EVALITA2009 UNINA Spoken Dialogue System

Gianluca Mignini and Francesco Cutugno

[1] Dipartimento di Scienze Fisiche – NLP Group Università degli studi di Napoli Federico II
{mignini, cutugno}@na.infn.it

Abstract. We present here the SDS system realised from NLP group of the University of Naples Federico II for EVALITA 2009. The system is based on the VoxNauta Loquendo^{TM} platform, implements dynamic database access, automatic VXML grammar generation. As far as semantic domain for SQL syntax is concerned, the system makes use of an ontology where synonyms and hyperonyms of reserved words equivalent to query actions are stored.

Keywords: Spoken Dialogue Systems, ontology, mixed initiative.

1 Introduction

In this paper we are presenting a mixed initiative spoken dialogue system for information retrieval in the sales force domain. The application models the interaction between a "human" salesman and his company, which he calls to retrieve information about customers, products and orders, and to open new invoices [1].

The Spoken Dialogue System (SDS) has been developed using JSP technology to create dynamic VoiceXML pages and to interact with a MYSQL database and an OWL knowledge base. The Voxnauta^{TM} platform by Loquendo [2] made available Text To Speech synthesis and Automatic Speech Recognition modules.

Spoken dialogue systems has been classified into three types [3] : system initiative, user initiative and mixed initiative. Commonly, most of the voice application are system initiative based, in which dialogue is controlled by the system and the user is only asked to explicit information sequences needed to complete a task. The main advantages of these kind of system are efficiency and robustness, specially in applications unknown by the users, but they are not suitable for advanced users and scenarios in which time is critical and there is no willing to fill single fields with multiple utterances. Another limiting aspect is that command recognition of information submitted by users is based on statical grammars hard to update.

Our mixed initiative approach gives dialogue initiative to the user, and the system eventually takes control to help user to explicit correctly his request. The application grammar is dynamically created extracting from the database information about the request object and from the knowledge base all the available expressions related to two classes/types of requests: insertion or selection. A middle layer decision block use expressions from the knowledge base to classify the request either as an insert

operation (i.e. for new orders) or a query operation (i.e. asking data about customers, products or open invoices) on the database. Another module, the acquisition block, use the decision block request interpretation to interact with the database and return the results.

The ontology base makes the system more flexible, being easily extensible using tools that don't require any domain or programming knowledge, and is reusable in further applications.

Besides, the dynamic main grammar ensures the data consistency after any update.

2 Application Architecture

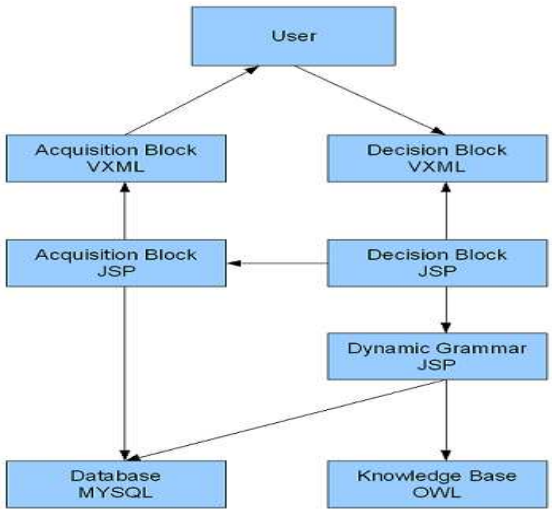

Fig. 1. Application architecture block schema

Figure 1 shows the Evalita Unina SDS modular architecture. It is composed by two main JSP blocks (Decision and Acquisition) to interact with the user, and an underlying JSP main grammar to gain access to updated data from database and knowledge base.

