Procesamiento del Lenguaje Natural, num. 29 (2002), pp.189-196 recibido 24-05-2002; aceptado 02-07-2002

Identification and Representation of Anaphoric Expressions in a Spoken Dialogue System

Esther Sena, Gabriel Amores, Jose F. Quesada
Universidad de Sevilla
Julietta Research Group in Natural Language Processing
esther@fing.cica.es, jgabriel@us.es, jquesada@cica.es

Resumen: Este artículo describe una estrategia para la identificación y representación de expresiones anafóricas en un sistema de gestión de diálogo hablado. Nuestro principal objetivo en este trabajo se ha centrado en la representación tanto de expresiones anafóricas como cuantificadas dentro del módulo de comprensión de lenguaje natural (NLU), para permitir la resolución de la anáfora. Sólo aquellos casos que no pueden ser resueltos por el módulo NLU pasan al módulo de Gestión del Diálogo, donde usando la historia del diálogo para resolver la anáfora a partir de la información suministrada por el módulo de Interpretación Semántica.
Palabras clave: Sistemas de gestión de diálogo hablado, Aplicaciones de PLN, Identificación y representación de expresiones anafóricas

Abstract: This paper proposes a strategy for the identification and representation of anaphoric expressions in a spoken dialogue system. Our main goal in this paper is to show the representation of both anaphoric and quantified expressions within the NLU Module, in order to be able to resolve the anaphora. Only those cases that cannot be resolved within the NLU Module, will go to the Dialogue Management Module, where the dialogue history will be consulted, taking into account the information previously provided by the Semantic Interpreter.
Keywords: Spoken Dialogue Systems, NLP Applications, Anaphora Identification and Representation

1. Introduction

One of the main goals in the design and implementation of human–machine interfaces and spoken dialogue systems is to achieve a natural and flexible interaction between the user and the system. The achievement of these goals goes hand in hand with the use of both complex linguistic phenomena and the appropriate mechanisms to cope with the features of natural spoken language. Some of these features are anaphoric and quantified expressions, the use of subdialogues, false starts, parameter and command accommodation, requests, coordination, error repairs, syntactic disorder, parameter negotiation, exceptions, etc.

This work focuses on the identification and representation of anaphoric expressions within a spoken dialogue system (Botley & McEnery 2000). For a better understanding of the topic, section 2 poses the problem of anaphora and its classification, and it provides two different contexts of occurrence of this phenomenon within this specific domain. Section 3 provides some sample interactions that show both the functionality of the system and the necessity of implementing sophisticated mechanisms to deal with anaphora within this specific domain. At the same time, it illustrates the interaction of anaphoric expressions with other features of natural spoken language.

Section 4 describes some strategies for the resolution of anaphora in spoken dialogue systems. In order to do so, some lexical and syntactic information along with information about the rest of the agents involved in the task is provided. Finally, section 5 describes the application of the whole strategy to a real dialogue. Our proposal has been used within the European project[1], which aimed at the

[1] **http://www.ling.gu.se/dhomme**

ISSN 1135-5948

implementation of spoken dialogue management systems in the home machine environment.

2. Anaphoric expressions in D'Homme

2.1. Contexts for the Anaphor

A very frequent problem in conversational systems appears when they have to face reference phenomena (Palomar & Martinez–Barco 2001; Eckert & Strube 1999).

In our dialogue system, the semantic interpreter makes use of a semantic–oriented grammar which formalizes the commands supported by the system. In the D'Homme environment the following commands are supported (Amores & Quesada 2000) :

- CommandOn
- CommandOff
- CommandDim
- CommandBright

as well as different types of requests such as

- RequestQuantity
- RequestExist
- RequestLocation
- RequestDevices
- RequestState

Finally, some commands may have specific parameters; namely, `DeviceSpecifier` and its corresponding arguments, and `NumericSpecifier`, in case a device is dimmable.

From a linguistic point of view, different cases of anaphora could be listed. In the English version of the system, two types of anaphor are currently supported: the so–called *One–anaphor* and *Pronominal anaphor*. As we will see, both kinds of anaphora result in the same semantic representation even if they have been treated as different lexical units.

2.2. Contexts for the Anaphor

One–anaphor and pronominal anaphor may be found in two different contexts:

- Within a single utterance (intra–sentential anaphora): the anaphora will be solved by means of strategies implemented in the grammar. E.g:
  Turn on the lights in the kitchen and the red one in the living–room.
  Turn on the lights in the kitchen and turn them off in the patio
- Across dialogue contributions: the anaphora will be solved in the Dialogue Management Module, as the examples below will illustrate.

