Case-Based Service Requests Interpretation

Chiung-Hon Leon Lee and Alan Liu
Department of Computer Science and Information Engineering,
ChungChou Institute of Technology.
Department of Electrical Engineering,
National Chung Cheng University.
leonlee@dragon.ccut.edu.tw and aliu@ee.ccu.edu.tw

ABSTRACT

A Case-Based Service-Request Interpretation (CBSRI) method is proposed to cope with the problem of how to extract service requests intention from the user's entered strings. Based on the proposed approach, the user's vague and imprecise service request will be extracted and mapped to a set of computer understandable and computable goal models. These goal models are elicited from the requirements specifications and used to represent the system capabilities to satisfy user's service requests. The CBSRI can reuse previous interpretation experience to interpret newly received service requests. The intention interpretation knowledge is stored in a case-base and the intention interpretation is based on the process of case retrieval and adaptation. A rule-based approach, a statistical-based approach, and the case-based approach are implemented to examine the efficiency of the proposed approach.

1: INTRODUCTIONS

Providing users with an environment in which interaction with computers is as easy as interaction with human beings is a long-term goal of natural language processing (NLP) [1]. There are many research issues in NLP and many techniques proposed to handle these issues such as using augmented finite state machines to parse sentence, translating parsed results into SQL, predicate calculus, or equivalent semantic networks, applying spelling and grammar checker to facilitate paper editing, and using keyword-based information retrieval techniques to retrieve information from large amounts of data. However, how to use natural language to query or command computers is still one of big challenges in computer science researches because the user might overshoot or undershoot the capabilities of natural language interface [2].

In this paper, we proposed a CBR-Based Service Request Interpretation (CBSRI) approach for interpreting service requests from system users. The extracted service request will be mapped to a set of goal models [3, 4] which is used to represent the system capabilities for satisfying user's service request. The advantage of CBSRI is that the previous request interpretation experience can be stored in a Case-Base (CB), when the system encounters a new service request, the old interpretation experience of service request can be retrieved and adapted to interpret the new service request. Since the new problem has been solved successfully and CB has no similar case, the system can store the new solution to CB for future usage.

"Understanding" a service request refers to the computer's ability to transform the verbal form entered by the user into machine-readable semantics. Service requests interpretation involves the extraction of *key concepts* from the service request, as well as inferring the *informational goal* therein. In our approach, the designed goal model will be a part of case in which contains a description of the goal model, then the terms in the service request will be used to retrieve similar cases. Using a heuristic designed knowledge base, the system can obtain the underlying information goal from the retrieved case.

We store goal models and their related service request information into CB and develop a novel similarity measure approach for mapping the user's service request to the case. An adaptation-knowledge base is developed to adapt the retrieved case. Finally, goal models could be extracted form the adapted case to represent the user's service request.

Generally, the techniques for NLP can be divided into two streams: rule-based and statistic-based [5]. Rule-based approach use handcrafting grammar for parsing. Grammars might process syntax, semantics, or both. This approach has been used in a number of systems to achieve NLP for restricted domains [6]. The shortage of the rule-based approach is that the construction of handcrafting grammar and heuristics for mapping parsed terms to semantic frames require substantial expertise and time. The process of putting capturing domain-specific knowledge, pragmatics, semantics, and syntactic together is a hard work and to write a rule set that has a good coverage of real data is difficult. Furthermore, it requires significant effort when the designer wants to expand the scope of the application domain or migrate to other domain.

The statistic-based approach attempts to extract the semantics of a service request by means of a stochastic model. This approach uses a learning mechanism to construct the stochastic model from a large annotated corpus. A suite of modeling techniques have been applies such as Hidden Markov Models (HMM) and Probabilistic recursive transition networks. The problem of requiring substantial expertise and time for constructing handcrafting grammar rules can be avoid in this approach. However, stochastic modeling needs a lot of annotated corpora for deriving the model. Manually annotating the corpora is also take time and how to acquire sufficient training data for training process is a problem.

