CL system: Giving instructions by corpus based selection

Luciana Benotti, Alexandre A. J. Denis

CL system: Giving instructions by corpus based selection

Luciana Benotti
PLN Group, FAMAF
National University of Córdoba
Córdoba, Argentina
luciana.benotti@gmail.com

Alexandre Denis
TALARIS team, LORIA/CNRS
Lorraine. Campus scientifique, BP 239
Vandoeuvre-lès-Nancy, France
alexandre.denis@loria.fr

Abstract

The CL system uses an algorithm that, given a task-based corpus situated in a virtual world, which contains human instructor's speech acts and the user's responses as physical actions, generates a virtual instructor that helps a user achieve a given task in the virtual world. In this report, we explain how this algorithm can be used for generating a virtual instructor for a game-like, task-oriented virtual world such as GIVE's.

1 Introduction

There are two main approaches toward automatically producing dialogue utterances. The most used one is the generation approach, in which the output is dynamically assembled using some composition procedure, e.g. grammar rules. The other is the selection approach, in which the task is to pick the appropriate output from a corpus of possible outputs. The selection approach has only been used in conversational systems that are not task-oriented such as negotiating agents (Gandhe and Traum, 2007a), question answering characters (Kenny et al., 2007), and virtual patients (Leuski et al., 2006). In this paper, we describe the algorithm used by the system CL for giving instructions by selecting utterances from automatically annotated human-human corpora. Our algorithm is the first one proposed for doing generation by selection for task-oriented systems, for details see (Benotti and Denis, 2011).

The advantages of corpus based generation are many. To start with, it affords the use of complex and human-like sentences without detailed analysis. Moreover, the system may easily use recorded audio clips rather than speech synthesis and recorded video for animating virtual humans. Finally, no rule writing by a dialogue expert or manual annotations is needed. Nowadays, most conversational systems require extensive human annotation efforts in order to be fit for their task (Rieser and Lemon, 2010). Semantic annotation and rule authoring have long been known as bottlenecks for developing conversational systems for new domains.

The disadvantage of corpus based generation is that the resulting dialogue may not be fully coherent. Shawar and Atwell (2003; 2005) present a method for learning pattern matching rules from corpora in order to obtain the dialogue manager for a chatbot. Gandhe and Traum (2007b) investigate several dialogue models for negotiating virtual agents that are trained on an unannotated human-human corpus. Both approaches report that the dialogues obtained by these methods are still to be improved because the lack of dialogue history management results in incoherences. Since in task-based systems, the dialogue history is restricted by the structure of the task, the absence of dialogue history management is alleviated by tracking the current state of the task.

In the next section we introduce the corpora used by the CL system. Section 2 presents the two phases of our algorithm, namely automatic annotation and generation through selection. In Section 3 we present a fragment of an interaction with a virtual instructor generated using the GIVE-2 Corpus (Gargett et al., 2010) and our algorithm. Finally, Section 5 discusses its advantages and drawbacks with respect to hand-coded systems.

2 The algorithms

Our algorithm consists of two phases: an annotation phase and a selection phase. The *annotation phase* is performed only once and consists of automatically associating the DG instruction to the DF reaction. The *selection phase* is performed every time the virtual instructor generates an instruction and consists of picking out from the annotated corpus the most appropriate instruction at a given point.

2.1 The automatic annotation

The basic idea of the annotation is straightforward: associate each *utterance* with its corresponding *reaction*. We assume that a reaction captures the semantics of its associated instruction. Defining reaction involves two subtle issues, namely *boundary* determination and *discretization*. We discuss these issues in turn and then give a formal definition of reaction.

We define the *boundaries* of a reaction as follows. A reaction R_k to an instruction U_k begins right after the instruction U_k is uttered and ends right before the next instruction U_{k+1} is uttered. In the following example, instruction 1 corresponds to the reaction $\langle 2, 3, 4\rangle$, instruction 5 corresponds to $\langle 6\rangle$, and instruction 7 to $\langle 8\rangle$.

DG(1): hit the red you see in the far room
DF(2): [enters the far room]
DF(3): [pushes the red button]
DF(4): [turns right]
DG(5): hit far side green
DF(6): [moves next to the wrong green]
DG(7): no
DF(8): [moves to the right green and pushes it]

As the example shows, our definition of boundaries is not always semantically correct. For instance, it can be argued that it includes too much because 4 is not strictly part of the semantics of 1. Furthermore, misinterpreted instructions (as 5) and corrections (e.g., 7) result in clearly inappropriate instruction-reaction associations. Since we want to avoid any manual annotation, we decided to use this naive definition of boundaries anyway.

