ONTOLOGY-LEARNING SUPPORTED SEMANTIC SEARCH USING COOPERATIVE AGENTS

Cheng Zhong[1] *Zilan (Nancy) Yang*[1] *Mohsen Afsharchi*[2] *Behrouz H. Far*[1]

[1] Department of Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary, Canada
{czhong, zyan, far}@ucalgary.ca

[2]Department of Electrical and Computer Engineering, University of Zanjan, Iran
afsharchim@iasbs.ac.ir

ABSTRACT

In this paper we present (1) a method for semantic search supported by ontological concept learning; and (2) a prototype multiagent system that can handle semantic search and encapsulate the complexity of such process from the users. Agents which conduct semantic search on behalf of a user, deploy ontologies to organize structured and unstructured documents in their corresponding repositories. The ontology for each repository is individualized and commitment to a common ontology is not required. The agents can improve their search capability by learning new concepts from each other. This method thus allows agents dynamically establish common grounds on concepts known only to some of them. The concept learning is realized by analyzing positive and negative examples from other agents, and/or taking votes in case of conflicts in the received knowledge by involving other agents again.

Index Terms — multi-agent system, semantic search, ontology, concept learning, semantic interoperability.

1. INTRODUCTION

In contrast with the traditional keyword search technology which purely depends on the occurrence of words in documents, semantic search denotes one or more concepts in the context of other concepts. Understanding the denotation of concepts can help retrieval part of search engine understand the context of search, the activity the users is trying to perform, thus drive expectations on the categories of documents [6]. The essence of semantic search is semantic interoperation towards denotation part in the search phrase. Nowadays, general denotation procedures are realized depending on ontology-oriented means, and ontologies adopted are usually evolved and maintained in a distributed way. Thus, multiplicity of ontologies raises the issue of integration and renders the communication between peers involved in a semantic search ineffective.

Establishment of a common ontology for a certain domain is one of the cornerstone among cooperative agents (peers) participating in semantic search. However, agreeing on a common ontology may not be realistic. In multi-agent systems (MAS) research concerning agents' communication, having a common ontology is only possible when the design rationale, the concepts and meanings assigned to the concepts as well as the context of applying the concepts are shared. In other words, the agents must be designed in such a way that all the domain concepts and their meaning (i.e. semantics) should be provided in advance. In heterogeneous MAS, for a single domain, usually there is no agreement on the ontology among developers, and for several domains, the potential ontologies are large, unwieldy and may lead to less resolution and higher abstraction.

Recently, the idea of having agents *learn* concepts from each other has been suggested as a solution to improve agent communication. For example, the work in [8] suggests a method for learning a language and the work in [9] has focused on interactions between two agents to learn a single concept. In our previous work, we have presented a method for agents to learn concepts from several peer agents [1-2] and a method for verification of the learnt concepts [5].

Euzenat in [4] defines semantic interoperability as the faculty of interpreting the annotations at the semantic level, i.e. to ascribe each imported piece of knowledge to the correct interpretation or set of models. Possible levels of interoperability needed to be considered when trying to understand an expression from other systems are in ascending order of semantic intensity:

- *encoding*: being able to segment the representation in characters;
- *lexical*: being able to segment the representation in words (or symbols);
- *syntactic*: being able to structure the representation in structured sentences (or formulas or assertions);
- *semantic*: being able to construct the propositional meaning of the representation;

- *semiotic*: being able to construct the pragmatic meaning of the representation (or its meaning in context).

This model resembles humans' natural communication style that each semantic level can be achieved only if the lower ones have been traversed. For example, people can exchange useful information only if they have chosen a language, and clarified meanings of concepts which are critical to the topic. The idea of layered semantic interoperability has already been applied to the WWW [3] and is directly applied in our architecture of layered semantic search.

In this paper we present a method and a system that uses the ontology learning in a multi-layer semantic search. The architecture and learning supported search mechanism will be explained through the prototype system in Sections 2 and 3. The implementation details are provided in Section 4 followed by an example in Section 5 and conclusions in Section 6.