In CBSRI, we use the concept of case-based reasoning (CBR) to extend the rule-based approach with learning capability and using learned case to interpret newly encountered service request. The mechanism used to interpret service request can be trained by pre-designed service requests and use the learning mechanism to extend interpretation capability.

This paper is organized as follows. First, we introduce the case representation method in Section 2. In Section 3, the mechanism of how to retrieve candidate cases from CB is shown. The case adaptation method is displayed in Section 4. The case storage method is shown in Section 5. A comparison result among rule-based, statistical-based, and case-based approaches is demonstrated in Section 6. The rationales of why case-based approach is more suitable for intention extraction are also introduced. Finally, we give a conclusion for this paper.

2: CASE REPRESENTATION

The target of acquiring precise requirements in software engineering [7] is similar to the target of acquiring the intention from the system services user. The difference is that the user does not provide too much information about what he wants when using the system services. The requirements engineer can elicit the requirements from the user persistently but the system services user cannot bear the system to ask him too many questions. In other words, the background knowledge about the services domain and the user should be constructed for the system to “guess” the user’s intention more precisely.

Goals identification is a crucial factor in the elicitation process of software requirements. Rolland and her colleagues [8] proposed a structure for analyzing the requirements based on a verb and its parameters. Lee et al [9] proposed a systematic approach to handling the interaction among nonfunctional requirements and their impacts on the structuring of requirement specifications. Inspired by the Rolland’s and Lee’s researches, we proposed a goal model to represent the user’s intention [3]. The goal will guide the system to generate a plan, and the execution of the plan will be expected to satisfy the user’s intentions.

There are three requirements on the goal model. First, the goal model should contain enough information for mapping the user’s entered string to a computable intention model. Second, the goal model should have a classification basis for the system designer to analysis their relationships. Finally, the goal model should contain the information of related sub-goal models to facilitate the retrieval process of the goal models. Based on these considerations, the basic attributes of a goal model are shown in Figure 1. A goal model consists of three parts: *Contents* to represent the variables which used for the mapping from extracted terms to goal models, *Properties* to describe attributes of the goal, and the *Relationships* represent the linkage of related goals or sub-goals.

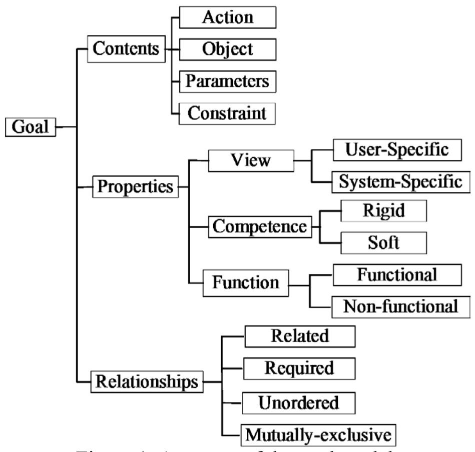


Figure 1. Anatomy of the goal model.

The relationships among the case-base, case, problem domain, and solution domain are shown in Figure 2. A case of our approach consists of three parts: the description part, solution part, and relation part. The description of the problem will be put in the description of the case here and later we will use this part for similarity measures during case retrieval. The descriptions usually are the feature set of the problem. Various representation methods can be applied here like lists, semantic networks, frames and objects. Sometimes, scenarios are also used here. We use the original service request extracted from requirement specification or entered by the user as the description part.

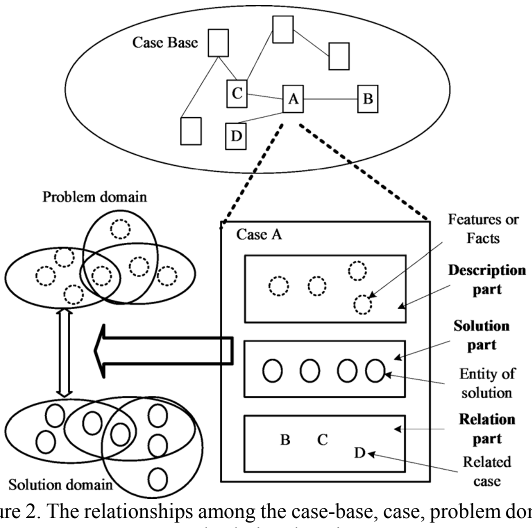


Figure 2. The relationships among the case-base, case, problem domain, and solution domain.

