Natural Language Based Heavy Personal Assistant Architecture for Information Retrieval and Presentation

Algirdas Laukaitis, Olegas Vasilecas, and Vilnius Gediminas

Technical University, Sauletekio al. 11,
LT-10223 Vilnius-40, Lithuania
{algirdas.laukaitis,olegas}@fm.vtu.lt

Abstract. In this paper we present the progress of the natural language usage as the paradigm for information extraction and presentation in the enterprise environment. Distributed heavy personal assistant architecture and its implementation is presented as the solution to overcome difficulties related of the natural language use in the information systems development. A new methodology based on connectionist and symbol processing techniques for a knowledge worker to process his documents and utterance is suggested. Then we suggest the results from those processes to reuse for new documents classification and generation of small atomic applications. Finally the experiment is presented. We compare Microsoft EQ, IBM WebSphere Voice Server NLU toolbox and our solution for concepts identification accuracy.

Keywords: Information extraction, Natural language understanding, Ontology, Personal assistant, Enterprise Semantic web.

1 Introduction

Significant part of the artificial intelligent (AI) research has been dedicated for the natural language formalization. Nevertheless natural language processing (NLP) and understanding (NLU) paradigms are still struggling to find their way into information systems (IS) development beyond documents indexing techniques. In that sense, the history of natural language database interfaces (NLDBI) can be a good example of NLP use as a programming paradigm in one of the "*simplest*" case, i.e., when the program is one structured query language (SQL) sentence (see [1] for the field review).

There are several projects that have been lunched recently in the effort to use NL as the programming paradigm but no one have reached the level where the others not involved in the project can verify achieved results. Actually, except Microsoft English Query [10], we haven't found any other products for the investigation. In this paper we attribute several factors that prevented natural language programming to mature up to the industrial implementation level:

1. ***Over-simplicity*** from the user's point of view and over-complexity from the implementation point of view. The users of IS are not very much interested with the systems that supports only one sentence utterance and one sentence program (i.e. SQL

M.J. Smith, G. Salvendy (Eds.): Human Interface, Part II, HCII 2007, LNCS 4558, pp. 54–63, 2007.

sentence). In our system we suggest to build atomic applications with the natural language and then stack those atomic applications to implement business solution.

2. ***Distributiveness*.** When we are dealing with the natural language formalization we are faced with the overwhelming state space search problem. An idea suggested in this paper is that users are building their local knowledge bases via communication with the personal assistant. Then, natural language utterance is projected on the set of those local knowledge bases.

3. ***Integration*** of NPL into the whole life cycle of IS development. About 15 years ago, Kevin Ryan claimed that NLP is not mature enough to be used in software engineering [13]. Nevertheless, Internet has boosted NLP research and nowadays the natural language formalization it is not seen as an unachievable goal. In this paper we extend the methodology suggested by Laukaitis et. al. [8] to combine connectionist and symbolic processing techniques for NLP integration into requirements engineering, conceptual modeling and final reuse in the IS interfaces.

4. ***Open source and free*** of charge vs. close and commercial solutions. Presented solution would be impossible without the support from the open source community. In early stages of the project we tried the solutions from Microsoft and IBM (both solutions are compared at the final section of the paper) but they have been discarded due the problems of integrity with the other modules of the system. Those are the main modules we reused form academic community: GATE - the framework of the natural language processing [2], WordNet dictionary [11], self-organizing map SOM toolbox [15] , Galicia - formal concept analysis software [9], JMining - framework of the web applications [9]. Economical soundness of the architecture suggested in this paper can be proved only by use of free software as we suggest to replicate typical enterprise server architecture to each personal computer of the knowledge worker.

By keeping in mind all those factors we state the following problem: *How to create an environment for developing business applications by use natural language*.

The solutions of stated problem organize the rest of the paper as follows. First, we present the general framework of the model generation system from the IS documentation and engineers utterance. Then we describe personal assistant for the creation of local knowledge basis and advocate for NL use in distributed enterprise computing. Next we introduce JMining framework and language to which natural language statements are transformed. NLP module is described in details in section 4. Finally an experiment is presented where we compare several solutions.

2 General Framework of the Solution

Conceptual models offer an abstracted view on certain characteristics of the domain under consideration. They are used for different purposes, such as a communication instrument between users and developers, for managing and understanding the complexity within the application domain, etc. The presence of tools and methodology that supports coordination of textual documents and local knowledge bases is crucial for the successful IS development. Even more, we can say that the essence of modeling is the ability of modeler to classify the textual data and then to present it by some formal modeling language. Then artificial intelligence technologies that will attempt to automate IS modeling and development must follow that cognition process of human modeler.