2. LAYERED SEMANTIC SEARCH ARCHITECTURE

Based on the semantic interoperability model [4], we have devised the layered semantic search architecture as illustrated in Figure 1. In this model peers can communicate and conduct search at various levels. This layered architecture will reduce complexity by breaking complex semantic interoperability into smaller problems; it standardizes interfaces between adjacent layers; it facilitates modular engineering and development of search tools; and

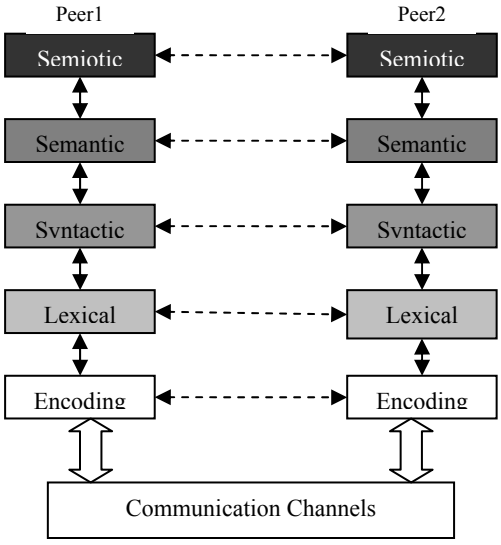

Figure 1. Architecture of layered semantic search

it accelerates evolution of technology.

The model puts some constraints on the communication between peers, namely:

1. One layer only talks with its peer layer on remote side under some agreements (or protocols). These agreements help both sides to settle natural languages, encoding standards for exchanging information, representation grammar of search phrase, etc.
2. A search phrase can be optionally initiated at any layer, then will be passed down, layer by layer, to the bottom layer (encoding layer). Each layer will add corresponding annotation information to the search phrase. Packaged phrase, finally, will be sent out.
3. Each layer can work relying on the ontology located on the same layer of semantics.

Definitions of functionalities of layers of semantic interoperability are given below.

The **encoding layer**, as base layer, defines encoding format of data exchange, thus implicitly defines the character sets of a natural language for exchanging a search phrase. ASCII and Unicode are mainly used as encoding formats. The **lexical layer** tokenizes the search phrase. At this layer, important identifiers of ontology components are identified. Functionality of this layer is not easily realizable for some natural languages such as Chinese because tokenization of sentences is a big issue due to the lack of explicit delimiter, except for punctuation, to separate each single word (or symbol). The **syntactical layer** identifies concepts by structuring words following grammar at query side, and it is capable of understanding the structured representation to extract concepts at responding side. The **semantic layer** provides ability to understand propositional meaning of the representation of search phrase. The **semiotic layer** provides ability to understand meaning of the representation of search phrase in a context (specific domain). The objective of this paper is to design and implement a prototype semantic search system to study annotation-learning workflow with focus on the lexical layer.

3. SYSTEM ANALYSIS AND DESIGN

The overview of prototype system for annotation-learning workflow within lexical layer is shown in Figure 2. In this system, software agents form a cooperative group to learn and search concepts. Each agent is responsible for a local unstructured data repository with a local ontology. The agents are responsible for organizing documents in their own repositories using any ontology they deem appropriate. The agents also communicate with each other to respond to search queries.

The system design assumptions, following the FIPA guidelines (http://www.fipa.org) are:

a) MAS is a close cooperative group. It means that there is, at least, one agent taking charge of registering service. Any other agent joining the group needs to register by offering its necessary information such service type and access point.
b) There is, at least, one agent that provides yellow page service to enable agents find each other.

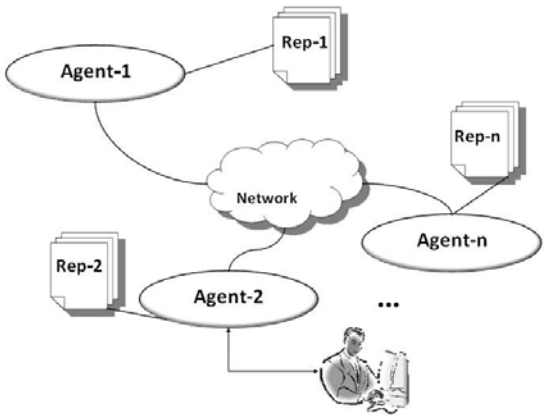

Figure 2. System overview

GAIA design methodology [10] is used to design the MAS. Each individual agent currently holds 4 roles: *Document Annotator*, *Concept Learner*, *Register Handler*, and *Concept Manager*. IBM's UIMA [7] is used to enable dynamical annotation of documents, and, therefore, enable classification of documents within their own repository. In a later phase, we will utilize an addition *Peer Finder* role to create an open cooperative MAS that enables agents automatically find each other. Details of the annotator and learner roles are explained below.