The semantic representation of anaphoric expressions involves adding a feature *CONT:Anaphor* within the `DeviceSpecifier` level together with a generic `DeviceType`, as one of the arguments of the `DeviceSpecifier`, whose *CONT* is instantiated to *device*. This generic *device* stands for any device or devices in the house. The apperance of this `DeviceType` as a generic referent for the anaphoric expression constitutes a first step towards the solution of the problem. One of its advantages lies in that it adds more transparency to the semantic representation we obtain after the sentence has been parsed, since it completes the missing elements in the utterance.

Apart from this, our system is able to process utterances containing anaphoric expressions interacting with quantified ones. The anaphor can also appear within coordinated sentences, within requests, and combined with other complex linguistic phenomena as it will be illustrated in the examples below.

3. Sample interactions

In this section, our goal is to show the functionality of the system in relation to the topic of anaphora resolution. The dialogue examples will serve to explain, in the next sections, the mechanisms which allow us to obtain the representation of anaphoric expressions, along with other linguistic phenomena.

Dialogue 1

- U1: Is there any light on in the sitting–room?
- S1: There are two lights on in the sitting–room.
- U2: Turn off the red one and also the TV.

In U1, the user is issuing a request (more specifically a `RequestExist` dia-

logue move), and an existential quantified expression, as one of the arguments of the `DeviceSpecifier` (*any light* on in the sitting–room). Quantified expressions are represented through a *QUANT* feature, which is part of the DeviceSpecifier type. The possible values for *QUANT* are *all*, *any*, and any numeric value.

In U2, the user's utterance exhibits coordination and *One–Anaphora*. In this case, the dialogue manager will have to go back in the dialogue history to look for an appropriate referent for the anaphoric expression.

Dialogue 2

- U1: What is the status of the light in the kitchen?
- S1: The light in the kitchen is on.
- U2: Turn it off, please.
- U3: And how many lights on are there on the first floor?
- S2: There are three lights on on the first floor.
- U4: Turn all of them off.

In this dialogue, we find an example of a pronominal anaphor in U2, while U4 shows the interaction of anaphora and universal quantification.

Dialogue 3

- U1: What devices are off in the house?
- S1: The lights in the bedroom, the lights in the living–room and the TV.
- U2: Turn on the lights in the bedroom and the small one in the living–room.
- U3: And now the TV

The system supports situations in which the identification of the devices is obtained through their state, like the one in U1. In U2, the dialogue manager will have to look for an appropriate referent within the same turn.

4. Resolution strategies

4.1. Inter–sentential Anaphora

4.1.1. Reference Resolution

In this section, we propose mechanisms for the resolution of some cases of anaphora in our dialogue system. These strategies will depend on the kind of context in which the anaphoric expression occurs. As it was mentioned above, the anaphora can be *intra–sentential* and *inter–sentential*. Intra–sentential anaphora will be solved by the NLU Module, while inter–sentential anaphora will be solved by the Dialogue Management Module.

First of all, we will describe the semantic representation obtained by the semantic interpreter, which serves as the input to the dialogue manager. In order to do this, we will have to provide some lexical and syntactic information.

As two lexical realizations of anaphora have been distinguished in this domain (*them, it / one, ones*), it is essential to provide some information about the linguistic motivation for having two different units. Recall that both of them can occur in both contexts. In the lexicon, there are two units with their corresponding plural counterparts, namely `LOneAnaphor` and `LPron`. The lexical information associated with these two units is as follows:

```
//Lexical entry for Pronouns
 IFPron1(LU)
  LClex( LU:base->LU,
         CAT:LPron,
         DMOVE:specifyParameter,
         TYPE:DeviceType,
         CONT:device,
         <Sg>)

//Lexical entry for One/ones
 IFOneAnaphor1(LU)
  LClex( LU:base->LU,
         CAT:LOneAnaphor,
         DMOVE:specifyParameter,
         TYPE:DeviceType,
         CONT:device,
         <Sg> )
```

In the lexicon, both units have some features associated with them in the form of a DTAC structure. As may be seen, they only differ from a categorial point of view (`LOneAnaphor` and `LPron` respectively). The reason for having two lexical units dealing with the same phenomenon is based on purely linguistic reasons: One–anaphor can only replace a NP realized just by a noun, and pronominal anaphor may have a whole NP as its referent. However, they are equivalent from a functional point of view since both of them are of `TYPE` *DeviceType*, that is, both of them are types of device.