The solution to the problem described in the description part will be recorded in the solution part. Once an identical problem comes, we can apply this solution to solve it. The representation of solution part, like description part, can be implemented by various knowledge representation methods. In our approach, the solution part contains the *Contents* and *Properties* of the goal model. Since the *Contents* and *Properties* are the solution, we can use the case to map the user entered string to a goal model for user intention representation. Finally, the *Relationships* of the goal model will be the relationship part. Using this part, related goal models that recorded in other cases could be retrieved to extend the original goal model.

3: CASE RETRIEVAL

To retrieve relevant cases from case base, there are three major tasks: case indexing, similarity measures and case selection. Case indexing is helpful to improve the efficiency of case retrieval especially when the CBR system is having a vast case base with large amount of cases in it. It eases the problem of growing size of case base production by case storage. Similarity measures, which lead to the result of case retrieval, calculate the similarity or dissimilarity between new query case and old cases. Usually the case having highest score means it is most similar to the query case, in similarity measures, will be selected to solve the problem in the new query case. Once, if more than one cases are retrieved from case retrieval component, a CBR system have to perform case selection to select a case from these candidate cases. A process of case retrieval is shown in Figure 3.

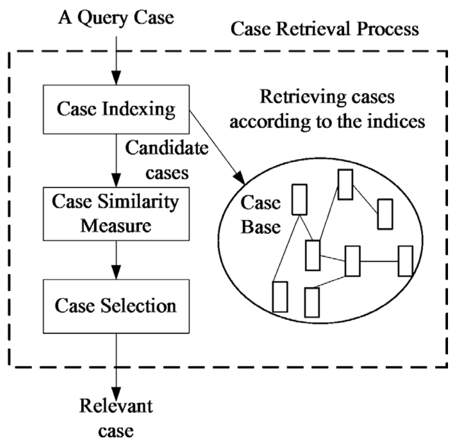


Figure 3. Case retrieval process.

3.1: CASE INDEXING

Because the description part can be seen as a set of keywords and each keyword is a feature of the description part, using the keywords as index seems reasonable. However, because the user might use different service request terms or make minor type error, we prefer to use *n-gram* approach rather than term-based approach to index the cases.

N-gram is a widely used technique in Information Retrieval (IR) research [10]. The premise of *n-gram* is to decompose terms into word fragments of size n, then design matching algorithms that use these fragments to determines whether or not a match exists. For example, the user might use terms like “scheduling”, “schedule”, or “schedul” in service request, if we only use the keyword to index cases, the first and third request might not retrieve proper cases. However, if we use 3-gram as index, the request term “schedul” will be decomposed into “sch”, “che”, “hed”, “edu”, and “dul” to retrieve relate case. In this case, the case with description feature “schedule” could be retrieved.

3.2: SIMILARITY MEASUER

A service request is a string which has various features can be used for similarity measurement such as token-based distance and edit-distance metrics [11]. For capturing all of these different similarity types, we adapt the approach proposed in the [12] to build up a set of similarity measurement vectors (SMV) that reflect the each of the different similarity types and use these vectors for similarity measurement. The vectors can be divided into three groups: token vectors, edit vectors, and other vectors. The token vector is the set of token level similarity scores between the service request entered by the user and the description part in the case. The functions for deriving token vectors include Jensen-Shannon distance (both with a Dirichlet prior and a Jelenik-Mercer mixture model) and Jaccard similarity. The edit vector can be derived from the following edit distance functions: Smith-Waterman distance, Levenstein distance, and Jaro-Winkler similarity. All of these string similarity measuring functions can be found in [11]. Lastly, the other vectors consists of the Soundex score and service request stemmer score between the service request and the description part.