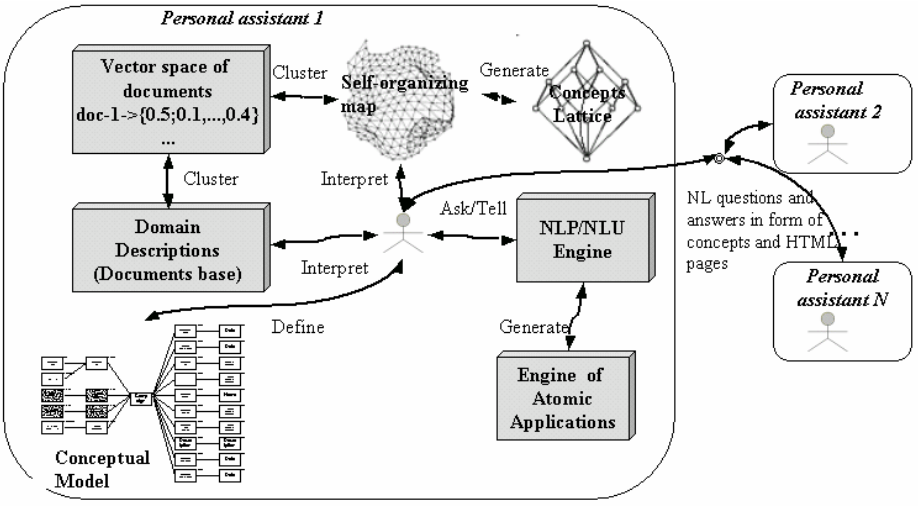

Fig. 1. Loopback of unstructured information flow

An overall process of automatically clustering the concepts and building knowledge basis is presented in Figure 1. First, the corpus is created form IS documents and utterance. Then, vector space of the corpus is created using natural language processing framework, domain ontology and WordNet ontology [11]. The self-organizing network [7] is build and used for cluster analysis. Next, with conceptual context and formal concept lattice [3] improvements are made in the understanding of clusters relationships. After the modeling stage the conceptual model self-organizing map and concept lattice are reused by natural language processing and understanding engine. The NLP/NLU engine combines map and lattice with such traditional NLP processing techniques like finite state traducer and gazetteer lookup.

Some philosophical arguments for the use of the connectionist paradigm in the context of information systems development can be found in the paper of Timo Honkela [5]. By the reference to the works of Von Foerster, he supposed that most information systems are developed as "trivial machines" to be predictable and controllable. All results in this paper can be interpreted as an implementation of these philosophical statements.

2.1 Heavy Personal Assistant

The idea behind heavy personal assistant architecture (HPAA) is to investigate the IS architecture that resembles society of humans in which natural language plays an important role. If the Figure 1 presents the general information flow processes in HPAA then the Figure 2 presents the technological implementation of the framework. As we can see, the central idea behind technological implementation of the heavy personal assistant is that traditional enterprise server architecture is replicated for each corporate personal computer to handle personal information flow and communication with other personal Web services in the Enterprise Semantic Web environment. We

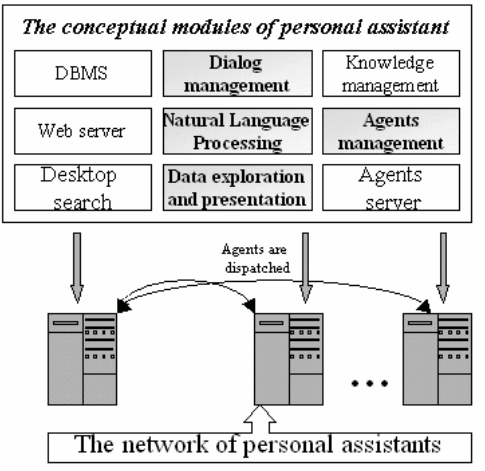

Fig. 2. Heavy personal assistant architecture

assume that all conceptual modules are installed on each personal computer involved in the information and knowledge management process. The grey shaded blocks represents the modules that have been originally build in current research project.

The white blocks represent software modules that we reused from open source community. All of them are free of charge (an important issuer if we are planning to install them on all working personal computers). The following example is the short description of the implementation we used in our framework (for more details we refer to [9]: 1. MySQL has been chosen as the database management systems. We store all annotated documents and instances of ontology classes in it. 2. We used the Apache Tomcat 5.5 Servlet/JSP container as the server of Web services. 3. Desktop search module represents software like Google desktop search. It's primary use in our architecture is to faster documents search for the agents that implements personal Web services. 4. For knowledge management we used *Protégé* application but problem with it is that it has difficulties for handling large OWL files.