The linguistic motivation for having this generic *DeviceType* when we have an anaphoric expression is to provide a generic referent for the anaphora at the NLU Module.

In the grammar, two rules may consume these lexical units:

The red one/s in *Turn on the red one/s*

```
(DeviceSpecifier -> LDescriptor
                    LOneAnaphor)
   { @up.Descriptor = @self-1;
     @up.DeviceType =a @self-2;
     @up.CONT =a Anaphor;
     @if (@self-2.AGR.num ==pl)
     @then { @up.QUANT.CONT =a all; }}
```

Them / it

```
(DeviceSpecifier -> LPron)
  { @up.CONT =a Anaphor;
    @up.DeviceType =a @self-1;
    @if (@self-1.AGR.num == pl)
    @then { @up.QUANT.CONT =a all;}}
```

4.2. Resolution of Intra–sentential anaphora

Some cases of intra–sentential anaphora are solved by the NLU Module. Consider the following grammar rule:

```
(DeviceSpecifier -> DeviceSpecifier
      Conjunction DeviceSpecifier)
{@up = @self-1;
 @up.DeviceSpecifierC = @self-3;
 @if (@self-3.CONT == Anaphor)
 @then
  {@up.DeviceSpecifierC.DeviceType.CONT
    =a @self-1.DeviceType.CONT;}
```

As it can be noticed in the above rule, *Conjunction* is represented as a recursion within the DeviceSpecifier feature. This rule supports sentences like *Turn on the lights in the kitchen and the red ones in the patio*, where the *CONT* of the first *DeviceType* would be copied onto the *CONT* of the generic *DeviceType*. The anaphor, in turn, would be solved at this stage.

So, as a general rule, everytime an anaphora appears, no matter its type, the feature *Anaphor* will be instantiated as the value of the attribute *CONT* in the *DeviceSpecifier*, and a *DeviceType* with *CONT:device* will be one of its arguments.

4.3. Inter–sentential Anaphora

Cases of inter–sentential anaphora require a more sophisticated resolution mechanism. The DTAC structure would contain the feature *CONT:Anaphor*, and the resolution of the anaphor will be carried out by the Dialogue Management Module together with the Knowledge Management Module (Quesada et al. 2001).

Once the semantic interpreter (parser) has analyzed the user's utterance, we must assume that the DTAC structure is semantically complete to serve as the input to the dialogue manager. It is at this point that several agents begin to take part in the process. But before moving on to this, it is essential to explain how the dialogue manager and the rest of the agents work together in order to find a referent through the dialogue history. In order to do this, we will begin by mentioning the `DevRes` or Device Resolution strategy.

4.3.1. Reference Resolution

In the context of a home environment with a large number of devices, one of the most important tasks to be carried out by the modules in charge of contextual interpretation and dialogue management involves the identification of the device or devices which are the object of a command. One of these tasks is the so–called Device Resolution. In this task, several components or agents in the proposed D'homme architecture are involved. Even though the semantic analysis module provides the formal representation of the user's instruction, the identification of the device or devices to which the desired command has to be applied requires both static (structure of the house, etc) and dynamic information (connected devices, their characteristics, state, etc.) which is stored in the Knowledge Manager. On the other hand, taking into account the previous dialogue history, the Dialogue Manager may apply different disambiguation strategies. Device Resolution must also incorporate anaphor resolution, as well as question accommodation, quantification, etc. This, in turn, imposes a higher degree of coordination and integration between the different modules involved. So Device Resolution constitutes the interface between the Dialogue and knowledge Managers (Quesada et al. 2001).

Upon receiving a DTAC structure with *CONT:Anaphor*, the Dialogue Manager must

detect the kind of Command which has to be executed, together with the number of devices over which that Command must be executed (depending this on the value of the feature *QUANT*). So, first it must identify the desired device which must be, for instance, turned on.