The second issue that we address here is *discretization* of the reaction. It is well known that there is not a unique way to discretize an action into subactions. For example, we could decompose action 2 into 'enter the room' or into 'get close to the door and pass the door'. Our algorithm is not dependent on a particular discretization. However, the same discretization mechanism used for annotation has to be used during selection, for the dialogue manager to work properly. For selection (i.e., in order to decide what to say next) any virtual instructor needs to have a *planner* and a *planning problem*: i.e., a specification of how the virtual world works (i.e., the actions), a way to represent the states of the virtual world (i.e., the state representation) and a way to represent the objective of the task (i.e., the goal). Therefore, we decided to use them in order to discretize the reaction.

For the virtual instructor we present in Section 3 we used the planner LazyFF and the planning problem provided with the GIVE Framework. The planner LazyFF is a reimplementation (in Java) of the classical artificial intelligence planner FF (Hoffmann and Nebel, 2001). The GIVE framework (Gargett et al., 2010) provides a standard PDDL (Hsu et al., 2006) planning problem which formalizes how the GIVE virtual worlds work.

Now we are ready to define *reaction* formally. Let S_k be the state of the virtual world when uttering instruction U_k, S_{k+1} be the state of the world when uttering the next utterance U_{k+1} and $Acts$ be the representation of the virtual world actions. The *reaction* to U_k is defined as the sequence of actions returned by the planner with S_k as the initial state, S_{k+1} as the goal state and $Acts$ as the actions.

Given this reaction definition, the annotation of the corpus then consists of automatically associating each utterance to its (discretized) reaction. The simple algorithm that implements this annotation is shown in Figure 1.

1: $Acts \leftarrow$ *world possible actions*
2: **for all** utterance U_k in the corpus **do**
3: $S_k \leftarrow$ *world state at* U_k
4: $S_{k+1} \leftarrow$ *world state at* U_{k+1}
5: $U_k.Reaction \leftarrow$ plan(S_k, S_{k+1}, $Acts$)
6: **end for**

Figure 1: Annotation algorithm

2.2 Selecting what to say next

In this section we describe how the selection phase is performed every time the virtual instructor generates an instruction.

The instruction selection algorithm, displayed in Figure 2, consists in finding in the corpus the set of candidate utterances C for the current task plan P (P is the sequence of actions that needs to be executed in the current state of the virtual world in order to complete the task). We define $C = \{U \in \text{Corpus} \mid P \text{ starts with } U.\textit{Reaction}\}$. In other words, an utterance U belongs to C if the first actions of the current plan P exactly match the reaction associated to the utterance U. All the utterances that pass this test are considered paraphrases and hence suitable in the current context.

1: $C \leftarrow \emptyset$
2: $Plan \leftarrow$ *current task plan*
3: **for all** utterance U in the corpus **do**
4: **if** $Plan$ starts with $U.Reaction$ **then**
5: $C \leftarrow C \cup \{U\}$
6: **end if**
7: **end for**
8: **return** C

Figure 2: Selection algorithm

Whenever the plan P changes, as a result of the actions of the DF, we call the selection algorithm in order to regenerate the set of candidate utterances C.

While the plan P doesn't change, because the DF is staying still, the virtual instructor offers alternative paraphrases of the intended instruction. Each paraphrase is selected by picking an utterance from C and verbalizing it, at fixed time intervals (every 3 seconds). The order in which utterances are selected depends on the length of the utterance reaction (in terms of number of actions), starting from the longest ones. Hence, in general, instructions such as "go back again to the room with the lamp" are uttered before instructions such as "go straight", because the reaction of the former utterance is longer than the reaction of the later.

It is important to notice that the discretization used for annotation and selection directly impacts the behavior of the virtual instructor. It is crucial then to find an appropriate granularity of the discretization. If the granularity is too coarse, many instructions in the corpus will have an empty reaction. For instance, in the absence of the representation of the user orientation in the planning domain, instructions like "turn left" and "turn right" will have empty reactions making them indistinguishable during selection. However, if the granularity is too fine the user may get into situations that do not occur in the corpus, causing the selection algorithm to return an empty set of candidate utterances. It is the responsibility of the virtual instructor developer to find a granularity sufficient to capture the diversity of the instructions he wants to distinguish during selection.

3 A sample interaction

In this section we illustrate the interaction between the CL system and a DF. On Figures 4 to 7 we show an excerpt of an interaction between the system and a DF. The figures show a 2D map from top view and the 3D in-game view. In Figure 4, the user, represented by a blue character, has just entered the upper left room. He has to push the button close to the chair. The first candidate utterance selected is "red closest to the chair in front of you". Notice that the referring expression uniquely identifies the target object using the spatial proximity of the target to the chair. This referring expression is generated without any reasoning on the target distractors, just by considering the current state of the task plan and the user position.

