Semi-automatic acquisition of command grammar

Thierry Poibeau* and Bénédicte Goujon**

* Laboratoire d'Informatique de Paris-Nord — CNRS UMR 7030 and Université Paris 13
99, av. JB Clément F-93430 Villetaneuse
thierry.poibeau@lipn.univ-paris13.fr

** Thales Research and Technology
Domaine de Corbeville F-91404 Orsay
benedicte.goujon@thalesgroup.com

Abstract

This paper presents an original strategy for the production of command grammars for complex systems. We show that command grammars describe a sub-language that can be processed by local syntactic rules and compositional semantic strategies. The paper shows how prepositional attachment ambiguity can be handled by such a system.

Introduction

Vocal interfaces offer an appropriate way to access information from complex systems. People have demonstrated that voice is more efficient, in such systems, to perform complex tasks, than using keyboard and mouse (Carbonnell and Dauchy, 1999). However, in order to be efficient, vocal interfaces should take into account various ergonomic constraints:

- An appropriate understanding of vocal utterances,
- Linguistic variation handling,
- Reduced processing time (Scapin, 1986).

In order to fulfil these requirements, systems generally include lexical data and grammars of the concerned domain in order to control what can be said in input and analyse this flow of data according to the model of the application.

Command grammars are complex systems requiring domain vocabulary and attested linguistic structures. Moreover, for each utterance, a set of possible variations must be specified: the system should be able to deal with implicit information and solve incomplete utterances. Lastly, the system should be able to send a warning when the utterance is not recognized with sufficient probability to infer a correct understanding. All these reasons explain why command grammars are most of the time written by hand from a model of the domain or from a set of example utterances.

The portability of command grammars is limited by different constraints: the grammar is generally represented by an unordered list of rules that mix syntax and semantics. The knowledge base used by the vocal system generally duplicated the knowledge base of the application, since formalisms are much complex and could cause some coherence and update problems. It is generally the work of professional analysts to develop resources (linguists, domain experts). The attempts to transfer this part of the development process to operational officers have always failed due to the complexity of the task.

This paper is focused on this question. We propose some elements for a better automation of the process. We propose to automatically derive a part of the knowledge required by the grammar elaboration phase from application models and from the set of examples provided by domain experts. We defend the hypothesis that command grammars can be assimilated to sub-languages. This sub-language can be described with a limited set of rules that are incrementally applied to the input utterances. The semantics attached to each utterance is calculated simultaneously from (non monotonic) compositional rules. This paper completes proposals presented in Poibeau and Goujon (2004) for French: we show that the model initially proposed for French can valuably be applied to English.

In the following section, we present some works that are partly similar to ours. We then detail the notion of sub-language applied to command grammars and we propose a dynamic model based on the knowledge base of the application. We then present several examples and an operational implementation of the system.

State of the art

Many authors have already studied the generation of command grammar from formal models. Mathieu *et al.* (2001) propose a general model in which a grammar is instantiated by a independently-elaborated semantic lexicon. This approach is interesting since the lexicon is not mixed with syntax, but several experiments have shown that abstract syntactic constructions elaborated apart from the application are, most of the time, too general to encode all data required for a given application. It is then necessary to modify the grammar model for each new application.

Our aim is to develop grammars that can easily take into account the domain and the application. Brown and Burton have addressed this kind of problems in 1976, in the Semantic Grammar framework. Several applications have been developed on top of this theory; among others the MYCIN expert system for NLP based interfaces. The limit of this approach is the fact that the description of the grammar and of the semantic lexicon was at the same time too costly and hardly re-usable.

Thot *et al.* (2002) propose a process to automatically generate grammars to query a database using NLP techniques (in English; but the process could applied to other kind of languages). An abstract grammar is developed with gaps that are filled by named elements from the database (table names, field names…). This application is interesting since there is a clear link between the application model and its interaction tier. However, this experiment is limited to databases. The aim

of this paper is to extend this kind of systems to classical vocal interfaces integrated on top of complex systems from the industry.