3 JMining

As we mentioned above, our goal is the creation of the information systems developing environment by natural language. Previous section presented the framework for classification of the IS documents and utterance. By being able to identify classes and their semantics (i.e. relationships) we now present the JMining framework and language for the generation of business applications.

There were numerous attempts to use natural language as the programming paradigm. The most of them have concentrated on structural query language (SQL), however, some of them tried to lift a formal language representation complexity up to such languages like Java [12]. But neither of them have withstood more than few years. JMining language was created to overcome difficulties stressed by numerous projects in the field.

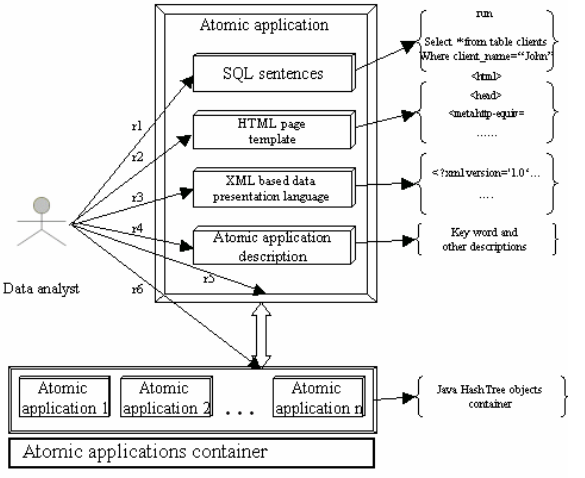

Fig. 3. Structures of the JMining framework

The central idea of the framework is an atomic application concept. An entire business solution is presented as a tuple of atomic applications liked by the URL references and sheared memory pool to store all parameter values entered by user. Figure 3 depicts the major structure of the atomic application. By its essence it is the small web application, which contains four components: {*database script, HTML page, data representation script (XML, XSL, etc.) and documentation page*}. No additional constrains are put on those components if they are accessed directly (i.e. manually corrected through Web Browser). If they are accessed using natural language interface, then all components must constrain their representation to the tuple calculus.

Each time the user enters NL sentence the system assumes that it is addressed to one of the six objects: container of the atomic applications, atomic application itself or one of the four objects that forms atomic application. Table 1 presents all state space variables that are used to control the system behavior. There are three futures of the JMining language that we attribute in the current research project:

1. *Expressiveness and complexity*. One of the reasons for the failure to lift NLDBI up to practical use is that most projects concentrated on the generation of one SQL sentence. However, practice shows that users are not interested in such interfaces and we solve this problem by addressing more objects than a single SQL sentence.

2. *Integrativeness*. No information systems development framework or programming language was developed to support IS development by means of NLP in all IS development life-cycle stages. In the previous section we introduced the modeling technique with the use of self-organizing maps. In the JMining language each atomic application has *documentation* object: the textual description of the atomic application. Each atomic application document has vector representation that can be measured by self-organizing map. Each time when the sentence is presented to the system it matches

self-organizing map fired neuron again atomic application vector. The closes match associates selected atomic application with the sentence.

3. *Distributiveness*. As we mentioned the state space produced by the natural language can be too huge to be matched by one ontology or conceptual model. In the JMining framework, if the personal assistant is not able to find the close match of the sentence in the local knowledge base then it tries to query other personal assistants. In the JMining framework remote personal assistant is using the same interface as the humans i.e. by using the Jakarta Commons HttpClient component it imitates Web Browser.

Table 1. The state space variables

Variable name	Example
agenda	Each item of the agenda is associated with some number: 0 - no agenda item selected, 1 - atomic application selected, 2 - get info from metadata storage, 3 - write script.
app. object	0 - no objects, 1-SQL, 2-HTML, 3-XML, 4-documentation
confidence	1 - if the object has been established, 0 -if not.
value_track	Tracks whether the system has obtained a value (no=0, yes=1).
number_of_ times	Tracks the number of times the dialog manager has asked for an attribute.

Primary the JMining has been developed to attribute IS development using only Web Browser on the "*thin*" computational device like mobile phone. And this means that all created atomic applications can be corrected by manually editing generated Jmining script using Web Browser.