Before proceding with the Device Resolution strategy, the Dialogue Manager must resolve the Anaphor. Recall that we have a feature *CONT: Anaphor*, and we have to find out its referent. In order to do this, this agent will go back in the dialogue history to find an appropriate referent that can replace the value *Anaphor* in the DTAC structure. Once the Dialogue Manager has detected the referent, the anaphor will be solved. After this, the Dialogue Manager sends a request of type DevRes to the Knowledge Manager Module, passing the information known about it. In this case, we know for sure that it has information either about *DeviceType* if we have One-anaphor, or about *DeviceType*, *Location* and/or *Descriptor* if we have pronominal anaphor. The Knowledge manager in turn traverses the semantic graph in the semantic network, looking for the values of each of the arguments provided by the DevRes. That is, it searches for the *CONT* of *DeviceType*, *Descriptor* and *Location*. Once the Knowledge Manager has identified the device / devices in question, it incorporates the state of the device in the response to the Dialogue Manager. Next, the Dialogue Manager triggers the execution of the Command through the Action Manager. In turn, the Action Manager will return a message to the dialogue manager to inform this agent of the result of the execution.

In this way, the anaphor is solved, the command is executed, and the system remains waiting either to exit the system, or to process another command from the user.

5. A real dialogue

In this section, the proposed strategy will be illustrated in a real dialogue management system. In particular, this proposal has been used in the *D'Homme* European project that was described at the beginning of this work. We will illustrate a specific situation in which the system will be forced to solve anaphor and quantification phenomena before going on with the execution of the corresponding commands. In order to do this, we will show the following dialogue example:

- U1: What is the status of the lights in the living–room?
- S1: The lights in the living–room are on.
- U2: Turn off all of them except the one in the corner.
- U3: Now turn on the light in the kitchen and the small one in the patio.

In **U1**'s DTAC (figure 1), a feature *DeviceState* with *CONT: ns* can be noticed at the level of the Command. This value *ns* will be present as a default value in all kinds of Requests. Other possible values for *DeviceState* are *on* in *What lights are turned on?*, and *off* in *How many lights are off in the living–room?*. Querying commands are described as a whole (Quesada et al. 2001), since they share the same basic strategy. The only difference lies in the corresponding *TYPE* label in the DTAC representation. The possible consult commands are *RequestState*, *RequestQuantity*, *RequestExist*, *RequestLocation*, and *RequestDevices*.

In **U2**'s DTAC (figure 2), the interaction of quantification with anaphora can be observed. This sentence presents both pronominal and one-anaphor. In both `DeviceSpecifier`, there is, in turn, a generic `DeviceType`, suggesting a possible referent. Here, the Dialogue Manager, working together with other agents, will find the appropriate referent.

In **U3**'s DTAC (figure 3), the anaphora has already been solved by the NLU Module. It is a case of intra–sentential anaphora, where the second `DeviceSpecifier` has copied the relevant information from the first one, and the referent of the anaphoric expression has been successfully found.

6. Conclusion and future work

In this paper we have presented a proposal for the representation of anaphoric expressions in a spoken dialogue system. At the moment we have all the elements we need to resolve reference phenomena, such as mechanisms for reference resolution (Device Resolution Algorithm), we have access to the dialogue history, and we have an appropriate representation of the anaphoric expression. We have also demonstrated how the NLU Module in the system is capable of solving

some cases of anaphoric reference, namely cases of Intra–sentential anaphora. By now, we are able to cover several cases of inter–sentential anaphora. We are currently working on both the application of the strategy to new domains and the extension of the algorithm to cover additional cases of anaphoric phenomena.

Bibliografía

Amores, J. G., Quesada, J. F. 2000. *Dialogue Moves in Natural Command Languages.* Deliverable 1.1. Siridus project.

Botley, S., McEnery, A. (eds.) 2000. *Corpus–based and Computational Approaches to Discourse Anaphora.* John Benjamins.

Eckert, M., Strube, M. 1999. *Resolving discouse deictic anaphora in Dialogues.* In Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics.(EACL 99).

Martínez–Barco, P. 2000. *Resolución Computacional de la Anáfora en Diálogos: estructura del discurso y conocimiento lingüístico.* Tesis Doctoral. Universidad de Alicante.

Palomar, M., Martínez–Barco, P. 2001. *Computational Approach to Anaphora Resolution in Spanish Dialogues.* Journal of Artificial Intelligence Research 15, pp. 263–287.

Quesada, J.F., Amores, J.G., Bos, J., Ericsson, S., Gorrell, G., Knight, S., Lewin, I., Milward, D., Raynes, M. 2001. *Configuring Linguistic Components in a Plug and Play Environment.* Deliverable 3.1. D'homme project.