3.1. Document Annotator

Regular search usually is capable of finding tokens without any relationship between them. The tokens are not able to reflect domain-specific properties similar to atoms in early chemistry which are unable to retain chemical properties of a complex chemical substance.

The documents annotator in this project is aiming at annotating "molecules", special combinations of tokens, on which some well-defined constraints are applied. Creating such type of annotation, especially dynamically creating annotations is a fundamental role, not only for concept learning, but also for semantic search involving newly learnt concept. As the following formula describes,

$$\textit{Annotation} = F_{constraints}(token1, token2, \ldots).$$

Constraints can be, for example, window size, appearing order within certain window size of text, etc. The annotation process is depicted in Figure 3. The methods used in the annotation process are:

CreateConceptHierarchy([concept], keyword1, keyword2, …, CH1)
Annotator first creates a Concept Hierarchy (CH) using received series of keywords. This CH directly goes into the Annotation Engine (AE) to tell what is needed to be searched from the document repository.

CreateAnnotationEngine (Type1, AE1)
This method takes CH as a parameter to dynamically build Annotation Engine (AE) which is the algorithm's container.

DoAnnotation(Annotator1, Doc1, Doc2, …)
Once annotation engine is created, it will be run against repository to annotate and grab satisfying documents.

ReplyQuery(Annotated Documents, PositiveExamples, NegtiveExamples)
This method takes charge of replying to query. In this project, it also implements some specific filtering work such as selection of positive examples and creation of negative examples (see Section 3.2 for details).

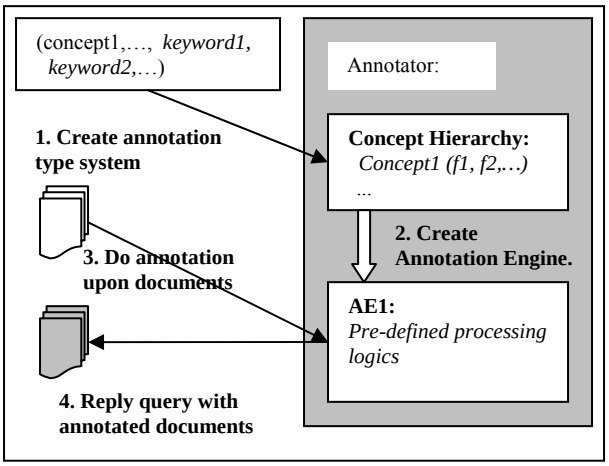

Figure 3. Process of Documents Annotation

3.2. Concept Learner

An agent knowing a concept is equivalent to having a defined classifier for that concept. The classifier is a binary function which decides whether a new incoming document belongs to (or explains) the concept. The *Concept Learner* role is used to generate this classifier.

We adopt a feature-based representation model to represent concepts. That is, each concept is composed of a set of features or keywords which are regarded as the best representation of this concept. For each concept, there exists a set of documents whose major topic belongs to this concept. Those documents are called positive objects for this concept. The list of features can be generated from the positive objects using a simple statistics method [11]. The similarity between a pair of concepts is defined as:

Sc_1,c_2: Similarity between concept c_1 and concept c_2

$Nc_1 \cap c_2$: Number of documents that belong to both concept c_1 and concept c_2 (with relevance degree greater that a defined threshold)

$Nc_1 U c_2$: Number of documents that belong to either concept c_1 or concept c_2.

Based on this definition for each concept set we have an ontology matrix as shown in Figure 4. The values in this matrix denote the similarity value between pairs of concepts. For example, the similarity value between c_1 and c_2 is 0.8, and there is no relationship between concept c_1 and c_3 since the similarity value between them is zero.

	c_1	c_2	c_3
c_1	1	0.8	0
c_2	0.8	1	0
c_3	0	0	1

Figure 4. Ontology Matrix

The learner is based on the training set available. For each concept, we select a set of positive objects (documents in this case) belonging to the concept and a corresponding set of negative objects using the ontology matrix. For example, if we want to create a classifier for concept c_1, we choose positive examples from the documents assigned to concept c_1. For negative examples, we choose documents which belong to concept c_3 because those documents do not represent concept c_1. This mechanism may not always lead to an optimum learning and we have investigated other algorithms [2]. Finally, using the positive and negative examples, we adopt a data mining algorithm to train and get the classifier for concept c_1.