Soundex is a phonetic index method and its key feature is that it codes a term based on the way term sounds rather than on how it is spelled. For example, terms that sound the same but are spelled differently, like “Smith” and “Smyth”, have the same code and are treated similar. Stemming is a technique for reducing words to their grammatical roots. A set of SMV can be derived by applying the string comparison functions to the service request and the description part of candidate cases. For example, a SMV could be <0.40, 0.40, 0.33, -8.0, 0.69, 0.77, 0.77, 0.75>.

3.3: CASE SELECTION

Many criteria can be applied for the selection of cases. For example, we can simply select the case with the highest similarity value or maybe the case with the highest adaptability if the CBR system has the ability to evaluate the adaptability of cases. The selection process used in our approach includes two steps: rescoring and testing. The rescoring step rescore each SMV for ranking candidate cases and the testing step test whether the candidate case with highest rank matches the service request.

After all of the candidate cases are scored, we then rescore each SMV for case selection. For each element of SMV, if the candidate cases with the maximum value that the value will be map to 1. The value of the rest candidate cases will be mapped to 0. For example, assume we have 2 SMV for two candidate cases SMV1 and SMV2:

SMV1=<0.42, 0.38, 0.30, -30, 0.77, 0.60, 0.77, 1.0>, and

SMV2 =<0.31, 0.28, 0.21, -37, 0.46, 0.65, 0.66, 0.94>.

After rescoring they become:

SMV1=<1, 1, 1, 1, 1, 0, 1, 0>, and

SMV2 =<0, 0, 0, 0, 0, 1, 0,1>.

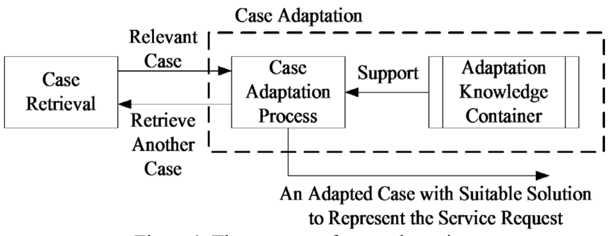
The rescoring helps to determine the best possible candidate match for the newly service request. Next, the score in the SMV will be summarized to deriving a ranking score to rank the candidate cases for case selection. For example, the ranking score of SMV1 and SMV2 is 6 and 2. Finally, the candidate case highest ranking score will be selected to test whether it suits for representing the service request. If the selected case pass the test, it will be chosen as relevant case, if not, this case will be removed from the candidate case set, and the SMV and ranking score will be recalculated again to select new case for testing. If the candidate case set becomes empty and no case pass the testing, it indicates the service request cannot be parsed.

When performing the case testing process for case selection, the system will use the *Action* and *Object* information of the goal model which stored in the candidate case to check whether the synonyms of the *Action*, *Object,* or *Parameters* can be found in the service request. If *Action* or *Object* cannot be found in the service request, the system can use the information of *Parameters* to "guess" possible *Action* or *Object.* If the information of *Parameters* is not enough to "guess" and the *Action* and *Object* cannot be found in the service request, the case will fail the testing.

4: CASE ADAPTATION

Once a similar past case is retrieved in the case retrieval process, this case is sent to the case adaptation process to adapt its solution for generating a goal model to represent the service request intention. Adaptation is important and necessary because we cannot expect the pre-designed goal models stored in the case-base covering all service requests come from the user. In other words, the CBR systems only retrieve a partial matching case to the query case in most of the time. For using this similar past case to solve current problem, we assume that similar problems have similar solutions [13]. The process of case adaptation is shown in Figure 4.

There are various techniques proposed for case adaptation. In many commercial CBR systems, null adaptation is used [14]. This approach leaves the problem of adaptation to the users themselves. This kind of system is called case retrieval systems since it is considered to have no reasoning part. Users of this kind of CBR systems must have expertise of the problem domain to adapt the cases by themselves. Automatic adaptation is needed to help those who do not have expertise of the problem domain and thus expand the applicable domains for CBR systems. Also automatic adaptation helps the CBR systems to perform autonomously and thus makes the CBR system learn without the assistance of human users. In addition, case adaptation draws the boundary of the domain within which problems can be solved. A better case adaptation is not only successful to adapt the past solution to current problem but also makes the boundary of the "can-be-solved" problem domain wider.