Command-grammar as a sub-language

Utterances found in command and control applications are very specific compared to everyday conversation. Utterances are above all orders that are expressed with imperative or infinitive clauses.

```
Zoom
Display the list of flights
```

Verb objects are simple or complex noun phrases. Thus, one of the major challenges for such grammars is to choose a good segmentation and solve ambiguities between possessives phrases and direct or indirect objects.

```
Display the list of flights from Paris
Display the list of flights on the screen
Display the list of arrival flights
```

The above-mentioned examples show some typical problem of prepositional phrase attachments in English. This problem is even more crucial in analytic language like French, in which most syntactic objects are introduced by the French preposition *à*. Sequences such as *display the list of the flights leaving from Paris or display the list of the flights to Paris* include possessive phrases attached to the noun phrase *flights* whereas *display the list of the flights on the screen* includes an indirect verb object (*display ~ on the screen*).

Information about verb sub-categorization frames, about preposition semantics and about the application model are necessary to process most of the ambiguities. For example, on the one hand, the `flight` object has properties like a `departure time`, an `arrival time`, etc. On the other hand, the `flight` object does not have a `screen` (of course a plane can integrate some screens but that point is not relevant from our point of view).

We have shown that a command grammar can be assimilated to a sub-language in which only a part of everyday language complexity is involved. This description, based on restricted syntactic and semantic regularities, is coherent with previous works form Mathieu *et al.* (2001). This kind of approach is most of the time based on a first phase using a corpus to acquire syntactic regularities by means of machine learning techniques. This point is relevant since, in complex industrial environments, end-users have to provide a set of key structures to the analysts who develop the project in coordination with application experts. The application database is then viewed as a reservoir of knowledge including a semantic dictionary for domain objects and their respective relations. For example, in the framework of navigation-simulated applications, all the environment description is included in a repository (application database). All information concerning the application objects is encoded in this database.

Formal data models

Data that are manipulated through an interface are generally stored in the application database. This database includes a formalized image of the application.

A database is a set of objects. Objects include attributes, which have values. Some values can be assigned by default (from the interface of the system, it is possible to specify a set of numeric values, a period of time, a symbolic value, etc.). The major part of the actions made through an interface must have access to at least a part of the application data.

An action can then be viewed as a process which aim is to modify an object or its values. A lot of formal representation systems have been developed in the 1990's, mainly for software development. The knowledge representation community has produced several studies concerning various formal frameworks, the MADS, KOD or KADS methods, among others (Guarino, 1995).

The main issue of this kind of system is to establish a connection between the formal domain (the formalized application database) and the set of utterances (natural language clauses used to designate these objects). That is the main reason why we propose to directly integrate, in the application database, information about possible linguistic realizations of a given piece of knowledge.

Incremental analysis

Thanks to the linguistic analysis, formal representations of utterances are built from the utterances.

Grammar rules

Grammar rules are intended to group together linguistically related set of words. The analysis is performed by different rules that are grouped by priority order. If a rule cannot be applied to an utterance, the system tries some other rules in order to obtain a complete analysis of the utterance. If the result is a partial analysis, the process has failed. This is briefly the different kind of rules:

- Simple syntactic clauses are isolated along with some propositions (relative clause, completive clause)
- The sub-categorization frame of the syntactic head (the verb, most of the time) is first taken into account. Thus, utterances are firstly segmented in accordance with this sub-categorization frame (several segmentations can be proposed in case of ambiguity)
- Possessive phrases introduced by of (in French) are arbitrary attached to their left-hand side syntactic segment. The same strategy is applied to French for the prepositions *à* and *de*.
- Syntactic phrases introduced by other prepositions are analysed according to the semantic value of the complement and of its introductive preposition. Preposition semantics is registered in a table in accordance with the proposals from Jensen and Nillson (2003). Due to lack of place, we do not describe this last point in this paper.

The system tries to build at the same time a syntactic and a semantic representation of the utterance. This technique is close to the one used in grammar formalisms like LFG (with the c-structure and the f-structure, Bresnan 2000) or to the TAGs (with the synchronous tags formalism, Shieber and Schabes 1990).