3.2.1. Concept Learning Process

We illustrate the learning process using the following actions (see Figure 5).

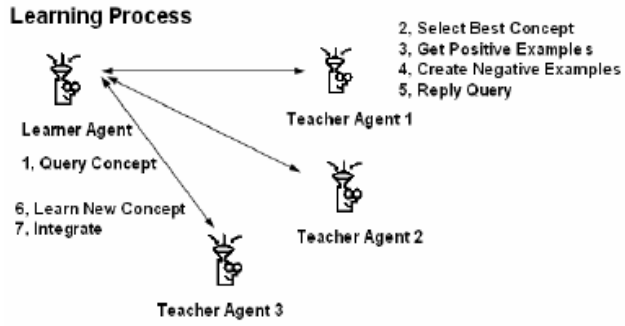

Figure 5: Concept Learning Process

QueryConcept ("keyword1", "keyword2", …)
The learner agent will start the concept learning by issuing action *QueryConcept* which will send the query to other agents. The parameters it takes are a list of keywords representing features of a potential concept.

SelectBestConcept ("keyword1", "keyword2", …)
On the receiver side, the agent first uses keywords to build annotator dynamically which then is used to annotate the candidate documents.

SelectPosEx(concept)
After getting the best concept and candidate documents associated to this concept, the receiver agent will select a given number of positive examples from the candidate documents.

CreateNegEx (concept)
The receiver agent also performs this action to produce a given number of negative examples for the concept based on the ontology matrix.

ReplyQuery(pi, ni)
The receiver agent i sends back the positive (p_i) and negative (n_i) examples to the learner agent.

Learn ((p1,n1), (p2,n2), …)
The learner agent will take all the documents transferred from the teacher agents as the training documents to form a new concept. We have examined various data mining algorithms such as Naïve Bayes and SVM. If there are any conflicting documents, they are dismissed. It means only documents which are agreed by all the agents are regarded as the documents under the new concept.

Integrate (concept)
With the new concept from the method *Learn*, the learner agent will assign a temporary name to it and suggests it to the administrator of the learner system. The administrator can approve the concept, assigns a meaningful name to it and add it to the local repository.

4. IMPLEMENTATION

4.1 System Architecture

Figure 6 shows the prototype system. Document Annotator is developed using IBM's UIMA (Unstructured Information Management Architecture) [7]. IDE Eclipse Europa is selected because it supports UIMA and Apache Tomcat http://www.eclipse.org/europa/) which is used to deploy document annotation service in this project. UIMA is featured by *type system* in which the data has a type and a set of *attribute*, *value* pairs [7], so that it is conceptually identical to concept from our point of view.

4.2. Document Annotator

The components to fulfill actions C*reateConceptHierarchy, CreateAnnotationEngine, DoAnnotation,* and *ReplyQuery* need to be built to form a complete Annotator. Central task is to deal with creation of type system.

4.2.1. Creation of Type System

According to type system definition specification [7], the first step is to build XML-based type system descriptor. Figure 7 shows an actual descriptor from the project. This is an aggregate type system which is composed of several basic primitive types. The lines enclosed by an oval represent one type definition which includes name, description, super-type name, and its features. UIMA provides APIs to build class components to dynamically adjust contents of descriptors and create a corresponding Java class.

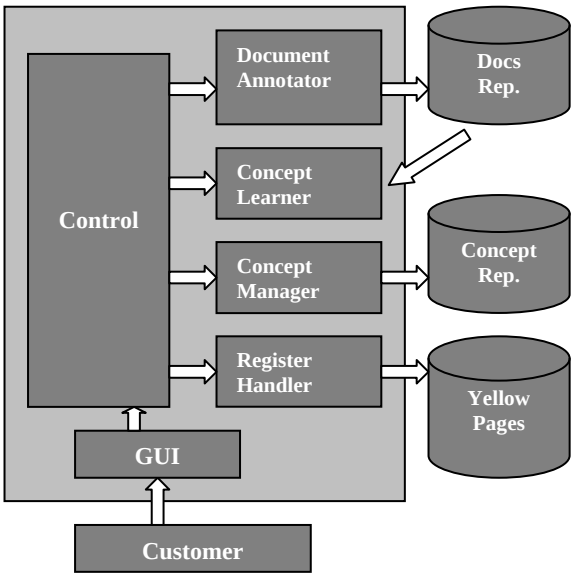

Figure 6. Architecture of Prototype System

4.2.2. Creation of Annotation Engine (AE)

Developer of an annotator has to implement a standard interface having several methods, such as *initialize()*, *process()* and *destroy()* to embed processing logic into it.