Case Adaptation
Relevant Case
Case Retrieval
Retrieve Another Case
Case Adaptation Process
Support
Adaptation Knowledge Container
An Adapted Case with Suitable Solution to Represent the Service Request
Figure4. The process of case adaptation.

In our approach, case adaptation is needed to help the intention extraction interface to adapt the user's input. However, we limit the service request which comes from the user cannot out of the system capability. In other words, for a service-oriented bookstore system, the request like "scheduling a meeting" will not be accepted because the request is out of the application domain. For covering wider application domain, we can design different service requester agents in different domain and design a cooperation mechanism to make them cooperate with each others to handle more complex service requests. However, this problem is out of the scope of this dissertation, we leave it as one of our future works.

For adapting the newly encountered terms in a service request, we index the concepts in the ontology by the *Contents* of goal model: *Action*, *Object*, *Constrain*, and *Parameters*. We add four relationship features: *Related*, *Required*, *Unordered*, and *Mutual-exclusive* to the concepts in the domain ontology. These features are similar to *Relationships* part of the goal model. The identification rules to determine whether the service term is a concept in the ontology is also bind to the concepts. We use these features and identification rules as guideline to explain the new term or to interact with the user for get explanation. For example, the concept "schedule" will be classified as an *Action* and the relationship features store it's related *Objects* such as "meeting", "flight", "taxi", and "bus" because the service requests might be "schedule a meeting", "schedule a flight", "schedule a taxi", or "schedule a bus". If the term "schedule" is identified in the service request but the system cannot find the predefined related *Object* "meeting", the system can "guess" whether the object of the action is "flight", "taxi", or "bus" by using the identification rules stored in the ontology. The features of *Object*, *Constrain*, and *Parameters* are defined in similar way, but the features of *Object* store its related *Parameters*, the features of *Constrain* store its related *Actions,* and the features of *Parameters* store its related *Objects.*

Using this mechanism, the interpretation capability of the system can be extend by learning the newly encountered terms. For example, assume that the service request entered by the user is "Plan a meeting on June 6 2 p.m." and the description part of the selected case might be "Schedule a meeting on June 6 2 p.m.". After interact with the user and confirm the new service request can be represented by the retrieved case, a new case "plan equals to schedule" can be added into the case-base.

5: CASE STORAGE

Case storage is the learning stage of a CBR system. With case storage, a CBR system can learn more and more cases and makes it more robust because more cases means more solutions and larger coverage of the solution space. A CBR system can store both success cases and failure cases. Once a case in the case base is retrieved and adapted for the new problem, the solution of this case can be applied to the new problem. The case which is not the original one from the case base anymore because the problem description of this case is changed to the new problem and the solution part of this case is modified from the origin one to be adapted to the new problem.

The case will be a success case if it can be used to solve new problem successfully or it will be a failure case if the result is failure. To record a success case, a CBR system learns how to solve a new problem. To record a failure case, a CBR system can avoid making the same mistake next time. In our approach, we use this storage mechanism to extend the interpretation capability and increase the interpretation correctness of the system.

One of the problem should be solved is how to maintain the case-base. If the service requests from the user change continuously and rapidly, the case-base will grow very fast, the system performance will be affected by many useless cases. In our approach, we adapt the algorithm for the growing cell structure (GCS) [15] for case-base maintenance. For implementing this algorithm, a table will be constructed to record a winning counter *C* for each learned case. The *C* is an integer to record the number of this learned case to be used most recently. The algorithm contains following steps.