The decision block creates a VXML page which contains the main application dialogue. Here users make their requests and the system assign one of the 8 available scenarios by means of interpretation tags in grammar semantics. This dialogue is thought as having mixed initiative features, so if system doesn't catch all the information needed to complete a scenario, it will ask for filling the missing or misunderstood fields. In next section we will see the reserved words used to activate different scenarios. Once all required information are obtained, these are passed to the Acquisition block, whose main task is to translate them into SQL query. So these

queries are submitted to the MYSQL database and results are suited for presenting to the user.

3 Owl Ontology

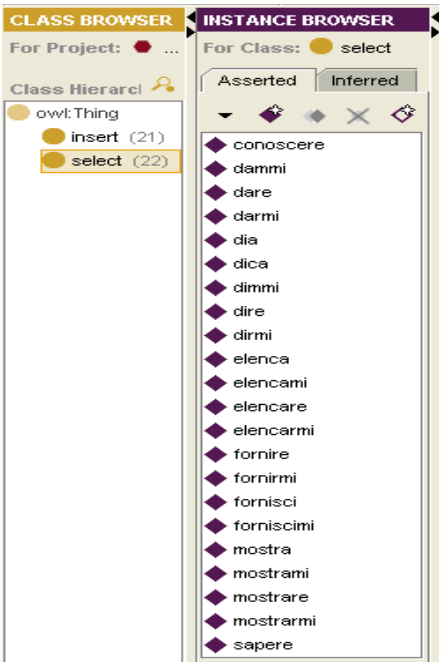

Fig. 2. Select class instances

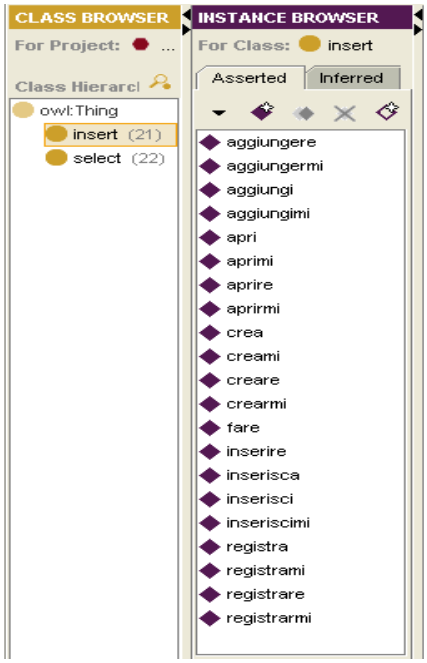

Fig. 3. Insert class instances

We have used *Protégé* [3], a free open source ontology editor as ontology-developing environment. Our purpose is to create an ontology-based dictionary of terms to classify user's requests in the form of either insertions or query operations on the database. The ontology is composed of two classes (Select and Insert), each one populated with instances containing some verbs commonly used in Italian language to express respectively the purpose of receive information and create invoices. These instances are used by the ABNF main grammar coupled with the reserved word *GARBAGE*, so we focused on explicating the main verb conjugations, disregarding how is used in the sentence context. In this way we don't bother if user made a request such as “can you tell me...”, “I'd like you to tell me...”, “please tell me”, etc. since he said the word “tell” contained in the knowledge base and associated to a request of information. The figures 2,3 show the content of our ontology-based dictionary, easily upgradeable with any ontology editor.

4 Application Features

After the identification of the caller as salesman, Evalita Unina SDS supports 8 different scenarios:

1. Ask about customer information (general information or specific such as address, shop name, VAT number, National Identification number).
2. Ask about product information (general information or specific such as category, price and discount).
3. Ask products catalog.
4. Ask customers list.
5. Open new orders.
6. Ask open invoices for a customer.
7. Ask for help.
8. Close the application.