We will describe some examples based on syntactic phrases introduced by ambiguous preposition that are the

most problematic ones for ambiguity resolution. The semantic value of the syntactic phrase is calculated in connecting the named elements from the utterances and the application knowledge base. From this point of view, a sequence can correspond to an attribute or a value, to an object description, to the instantiation of the argumental structure or to a complex predicate, etc.
The semantic compositional analysis is not monotonic. Some elements produce specific logical form according to their semantic value, among others complex determiners and quantifiers. For example, *the list of all the flights* (*la liste de tous les vols* in French) means: "query the database and search for all the objects which type is `flight`" (see below). The same model is applied for quantifiers and operations on sets of objects. Quantifier focus analysis is limited given that most of the time, in this kind of sub-language, the quantifier only concerns the syntactic phrases immediately following the quantifier itself.

A simple analysis example

The verb of the sentence is first isolated. Objects are then analysed and indirect objects are typed according to their introductive preposition (minimal phrase analysis).

```
(V Display) (NP the list) (PP of the flights)
(PP leaving from Paris)
```

If a verb sub-categorizes an object introduced by a registered preposition and if a syntactic phrase is found being introduced by this preposition, the sequence is segmented according the verb sub-categorization. Otherwise, syntactic phrases introduced by light prepositions (of in English, *à* or *de* in French) are by default attached to the preceding group. Ambiguity is reduced since most syntactic groups are then related to assumed syntactic heads.

```
(V Display) (NP (NP the list) (PP (PP of the
flights) (PP leaving from Paris)))
```

At the same time, the system search for correspondences between application objects and linguistic designations in utterances in order to dynamically build logical forms. For example, *Paris* is identified as a location name, *leaving from* correspond to an attribute from the object `flight` and the string *flight* matches the object `flight` (from the application knowledge base). The following structured can be generated from the text:

```
(PP leaving from Paris)
departure = Paris

(PP (PP of the flights)      (PP leaving from
Paris))
Flight: departure = Paris

(NP (NP the list) (PP (PP of the flights) (PP
leaving from Paris)))
∀x such as x = (Flight : departure = Paris)

(V Display) (NP (NP the list) (PP (PP of the
flights) (PP leaving from Paris)))
Display ∀x such as x = (Flight: departure =
Paris)
```

Words related to grouping of objects (*the list of*, *the set of*) are analysed according to pseudo-logic forms from which the knowledge base can be queried.
The list of application objects has to be known from the system. For safety reasons (vocal interfaces can be implemented on top of complex and sensitive systems), an analysis is not validated of the head of a syntactic phrase not has been recognized (but the application can be tuned so that an utterance is acceptable even if some elements from the syntactic analysis are not recognized).

Complex verb sub-categorization frames

When the verb sub-categorizes several syntactic elements, the segmentation phase is intended to process these elements first. In the example *Move the tank from point alpha to point beta*, the verb *move*r requires three arguments: the object that has been moved (*the tank*), the starting point (*point alpha*) and the ending point (*point beta*).

```
(V Move  (NP the tank) (PP from point alpha)
(PP to point beta))
```

Several partial analyses are process at the same time in case of ambiguity. Segmentation faults are solved during the analysis: if no logical form can be obtained from a given segmentation, the analysis process rejects the former. The decision is only taken once the system has access to the whole necessary information.

Discontinuous expressions

A sequence like *Display the list of the flights on the* screen (the same structure can be found in French *Afficher la liste des vols à l'écran*) causes a problem if the noun phrase *on the screen* is not included in the knowledge base as related to the verb. The following analysis is not possible:

```
(PP (PP of the flights) (PP on the screen))
flight = screen
```

because flight and screen are not a valid attribute-value pair. For the sequence *Display the list of the flights on the* screen to be valid, the complex lexical entry *Display ~ on the screen* must be registered in the application knowledge base. When the system segments the sentence in minimal chunks, a pre-processing step is intended to group together compound words and other complex syntactic phrases. The following segmentation is thus obtained:

```
(V Display_on_the_screen) (NP the list) (PP of
the flights)
```

In the end, given that the semantic value of *Display on the screen* is the same as the one of *Display* (the two expressions are synonyms) the noun phrase is deleted from the logical form so that the same result is obtained as for *Display the list of the flights*.