Traum, D., Bos, J., Cooper, R., Lewin, I., Mathson, C., Poesio, M. 1999. *A Model of Dialogue Moves and Information State Revision.* Deliverable 2.1. Trindi project.

$$
\left[\begin{array}{lcl}
DMOVE & : & specifyCommand \\
TYPE & : & RequestState \\
ARGS & : & [DeviceSpecifier] \\
DeviceState & : & ns \\
 & & \left[\begin{array}{lcl}
DMOVE & : & specifyParameter \\
TYPE & : & DeviceSpecifier \\
ARGS & : & [DeviceType, Location] \\
QUANT & : & (CONT : all) \\
DeviceSpecifier & : & \left[\begin{array}{lcl}
DeviceType & : & \left[\begin{array}{lcl} CONT & : & light \\ DMOVE & : & specifyParameter \\ TYPE & : & DeviceType \\ AGR & : & [\ num\ :\ pl\ ] \end{array}\right] \\
Location & : & \left[\begin{array}{lcl} DMOVE & : & specifyParameter \\ TYPE & : & Location \\ CONT & : & living-room \end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figura 1: **U1:** *What is the status of the lights in the living–room?*

$$
\left[\begin{array}{lcl}
DMOVE & : & specifyCommand \\
TYPE & : & CommandOff \\
ARGS & : & [DeviceSpecifier, NumericSpecifier] \\
 & & \left[\begin{array}{lcl}
DMOVE & : & specifyParameter \\
TYPE & : & DeviceSpecifier \\
ARGS & : & [DeviceType] \\
\mathbf{CONT} & : & \mathbf{Anaphor} \\
\mathbf{QUANT} & : & [\ \mathbf{CONT}\ :\ \mathbf{all}\ ] \\
DeviceSpecifier & : & \left[\begin{array}{lcl}
DeviceType & : & \left[\begin{array}{lcl} DMOVE & : & specifyParameter \\ TYPE & : & DeviceType \\ \mathbf{CONT} & : & \mathbf{device} \\ AGR & : & [\ num\ :\ pl\ ] \end{array}\right] \\
Exception & : & \left[\begin{array}{lcl}
DMOVE & : & specifyParameter \\
TYPE & : & DeviceSpecifier \\
ARGS & : & [DeviceType, Location] \\
\mathbf{CONT} & : & \mathbf{Anaphor} \\
DeviceType & : & \left[\begin{array}{lcl} DMOVE & : & specifyParameter \\ TYPE & : & DeviceType \\ \mathbf{CONT} & : & \mathbf{device} \\ AGR & : & [\ num\ :\ sg\ ] \end{array}\right] \\
Location & : & \left[\begin{array}{l} DMOVE : specifyParameter \\ TYPE : Location \\ CONT : corner \end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figura 2: **U2**: *Turn off all of them except the one in the corner*

$$
\begin{bmatrix}
DMOVE & : & specifyCommand \\
TYPE & : & CommandOn \\
ARGS & : & [DeviceSpecifier] \\
 & & \begin{bmatrix}
DMOVE & : & specifyParameter \\
TYPE & : & DeviceSpecifier \\
ARGS & : & [DeviceType, Location] \\
 & & \\
DeviceSpecifier & : & DeviceType : \begin{bmatrix} DMOVE & : & specifyParameter \\ TYPE & : & DeviceType \\ CONT & : & light \\ AGR & : & [\ num\ :\ sg\ ] \end{bmatrix} \\
 & & Location : \begin{bmatrix} DMOVE & : & specifyParameter \\ TYPE & : & Location \\ CONT & : & kitchen \end{bmatrix}
\end{bmatrix} \\
 & & \begin{bmatrix}
DMOVE & : & specifyParameter \\
TYPE & : & DeviceSpecifier \\
ARGS & : & [Location, DeviceType, Descriptor] \\
 & & \\
DeviceSpecifierC & : & Descriptor : \begin{bmatrix} DMOVE & : & specifyParameter \\ TYPE & : & Descriptor \\ CONT & : & red \end{bmatrix} \\
 & & DeviceType : \begin{bmatrix} DMOVE & : & specifyParameter \\ TYPE & : & DeviceType \\ \mathbf{CONT} & : & \mathbf{light} \\ AGR & : & [\ num\ :\ sg\ ] \end{bmatrix} \\
 & & Location : \begin{bmatrix} DMOVE & : & specifyParameter \\ TYPE & : & Location \\ CONT & : & patio \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Figura 3: **U3**: *Now turn on the light in the kitchen and the red one in the patio*