Table 1. Supported utterances and related scenarios

1	SELECT **…** **[** informazioni specifiche] … (“cliente” | $nome $cognome) *SELECT … [specific information] … (“customer” | $name $surname)*
2	SELECT **…** **[** informazioni specifiche] … (“prodotto” | $nome $marca) *SELECT … [specific information] … (“product” | $name $brand)*
3	SELECT **…** (“elenco” | “lista” | “catalogo” | “tutti”) … “prodotti” *SELECT … (“list” | “catalog” | “listing” | “all”) … “products”*
4	SELECT **…** (“elenco” | “lista” | “tutti”) … “clienti” *SELECT … (“list” | “listing” | “all”) … “customers”*
5	INSERT **…** (“ordine” | “acquisto” | “partita”) ... [“cliente” | $nome $cognome] *INSERT … (“order” | “purchase” | “invoice”) ... [“customer” | $name $surname]*
6	SELECT … (“ordini” | “acquisti” | “partite”) [“cliente” | $nome $cognome]*SELECT … (“order” | “purchase” | “invoice”) ... [“customer” | $name $surname]*
7	“aiuto” *“help”*
8	“arrivederci” | “esci” | “fine” *“goodbye” | “exit” | “end”*

Each scenario is activated pronouncing some reserved words in the main dialogue, where the identified salesman is asked to choose an action the system performs. A relevant feature available in our model is the immediacy of the interaction. In fact, thanks to mixed initiative, every scenario, except “open new orders”, can be completed in only one conversation turn, if properly submitted. The absence of a request for confirmation in querying scenarios makes the system faster at the expense of robustness, but it seems to be a reasonable compromise for a limited domain like this. For sake of data protection, we choose confirm only operation that must be executed in the “new order” scenario, which applies changes to the database.

Despite Unina Evalita SDS is mixed-initiative based, the application switch to system-initiative if users are not able to express their requests in the expected way two consecutive times. This feature makes easier both expert and novice callers to use the system. Besides, in presence of high environmental noise or personal preferences, users can anytime choose the scenario by using DTMF with a dial tone phone.

Insert and Select are intended as ontology instances of the respectively classes, while "$" prefixed words are database instances. Square brackets means optional sentences and round brackets means mandatory sentences. The "|" symbol stand for "or" clause.

5 Conclusions

System global performances and details on single subtask represented by different scenarios, are reported in table 2 and 3. The average dialogue duration is 11 turns, and task success rate for the system is 58,4%. The results indicate a good behavior in dialogue duration, since each evaluation scenario included 4-5 tasks and was completed in about 2.4 turns per task. Meanwhile, task success rate may be improved, but we have to point out that some tasks included in evaluating scenarios were not supported by our application, since not defined in Evalita guidelines but included in the final reports.

Table 2. Dialog level statistics: Dialog average duration in seconds and turns.

System	Duration (sec)	Duration (# Turns)
UniNA SDS	145.8±72.7	11.0±5.7

Table 3. Task level statistics: Task Duration and Task Success Rate

Task	Duration (turns)	Success Rate (corr/req)
Identify representative	1.9 ± 0.4	100.0% (19/19)
Ask customer detail	2.0 ± 0.0	83.3% (5/6)
List orders	2.5 ± 1.5	0.0% (0/8)
Show last order	2.0 ±0.0	100% (1/1)
List customers	2.0 ± 0.0	50.0% (2/4)
New order	4.6 ± 1.5	36.4% (4/11)
List products by category	3.0 ± 1.0	14.3% (1/7)
List products by brand	-	-
List products - other	2.0 ± 0.0	0.0% (0/4)
Search single product	2.3 ± 0.4	55.6% (5/9)
Ask for help	2.0 ± 0.0	100% (3/3)
Exit application	2.5 ± 0.5	100.0% (5/5)
Overall (corr/req)	-	**58.4% (45/77)**

References

1. Baggia P., Cutugno F., Danieli M., Pieraccini R., Riccardi G.: Evalita 2009 – Spoken Dialogue System Evaluation task: Detailed guidelines, http://evalita.fbk.eu/dialogue.html (2009)
2. Loquendo. http://www.loquendo.com
3. Jurafsky D., Martin J.H.: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall, Upper Saddle River (2008)
4. Protégé, http://protege.stanford.edu