system can be either a natural language answer by a user, or a document maintained by this user) is typically large in content and thus in number of keywords, like the body of the paper. The contexts in our experiment will be given by two taxonomies classifying scientific papers. Taxonomy A has been created by a PhD student, to collect papers of her interest, prior to the development of the system, containing 256 documents contextualized in 24 concepts. Taxonomy B is taken from the ACM Computing Classification System [4], having 105 documents distributed within 6 concepts.

Our experiment has tested if the algorithm is able to retrieve relevant answers giving paper abstracts as queries. Relevant answers here are the documents belonging to the concept of the "target" paper (i.e. the paper whose abstract is used as a query). We compared our algorithm with the standard approach in terms of recall (i.e. the fraction of relevant documents retrieved), precision (i.e. the fraction of retrieved documents that are relevant) and the harmonic mean "F1" of recall and precision [8, p.82]. The results of our comparison are shown in table 1.

	Standard Approach	Proposed Approach
Number of Queries	100	100
Number of Rejected Queries	31	31
Number of Relevant Documents	288	456
Total Number of Relevant Documents	2000	2000
Recall	0,144	0,228
Precision	0,417391304	0,660869565
F1	0,287856072	0,455772114

Table 1. Experiment Evaluation Results

[4] http://www.acm.org/class/

As the table shows, we have submitted, in total, a 100 queries to each algorithm. The rejected queries are those that did not return any documents. This number has been the same for both approaches. The total relevant documents is the same for both approaches, since we are interested on finding as many documents as possible within a target concept. The number of relevant documents found by our approach was significantly higher than the one by the standard approach. This shows that our assumptions regarding the fact that our algorithm reduces the noise in the recommendations were correct.

7 Related Work

In this section we cite some applications with a simlar scope to KARe and we present the differences to our approach.

Google Personalized Search was recently release and it customizes search results of a web search engine based on a profile that describes the user's interests. The results are rearranged by dragging a slider that tells the personalization level (from no personalization to full personalization) [16]. The main differences between these two systems are: a) in Google, the users do not have control over the artifacts being searched; and b) Google is responsible to capture and classify information from the web, while in KARe these tasks are performed by the users.

KEEx [12] was the system that inspired KARe. Although it is also distributed in a peer-to-peer framework and allows users to share their knowledge in the form of a concept hierarchy, we can see the differences to KARe. First, we propose and implement an algorithm based on the vector model for retrieving the artifacts. Further, the approach of identifying the system roles as agents is not present there and, finally, we aim to simulate the social processes that are involved in knowledge sharing by providing a question-answering interface.

8 Conclusions and Future Work

Here we presented KARe, a multi-agent recommender system that simulates knowledge sharing social behaviors in a peer-to-peer environment. The main idea behind KARe is to promote interaction through questions and answers, aiming at facilitating the exchange of both explicit and tacit knowledge. The core of the system is an information retrieval algorithm that has been the focus of this paper.

The results of the performed experiment has shown a gain in the recommendation quality using the proposed approach. In the future, we aim at confirming this conclusion by experimenting the algorithm against different and larger datasets. However, the experiment results show that some tunning should take place, since we consider the number of rejected queries (equal for both approaches) to be relatively high. We attribute this to the fact that we used an index size of 500 keywords. Choosing the right size of the index always involves a balance between trying to be specific in the description of a knoweldge base, and running the risk of having an empty result set. We hope to make other studies in the future to realize what is the ideal index size in our case.

One interesting issue to be investigated about our algorithm is the possibility of finding not only one but several similar concepts in the responder's side. This would allow us to get better results considering that the responder's taxonomy can be more refined than the questioner's context. At the same time, we can verify the possibility of allowing the questioner to contextualize the query in more than one concept. Our plans also include the investigation of the use of information about the relations between nodes (parents and siblings) to enhance the query scope reduction process.

Another direction for future work is the design of an architecture for nomadic recommender systems, porting KARe to a mobile platform. Such service would seek service providers at the user range and exchange knowledge with them. The goal of this architecture is to take advantage of current wireless technologies, while at the same time, coping with current processing and memory limitation in mobile devices.

References

1. O'Leary, D.E.: Enterprise knowledge management. Computer **31** (1998) 54–61
2. Pumareja, D., Bondarouk, T., Sikkel, K.: Supporting knowledge sharing isn't easy - lessons learnt from a case study. In: Proceedings of the Information Resource Management Association International Conference (IRMA'03), Philadelphia, USA. (2003)
3. Bonifacio, M., Bouquet, P.: Distributed Knowledge Management: a Systemic Approach. In: Gianfranco Minati and Eliano Pessa (Eds.) Emergence in Complex, Cognitive, Social and Biological Systems. Kluwer Academic / Plenum Publishers, New York (2002)
4. Tiwana, A.: Affinity to infinity in peer-to-peer knowledge platforms. Communications of ACM **46** (2003) 76–80
5. Wenger, E.: Communities of Practice: learning, meaning and identity. Cambridge University Press (1998)
6. Nonaka, I., Takeuchi, H.: The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. Oxford University Press (1995)
7. Freire, P., Faundez, A.: Learning to Question: A Pedagogy of Liberation. Continuum Intl Pub Group (1992)
8. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc. (1999)
9. Balabanovic, M., Shohan, Y.: Fab: Content-based, collaborative recommendation. Communications of the ACM **40** (1997) 66–72
10. Miquel Montaner, B.L., de la Rosa, J.L.: A taxonomy of recommender agents on the internet. Artificial Intelligence Review **19** (2003) 285–330
11. Good N., Schafer J. B., Konstan J., Borchers A., Herlocker B., and Riedl J.: Combining Collaborative Filtering with Personal Agents for Better Recommandations. In: Proceedings of the 1999 Conference of the American Association of Artificial Intelligence (AAAI-1999). (1999) 439–446
12. Bonifacio, M., Bouquet, P., Mameli, G., Nori, M.: Kex: a peer-to-peer solution for distributed knowledge management. In: Proceedings of the Fourth International Conference on Practical Aspects of Knowledge Management (PAKM02). (2002)

13. Adami, G., Avesani, P., Sona, D.: Clustering documents in a web directory. In: Proc. of WIDM-03, 5th ACM Int. Workshop on Web Information and Data Management, ACM Press, New York, US (2003) 66–73
14. Guizzardi, R.S.S., Dignum, V., Perini, A., Wagner, G.: Towards an integrated methodology to develop km solutions with the support of agents. In: Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems, Waltham, Massachusetts. (2005) 221–226
15. Wagner, G.: The agent-object-relationship metamodel: towards a unified view of state and behavior. Inf. Syst. **28** (2003) 475–504
16. Google: Google Personalized Search. http://labs.google.com/personalized (2005)