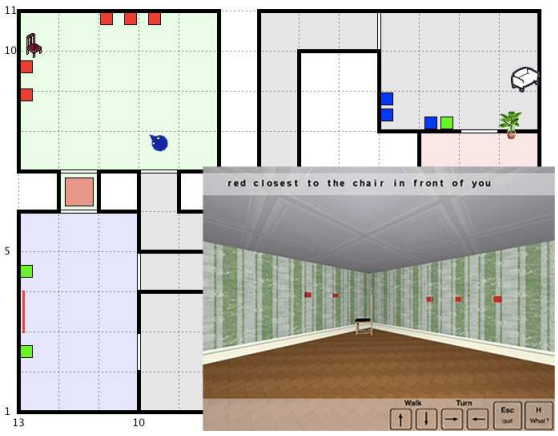

Figure 4: "red closest to the chair in front of you"

After receiving the instruction the user gets closer to the button as shown in Figure 5. As a result of the

L	go
yes	left
straight	now go back
go back out	now go back out
closest the door	down the passage
go back to the hallway	nowin to the shade room
go back out of the room	out the way you came in
exit the way you entered	ok now go out the same door
back to the room with the lamp	go back to the door you came in
Go through the opening on the left	okay now go back to the original room
okay now go back to where you came from	ok go back again to the room with the lamp
now i ned u to go back to the original room	Go through the opening on the left with the yellow wall paper

Figure 3: All candidate selected utterances when exiting the room in Figure 7

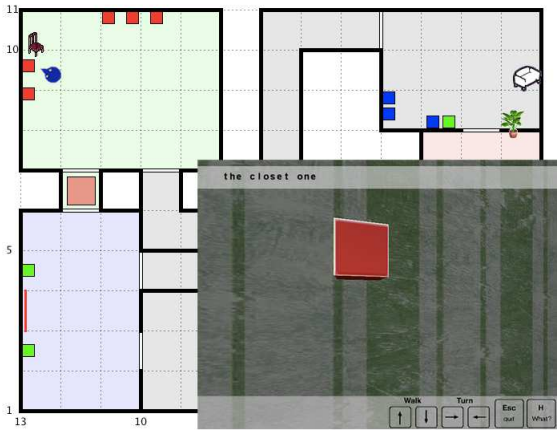

Figure 5: “the closet one”

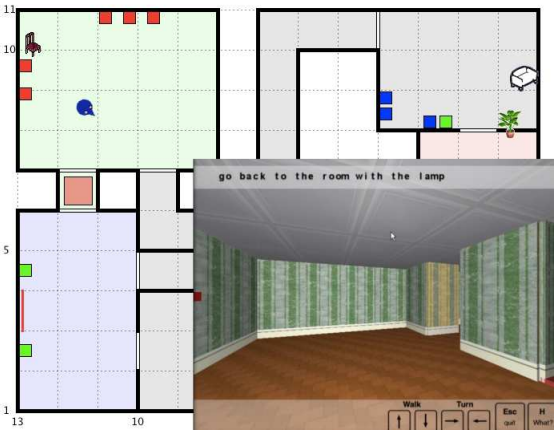

Figure 7: “go back to the room with the lamp”

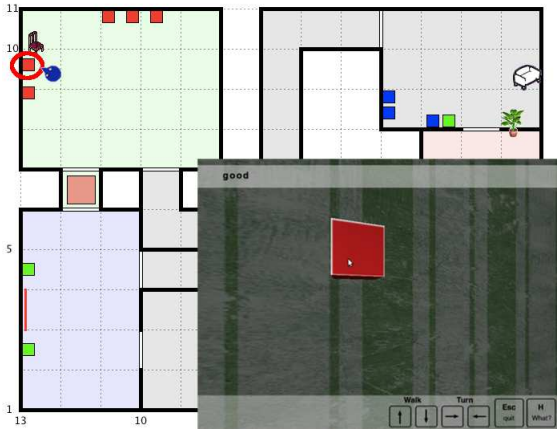

Figure 6: “good”

new user position, a new task plan exists, the set of candidate utterances is recalculated and the system selects a new utterance, namely “the closet one”.

The generation of the ellipsis of the button or the chair is a direct consequence of the utterances normally said in the corpus at this stage of the task plan (that is, when the user is about to manipulate this object). From the point of view of referring expression algorithms, the referring expression may not be optimal because it is over-specified (a pronoun would be preferred as in “click it”), Furthermore, the instruction contains a spelling error (‘closet’ instead of ‘closest’). In spite of this non optimality, the instruction led our user to execute the intended reaction, namely pushing the button.

Right after the user clicks on the button (Figure 6), the system selects an utterance corresponding to the new task plan. The player position stayed the same so the only change in the plan is that the button no

longer needs to be pushed. In this task state, DGs usually give acknowledgements and this is then what our selection algorithm selects: "good".