(1). Increment input counter, *c*, which number the times of the service request.

$$c(t+1) = c(t) + 1.$$

(2). Update the winning counter for the case, if the case is selected as a relevant case *rc*. The updating rule is defined as follows.

$$C_j(t+1) = \begin{cases} c(t+1), & for \quad j = rc \\ C_j(t), & otherwise. \end{cases}$$

(3). Compute the difference *D* between the input counter, c, and the winning counter, *C,* as following equations.

$$D_j(t+1) = c(t+1) - C_j(t+1).$$

Finally, a case deleting threshold D_t can be defined for case-base maintenance. The case deleting rule is defined as follows.

IF $D_j > D_t$ THEN remove the case *j*.

6: EXPERIMENT RESULTS

How to separates, parses, and abstracts the entered keywords to derive a set of annotated terms is an important step for retrieving correct related goal models. In general, accurately assigning correct morphological tags to input text is an important challenge of Information Retrieval and NLP. One problem with assigning parts of speech is that a given word can be used in many ways such as the word Schedule could be a noun or verb. For getting a suitable parsing approach for intention extraction, we implement three parsing approaches for comparison: an augmented transition networks (ATN) parser, an HMM parser, and a Case-based parser. The ATN and HMM parser are adapted from [16].

In this experiment, we construct an experimental lexicon which contains some words and related part of speeches of these words. A training service request set is automatically generated by referring the lexicon as the training file for HMM and Case-based Parser. A set of parsing rule for ATN is also designed.

Three variables are defined for evaluating the performance of different parsing approach: the number of training service requests $N_{SR\_Traning}$, the number of testing service requests $N_{SR\_Testing}$, and the parsing rate. Assume that number of tokens in a service request string is N_{tokens} and the number of correct tagged tokens is N_{tagged_tokens}, the parsing rate $PR = N_{tokens} / N_{tagged\_tokens}$. The program is implemented by Java language and the platform used in this experiment is Windows XP Professional, Intel Pentium M 1.6G, 512M RAM.

First, we evaluate the recall performance of ATN, HMM, and Case Retrieval (CR) approaches. Ten input testing requests are generated by referring the predefined lexicon and without unknown words. The CR retrieves a similar previous service request for tagging the newly entered service request but without learning and reasoning. The evaluation result is shown in Figure 5. It is obvious that PR of HMM is best when using the words in predefined lexicon to generate service request strings. The PR of HMM and CR reached highest point when the number of training requests is increase to 200. The maximum PR of CR is about 0.8, because it only simply use the synonym words of retrieved sentence to parse the newly sentence.

Next, we evaluate the parsers by inputting requests which contain unknown words. Because different users might use different words for service request, we randomly replace terms in the automatically generated service request to emulate this situation. The expected result of this experiment is the PR of CBR parser should increase to a degree, because it can accumulate the parsing experience and use old parsing result to parse newly encountered requests. It should be noted that in this experiment we do not apply any adaptation knowledge for parsing to simplify the problem, because different design of adaptation knowledge will lead to different parsing rate. The experiment results proved our expectancy are shown in Figure 6.

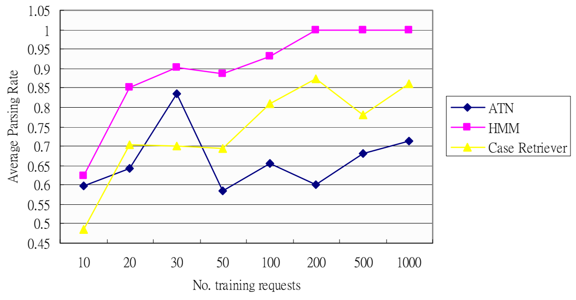


Figure 5. Evaluation results of ATN, HMM, and Case Retriever.