There are two ways for creator to tell *process()* method which types need to take and which types need to produce. One is manually creating a component descriptor which is XML-based document, like type system descriptor; another is using APIs that come along with UIMA to dynamically set this component descriptor. The latter is preferable because developer can select input/output type system at run-time. Immediately after the Annotation Engine is created, the *process()* will take over to scan documents and produce types as specified in AE descriptor.

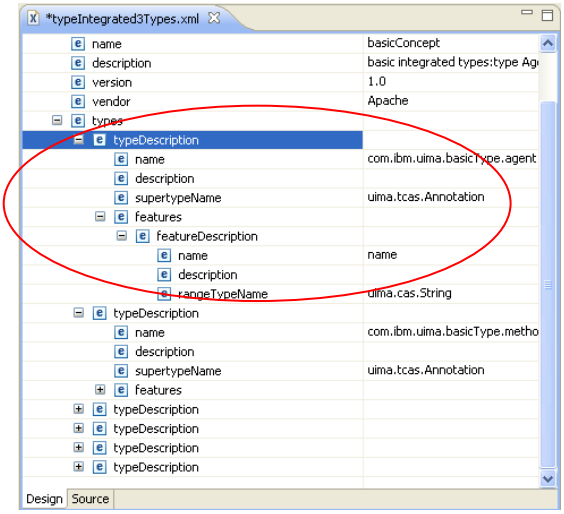

Figure 7. A Type System Descriptor

4.3. Concept Learner

The *Concept Learner* role is responsible for implementing action *Learn* which takes training documents as input, and produces concept classifier.

4.4. Concept Manager

Concept Manager role helps agent to manage set of concepts coming from three sources: newly learnt through training, selected by experts of specific domain, and newly learnt through semantic search. It need implement actions: *SaveConcept, RetrieveConcept.*

5. ILLUSTRATIVE EXAMPLE

To illustrate semantic search-learning process, here we provide a simple but illustrative example. In this scenario, three existing repositories of documents concentrating on software testing (R_{test}), object-oriented analysis and design (R_{oo}), and agent-based software engineering (R_{agent}) are created. Agent-based software engineering (R_{agent}) is considered as the local repository.

In the local repository there are several documents: some of them are about MAS methodology; some about other concepts. The initial status, as shown in Figure 8, tells that all documents, having not been annotated, are organized in flat structure.

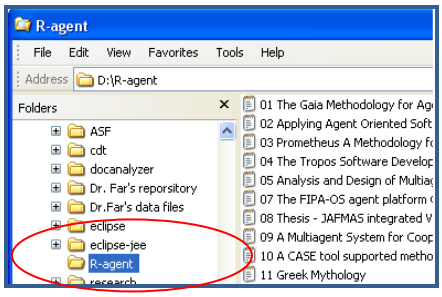

Figure 8. Snapshot of Initial Repository

Based on current status, a regular query, as shown in Figure 9, is initiated. In this case, a user wants to know "what the token Prometheus means in software engineering". Processing this query with no semantic search (as depicted by empty "Concepts" box in Figure 9) will return two documents, with completely different contents. The document 03 is talking about Prometheus MAS design methodology; meanwhile, the document 11 is about Greek mythology which is apparently not the one that user intended to receive.

To disambiguate search results, the agents will take the following steps to kick off a semantic search:

1. A concept learning routine is started to evaluate returned documents, through which new concepts, for example, "agent" (including its features) is identified; it then will be propagated within the group (see [11] for details).

2. Each agent, upon receiving this concept, will annotate its own repository. Annotation procedure re-categorizes repository by conforming to concept hierarchy. Figure 10 shows changes happened to repository R-agent.
3. New concepts will be added to the concept repository in order to support decoding query phrase, or searching.