After receiving the acknowledgement, the user turns around and walks forward, and the next action in the plan is to leave the room (Figure 7). The system selects the utterance "go back to the room with the lamp" which refers to the previous interaction. Again, the system keeps no representation of the past actions of the user, but such utterances are the ones that are found at this stage of the task plan.

We show in Figure 3 all candidate utterances selected when exiting the room in Figure 7. That is, for our system purposes, all the utterances in the figure are paraphrases of the one that is actually uttered in Figure 7. As we explained in Section 2.2, the utterance with the longest reaction is selected first ("go back to the room with the lamp"), the second utterance with the longest reaction is selected second ("ok go back again to the room with the lamp"), and so on.

4 Portability to other virtual environments

The other systems that participated in the challenge do not need a corpus in a particular GIVE virtual world in order to generate instructions for any GIVE virtual world, while our system cannot do without such corpus. As a result these systems are more complex (e.g. they include domain independent algorithms for generation of referring expressions) and take a longer time to develop.

Our algorithm is independent of any particular virtual world. It can be ported to any other instruction giving task (where the DF has to perform a physical task) with the same effort than required to port it to a new GIVE world. This is not true for the other systems that participated in the GIVE-2.5 Challenge. The inputs of our algorithm are an off-the-shelf planner, a formal planning problem representation of the task and a human-human corpus collected on the very same task the system aims to instruct. It is important to notice that any virtual instructor, in order to give instructions that are both causally appropriate at the point of the task and relevant for the goal cannot do without such planning problem representation. Furthermore, it is quite a normal practice nowadays to collect a human-human corpus on the target task domain. It is reasonable, then, to assume that all the inputs of our algorithm are already available when developing the virtual instructor.

Another advantage of our approach is that virtual instructor can be generated by developers without any knowledge of generation of natural language techniques. Furthermore, the actual implementation of our algorithms is extremely simple as shown in Figures 1 and 2. This makes our approach promising for application areas such as games and simulation training.

5 Conclusions

In this paper we presented the system CL, which uses a novel algorithm for doing generation by corpus based selection from human-human corpora without manual annotation.

The algorithms we presented solely rely on the plan to define what constitutes the context of uttering. It may be interesting though to make use of other kinds of features. For instance, in order to integrate spatial orientation and differentiate "turn left" and "turn right", the orientation can be either added to the planning domain or treated as a context feature. While it may be possible to add orientation in the planning domain of GIVE, it is not straightforward to include the diversity of possible features in the same formalization, like modeling the global discourse history or corrections.

In sum, this paper presents the first existing algorithm for fully-automatically prototyping task-oriented virtual agents from corpora. The generated agents are able to effectively and naturally help a user complete a task in a virtual world by giving her/him instructions.

References

Luciana Benotti and Alexandre Denis. 2011. Giving instructions in virtual environments by corpus based selection. In *Proceedings of the SIGDIAL 2011 Conference*, pages 68–77, Portland, Oregon, June. Association for Computational Linguistics.

Sudeep Gandhe and David Traum. 2007a. Creating spoken dialogue characters from corpora without annotations. In *Proceedings of 8th Conference in the Annual Series of Interspeech Events*, pages 2201–2204, Belgium.

Sudeep Gandhe and David Traum. 2007b. First steps toward dialogue modelling from an un-annotated human-human corpus. In *IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Hyderabad, India.

Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Kristina Striegnitz. 2010. The GIVE-2 corpus of giving instructions in virtual environments. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*, Malta.

Jörg Hoffmann and Bernhard Nebel. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302.

Chih-Wei Hsu, Benjamin W. Wah, Ruoyun Huang, and Yixin Chen. 2006. New features in SGPlan for handling soft constraints and goal preferences in PDDL3.0. In *Proceedings of ICAPS*.

Patrick Kenny, Thomas D. Parsons, Jonathan Gratch, Anton Leuski, and Albert A. Rizzo. 2007. Virtual patients for clinical therapist skills training. In *Proceedings of the 7th international conference on Intelligent Virtual Agents*, IVA '07, pages 197–210, Berlin, Heidelberg. Springer-Verlag.

Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. 2006. Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, SigDIAL '06, pages 18–27, Stroudsburg, PA, USA. Association for Computational Linguistics.

Verena Rieser and Oliver Lemon. 2010. Learning human multimodal dialogue strategies. *Natural Language Engineering*, 16:3–23.

Bayan Abu Shawar and Eric Atwell. 2003. Using dialogue corpora to retrain a chatbot system. In *Proceedings of the Corpus Linguistics Conference*, pages 681–690, United Kingdom.

Bayan Abu Shawar and Eric Atwell. 2005. Using corpora in machine-learning chatbot systems. volume 10, pages 489–516.