4 Natural Language Processing

In this section we present NLP techniques because it is a core module in the HPA architecture. Figure 4 shows the main steps in the NLP processing. The goal of the engine is to produce vector representations and ontology annotation sets for each sentence that passes it. The vector space model (VSM) for documents transformation to the vectors is a well-known representation approach that transforms a document to a weight vector in automatic text clustering and classification. The method is based on the bag-of-words approach, which ignores the ordering of words within the sentence and uses basic occurrence information [14]. The NLP processes are depicted in Figure 4 and the main processes are described in details below.

GATE – general Architecture for Text Engineering - is a well-established infrastructure for customization and development of NLP components [2]. We briefly describe modules used in our research for building concepts vector spaces. The Unicode tokeniser splits the text into simple tokens and is used for the next steps of the natural language processing. The tagger is a modified version of the Brill tagger, which produces a part-of-speech tag as an annotation on each word or symbol. We

used it to extract nouns and verbs and remove all other words from the dictionary. The gazetteer further reduces dimensionality of the documents corpus prior to classification. It uses the lists of named entities and annotates text with class labels such as cities, organizations, days of the week, etc. We replaced each named entity with the label of the class. Semantic tagger - provides finite state transduction over annotations based on regular expressions. It produced additional set of named entities and we replaced each named entity with the class label. Orthographic Coreference - the module adds identity relations between named entities found by the semantic tagger. Reduction of the state space dimensionality is achieved by replacing marked tokens with named entities class labels found by the semantic tagger. SUPPLE is a bottom-up parser that constructs syntax trees and logical forms for English sentences. We used it only to remove tokens not annotated by this module. All modules within the GATE produced annotations - pairs of nodes pointing to positions inside the document content, and a set of attribute-values, encoding linguistic information.

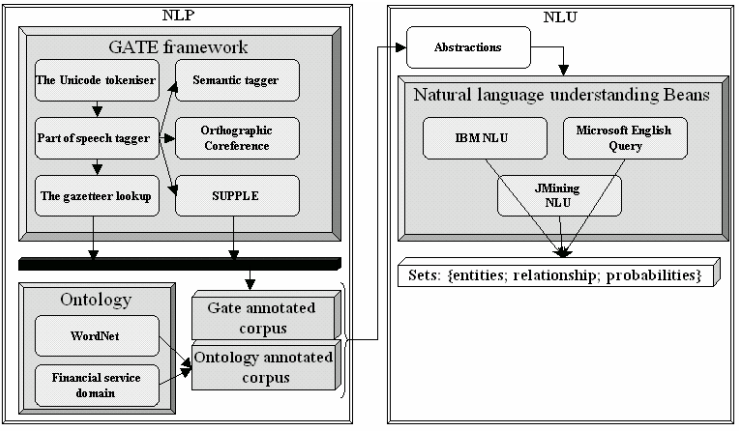

Fig. 4. Processes of the natural language processing and natural language understanding

Abstraction. The basic idea of the abstraction process is to replace the terms by more abstract concepts as defined in a given thesaurus, in order to capture similarities at various levels of generalization. For this purpose we used WordNet and annotated GATE corpus as the background knowledge base. WordNet consists of so-called synsets, together with a hypernym/hyponym hierarchy [4]. To modify the word vector representations, all nouns have been replaced by WordNet corresponding concept ('synset'). Some words have several semantic classes ('synsets') and in that case we used a disambiguation method provided by WordNet - the 'most common' meaning for a word in English was our choice. The words replaced by the GATE named entities annotation scheme were not included for the WordNet processing.

Vectors space. In our experiments we used vector space of the terms vectors weighted by *tfidf* (term frequency inverse document frequency)[14].

5 Experiment

In the previous sections we have shown how to build local knowledge bases for personal assistant and how to share those bases in the distributed enterprise environment of the personal assistants. In this section we provide description of two experiments we conducted to provide the answers for the following questions: 1. *What is rate of concept identification accuracy from user's utterance?* 2. *If the following hypothesis have any sound meaning? H0: If we measure the number of correctly answered questions of personal assistant to the questions brought by other personal assistants and then reword the knowledge worker to the extend of personal assistant performance, can we expect that the knowledge worker will be more interested in teaching it personal assistant.*

To answer the first question we conducted the following experiment. The Microsoft English Query [10] and IBM WebSphere Voice Server NLU toolbox V5.1 [6] have been chosen as benchmarks for the comparison with suggested solution in this paper. Microsoft product has been chosen because it's demo version is freely available and it comes as the most user friendly product available on the market (it seems that currently there is no other product left as several authors and companied declined our request to test their solutions). Comparison between Microsoft and IBM was interested because Microsoft utilizes rule based approach (symbolic processing) and IBM utilizes statistical approach (close to the connectionist paradigm). From the IBM presentation [6] it appear that the system is primarily intended to support database interfaces in the telecommunication market. It was a challenging task to test it on the more complex system e.g. a full Enterprise conceptual model for the financial market.