Attribute name deduction from utterance

Human discourse generally includes ellipses, that is to say non-explicitly specified elements. For safety reasons, most implicit elements are not allowed in command and control grammars: when commanding a complex system, all pieces of information are required and must be given explicitly. However, the system allows a limited number

of ellipses when they do not introduce ambiguity in the discourse and when they produce a more natural conversation style.
The ellipses allowed by the system mainly concerned attribute names. For example, if someone says *the two planes*, the system can easily infer that the number refers to the number of objects designated by the noun phrase. For a sequence like *the grey planes*, the information that *grey* is a colour is found in the application database. If the noun *grey* was not known by the system, the analysis would failed since the value could not be related to an attribute name.

Syntactic variations

A set of variants is integrated to the system for the analysis of sentences such as *I want to zoom*. Most of these variants can be represented as adjuncts (*I want to*) to simple basic forms (*zoom*). These adjuncts do not modify the semantic value of the original form on which they are added. Their deletion allows to go back to a sequence recognized by the application grammar.

Implementation

The proposed models could be entirely dynamic. Utterances are segmented and analysed from generic analysis rules and the logical form elaborated *on the fly* from information contained in the application database. However, practically speaking, most analysers need a static knowledge base. The explicit production of the application grammar is also useful to check the set of rules and modify them whenever necessary.
The generation of the set of grammar rules can however cause some maintenance problems. If some modifications are introduced manually in the grammar, they will be lost when a new version of the grammar will be automatically generated from the application database and the set of abstract un-instantiated grammar rules (which is necessary when the application models change).
The grammar obtained from the method hereby described is independent from the analyser: the system must only be compliant with the input format of the analyser. A wrapper can be added at the end of the generation phase in order to produce grammars in an appropriate format. The generation phase is largely independent from the application knowledge base.
The overall result is comparable with a manually generated grammar. However, the automation of the process produces more coherent and homogeneous grammars. Lastly, the analysis of a sample of utterances provided by the end-user allows to check if the grammar can analysed and produced valid logical forms from application utterances.

Conclusion

This paper describes a system that semi-automatically produces grammars for vocal interfaces of complex systems. The system uses, on the one hand, the application model and knowledge base and, on the other hand, a set of un-instantiated grammar rules. The compilation of these two sources generates some easily adaptable application grammars. Logical forms are built during the analysis so that most ambiguities can be solved dynamically, when the system has relevant information available. The overall semantics representation process is most of the time compositional but we have also described some non-monotonic cases.

References

Bonnet A. (1980) "Les grammaires sémantiques, outil puissant pour interroger les bases de données en langage naturel". In *R.A.I.R.O. Informatique*, vol. 14, n°2, pp. 137– 148.

Bresnan J. (2000) *Lexcical functional syntax*. Blackwell. Oxford.

Carbonell, N. et Dauchy, P. (1999). "Empirical data on the use of speech and gestures in a multimodal human-computer environment". *Proceedings HCI International'99*, Munich, pp. 446-450.

Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *International Journal Human-Computer Studies*, 43(5). 625--640.

Jensen et Nillson (2003) « Ontology-Based Semantics for Prepositions ». *Proceedings of the ACL-SIGSEM workshop on the syntax and semantics of prepositions and computational linguistics applications*, Toulouse.

Mathieu F.A., Surcin S. et Sedogbo C (2001) « Un système de commande vocale multimodale : ThomSpeaker". *Technique et science informatiques*. Vol 20, n°3/2001, p.337-368.

Poibeau and Goujon (2004). Génération semi-automatique de grammaires de commande vocale. Journées d'Etudes de la Parole (JEP 2004). Fes (Marocco).

Scapin, D.L. (1986). *Guide ergonomique de conception des interfaces humain-machine*. INRIA.

Shieber S. and Schabes Y. (1990). Synchronous Tree Adjoining Grammars. In *13rd Computational Linguistics* (COLING 1990). Helsinki. Pp. 253-260.

Toth A.R., Harris T.K. , Sanders J., Shriver S., Rosenfeld R (2002) Towards every-citizen's speech interface : an application generator for speech interfaces to databases. *ICSLP 2002*. Denver. pp. 1497-1500.