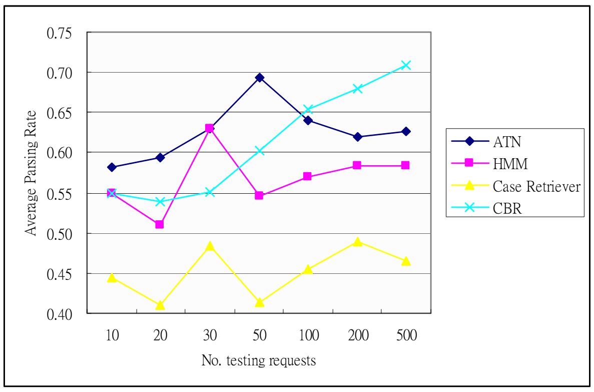


Figure6. Evaluation results of ATN, HMM, Case Retriever, and CBR. ($N_{SR\_Traning}$ =20)

The experiments results shows that the CBR parser approach has best parsing rate if the user entered requests contain unknown words for the system. The PR of the trained CBR parser is about 0.72. It means that there are about 30 % terms in a service request cannot be correctly tagged. By using adaptation knowledge of the CBR parser and the information stored in the retrieved cased the PR can be increase.

7: CONCLUSION

We proposed a method to interpret user's service requests by case-based approach. By the proposed approach, the system can transfer the entered string into goal models. Based on the goal model, the system can perform a series of actions to achieve the goal. Once the goal has been achieved, the user request is satisfied.

The experiment result shows that the case-based approach has best performance to adapt different inputs from different users.

ACKNOWLDGEMENT

This research was partly sponsored by Ministry of Economic Affairs under the grant 95-EC-17-A-02-S1-029 and National Science Council under the grants 95-2752-E-008-002-PAE and 95-2221-E-194-009.

REFERENCES

[1] J. Allen, *Natural language understanding 2^{nd} ed.*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, 1994.

[2] C.W. Thompson, P. Pazandak, and H.R. Tennant, "Talk to your semantic Web," *IEEE Internet Computing*, vol. 9, no. 6, 2005, pp. 75-78.

[3] C.H.L Lee and A. Liu "Model the query intention with goals," Proc. of the USW2005, Mar. 2005, pp. 535-540.

[4] C.H.L Lee and A. Liu "Toware intention-aware semantic Web services," Proc. of the IEEE SCC2005, July. 2005, pp. 69-76.

[5] C.D. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, 1999.

[6] H.M. Meng and K.C. Siu, "Semiautomatic acquisition of semantic structures for understanding domain-specific natural language queries," *IEEE trans. on Knowledge and Data Engineering*, vol. 14, no. 1, 2002, pp. 172-181.

[7] P. Loucopoulos and V. Karakostas, *System Requirements Engineering*, McGraw-Hill, London, 1995.

[8] C. Rolland, C. Souveyet, and C. B. Achour, "Guilding goal modeling using scenarios," *IEEE Tran. On Software Engineering*, vol. 24, no. 12, 1998, pp. 1055-1071.

[9] J. Lee, N.L. Xue and K.Y. Kuo, "Structuring requirement specifications with goals," *Information and Software Technology*, vol. 43, 2001, pp. 121-135.

[10] D.A. Grossman and O. Frieder, *Information Retrieval: Algorithms and Heuristics*, Kluwer Academic Publishers, Boston, 1998.

[11] W.W. Cohen, P. Ravikumar, and S.E. Fienberg, "A comparison of string metrics for matching names and records," In Proc. IIWeb 2003 (IJCAI2003 Workshop), 2003, pp. 73-78.

[12] M. Michelson and C.A. Knoblock, "Semantic annotation of unstructured and ungrammatical text," In Proc. of IJCAI2005, 2005, pp. 1091-1098.

[13] W. Wilke and R. Bergmann, "Techniques and Knowledge used for Adaptation during Case-Based Problem Solving", In Proc. IEA/EIA conference, LNCS, vol. 1416, 1998, pp. 497-506.

[14] I. Watson, *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann Publishers, California, 1997.

[15] L. Wu, L. Liu, J. Li, and Z. Li, "Modeling user multiple interests by an improved GCS approach," *Expert Systems with Application*, vol. 29, 2005, pp. 757-767.

[16] W. Mark, *Practical Artificial Intelligence Programming in Java*, 2005. Available at: http://www.markwatson.com/opencontent/.