Table 2. Concept identification comparison between tree approaches and with two conceptual models

	Northwind database	CN=9	CN=50	CN=200	CN=400	CN=500
IBM NLU	30.82	36.82	17.26	14.82	11.15	8.22
Microsoft EQ	80.46	-	-	-	-	-
JMining	28.93	36.47	21.49	17.85	14.88	11.41

After the training process the following experiment has been conducted. A group consisting of 9 students has been instructed about two database models: 1) Northwind example comes with Microsoft EQ demo and IBM financial services conceptual model (see[9] for more details). They queried the system with about 20 questions and tried to identify the "*customer*" concept. For the Northwind example the second column in the table 2 shows the percentage rate of correct answers. No surprise that Microsoft scored the best as the model roles have been accurately written for this example. On the other hand for the second example where we interactively increased the set of concepts from 9 up to 500 there was no way to teach Microsoft approach due to limited resources. To answer the second question we made the following

experiment with 3 students group. The 50 $ was the budget distributed proportionally with the knowledge contributed by the user working environment to the other users' personal assistants. The students where instructed about the IBM financial service model. At the first stage nor reword has been mentioned. As the consequence the personal assistants have shown very weak performance. As the reword has been introduced we noticed that the performance of the personal assistants increased about 50{\%}. Finally, we admit that the current experiment is not very sound but we conducted it anyway because question how to reword people for teaching their robots can be interesting for future research in the field of artificial intelligence.

4 Conclusion

Natural language based IS development can supplement traditional IS development approaches. In this paper we have shown how connectionist and symbol-processing paradigms can be combined within one single framework for information systems development. Distributed personal assistants architecture has been suggested to utilise this framework and create new working environment for the knowledge based professions. Provided evidences have shown that high quality documentation can be reused in all stages of the information systems development and increase company productivity.

References

1. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Time, Tense and Aspect in Natural Language Database Interfaces. Natural Language Engineering 4, 229–276 (1998)
2. Cunningham, H.: GATE, a General Architecture for Text Engineering. Computers and the Humanities 36, 223–254 (2002)
3. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin-Heidelberg (1999)
4. Hofmann, T.: Probabilistic latent semantic indexing. In: Research and Development in Information Retrieval, PP. 50–57 (1999)
5. Honkela, T.: Von Foerster meets Kohonen - Approaches to Artificial Intelligence, Cognitive Science and Information Systems Development. Kybernetes 34(1/2), 40–53 (2005)
6. IBM Voice Toolkit V5.1 for WebSphere Studio. Accessed (January 2007) http://www-306.ibm.com/software/
7. Kohonen, T.: Self-Organizing Maps. Springer-Verlag, Heidelberg (2001)
8. Laukaitis, A., Vasilecas, O.: Self-organizing map for conceptual modelling. In: Proceedings of the International Conference on Computer Systems and Technologies CompSysTech'06, pp. 141–147 (2006)
9. Laukaitis, A., Vasilecas, O., Berniunas, B.: JMining – information delivery web portal architecture and open source implementation. Information Systems Development. Advances in Theory, Practice and Education. Springer Science, pp. 199–206 (2005)
10. Microsoft corporation. SQL Server and English Query. Accessed (January 2007), http://msdn.microsoft.com/
11. Miller, G.A.: WordNet: A Dictionary Browser. In; Proc. 1st Int'l Conf. Information in Data, pp. 25–28 (1985)

12. Price, D., Riloff, E., Zachary, J., Harvey, B.: NaturalJava: A Natural Language Interface for Programming in Java. In: the Proceedings of the International Conference on Intelligent User Interfaces (2000)
13. Ryan, K.: The role of natural language in requirements engineering. In: Proceedings of IEEE International Symposium on Requirements Engineering, pp. 240–242. IEEE Computer Society Press, Washington (1993)
14. Salton, G.: Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer. Addison-Wesley, New York (1989)
15. Toolbox, S.O.M. Accessed (January 2007), http://www.cis.hut.fi/projects/somtoolbox/
16. Valtchev, P., Grosser, D., Roume, C., Rouane, H.M.: GALICIA: an open platform for lattices. In: de Moor, A., Ganter, B. (eds.) Using Conceptual Structures: Contributions to 11th Intl. Conference on Conceptual Structures, pp. 241–254 (2003)