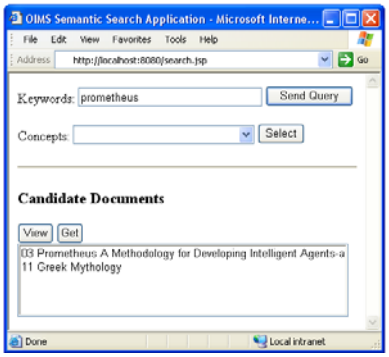

Figure 9. Illustration of Regular Search Procedure

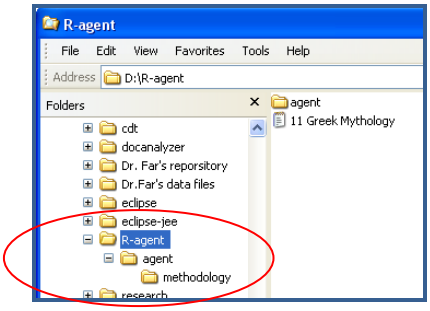

Figure 10. Snapshot of Annotated Repository

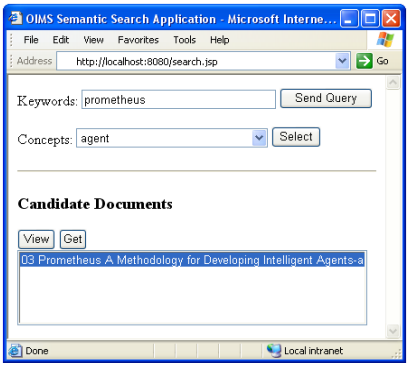

Figure 11. Illustration of Semantic Search Procedure

Once these steps are completed, a next round of search will start by involving the newly learnt concepts. Figure 11 shows a disambiguated result by sending query phrase consisting of both keywords and concepts.

From the procedure explained above, we can conclude that dynamical annotation guided by concept hierarchy is capable of categorizing the repository, consequently, making retrieval of documents more efficient.

6. CONCLUSION AND FUTURE WORK

In this paper we presented a method and a prototype MAS for semantic search-learning. This method is based on the architecture of layered semantic interoperability. The central procedure is composed of dynamical document annotation and concept learning mechanisms to solve the problem of semantic heterogeneity in distributed information management with minimum overhead and no need to commit to a common ontology. From the implementation, we can conclude that the semantic search supported by concept learning is a generative evolutional procedure. It is started by a regular search leading to learning a new concept. Then later the learnt concept is used in semantic search, and the search resolution will improve.

Future work includes implementation of the role *PeerFinder* which will lead to an open MAS. Also, research efforts will be put on organizing concepts into a hierarchy through learning. Finally, based on research achievements, we will propose a protocol for search on other layers of the semantic interoperability model.

7. REFERENCES

[1] M. Afsharchi, B.H. Far, J. Denzinger, "Ontology Guided Learning to Improve Communication among Groups of Agents," Proc. AAMAS'06, pp. 923-930, 2006.

[2] M. Afsharchi, B.H. Far, and J. Denzinger, Learning Non-Unanimous Ontology Concepts to Communicate with Groups of Agents, Proc. IAT'06, IEEE Press, pp. 211-217, 2006.

[3] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, I. Horrocks, "Semantic Web: The Roles of XML and RDF", *IEEE Internet Computing*, 2000.

[4] J. Euzenat, "Towards a principled approach to semantic interoperability," A. Gomez-Perez et al (eds.) IJCAI'2001 Workshop on Ontologies and Info Sharing, Seattle, 2001.

[5] B.H. Far, A.H. Elamy, N. Houari and M. Afsharchi, "Adjudicator: A Statistical Approach for Learning Ontology Concepts from Peer Agents," Proc. SEKE'07, 2007.

[6] R. Guha, R. McCool, E. Miller, "Using the semantic web: Semantic search," Proceedings of the WWW'03, 2003.

[7] IBM, "Unstructured Information Management Architecture (UIMA)", http://domino.research.ibm.com/comm/research_projects.nsf/pages/uima.index.html, 2007.

[8] K.C. Jim, C.L. Giles, "Talking Helps: Evolving Communicating Agents for the Predator-Prey Pursuit Problem," *Artificial Life 6(3)*, 2000, pp. 237–254.

[9] A.B. Williams, "Learning to Share Meaning in a Multi Agent System, Autonomous Agents and Multi Agent Systems 8(2)," pp. 165–193, 2004.

[10] N. Wooldridge, and D. Kinny, "The GAIA methodology for Agent-Oriented Analysis and Design," 2000.

[11] Y. Zilan, C. Zhong, B.H. Far, "A Practical Ontology-Based Concept Learning in MAS," Proc. IEEE CCECE'08, (to appear), 2008.