Giving instructions in virtual environments by corpus based selection

Luciana Benotti, Alexandre A. J. Denis

Giving instructions in virtual environments 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

Instruction giving can be used in several applications, ranging from trainers in simulated worlds to non player characters for virtual games. In this paper we present a novel algorithm for rapidly prototyping virtual instruction-giving agents from human-human corpora without manual annotation. Automatically prototyping full-fledged dialogue systems from corpora is far from being a reality nowadays. Our approach is restricted in that only the virtual instructor can perform speech acts while the user responses are limited to physical actions in the virtual worlds.

We have defined 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 robustly helps a user achieve a given task in the virtual world. We explain how this algorithm can be used for generating a virtual instructor for a game-like, task-oriented virtual world. We evaluate the virtual instructor with human users using task-oriented as well as user satisfaction metrics. We compare our results with both human and rule-based virtual instructors hand-coded for the same task.

1 Introduction

Virtual human characters constitute a promising contribution to many fields, including simulation, training and interactive games (Kenny et al., 2007; Jan et al., 2009). The ability to communicate using natural language is important for believable and effective virtual humans. Such ability has to be good enough to engage the trainee or the gamer in the activity. Nowadays, most conversational systems operate on a dialogue-act level and require extensive 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.

In this paper, we present a novel algorithm for generating virtual instructors from automatically annotated human-human corpora. Our algorithm, when given a task-based corpus situated in a virtual world, generates an instructor that robustly helps a user achieve a given task in the virtual world of the corpus. There are two main approaches toward automatically producing dialogue utterances. One is the selection approach, in which the task is to pick the appropriate output from a corpus of possible outputs. The other is the generation approach, in which the output is dynamically assembled using some composition procedure, e.g. grammar rules. The selection approach to generation 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). To the best of our knowledge, our algorithm is the first one proposed for doing corpus based generation and interaction management for task-oriented systems.

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. The disadvantage of corpus based generation is that the resulting dialogue may not be fully coherent. For non-task oriented systems, dialogue management through corpus based methods has shown coherence related problems. 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 of the current state of the task.

In the next section we introduce the corpora used in this paper. Section 3 presents the two phases of our algorithm, namely automatic annotation and dialogue management through selection. In Section 4 we present a fragment of an interaction with a virtual instructor generated using the corpus and the algorithm introduced in the previous sections. We evaluate the virtual instructor in interactions with human subjects using objective as well as subjective metrics. We present the results of the evaluation in Section 5. We compare our results with both human and rule-based virtual instructors hand-coded for the same task. Finally, Section 7 discusses the weaknesses of the approach for developing instruction giving agents, as well as its advantages and drawbacks with respect to hand-coded systems. In this last section we also discuss improvements on our algorithms designed as a result of our error analysis.

2 The GIVE corpus

The Challenge on Generating Instructions in Virtual Environments (GIVE; Koller et al. (2010)) is a shared task in which Natural Language Generation systems must generate real-time instructions that guide a user in a virtual world. In this paper, we use the GIVE-2 Corpus (Gargett et al., 2010), a freely available corpus of human instruction giving in virtual environments. We use the English part of the corpus which consists of 63 American English written discourses in which one subject guided another in a treasure hunting task in 3 different 3D worlds.

The task setup involved pairs of human partners, each of whom played one of two different roles. The "direction follower" (DF) moved about in the virtual world with the goal of completing a treasure hunting task, but had no knowledge of the map of the world or the specific behavior of objects within that world (such as, which buttons to press to open doors). The other partner acted as the "direction giver" (DG), who was given complete knowledge of the world and had to give instructions to the DF to guide him/her to accomplish the task.

The GIVE-2 corpus is a multi-modal corpus which consists of all the instructions uttered by the DG, and all the object manipulations done by the DF with the corresponding timestamp. Furthermore, the DF's position and orientation is logged every 200 milliseconds, making it possible to extract information about his/her movements.

3 The unsupervised conversational model

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.

3.1 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 reac-

tion $\langle 2, 3, 4 \rangle$, 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. We discuss in Section 5 the impact that inappropriate associations have on the performance of a virtual instructor.

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 4 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. Both the LazzyFF planner and the GIVE planning problem are freely available on the web[1].

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. Moreover, we provide a fragment of the resulting annotated corpus in Appendix A.

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

3.2 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.

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.

[1] `http://www.give-challenge.org/`

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

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 (as is the case for the virtual instructor we evaluate in Section 5), 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.

4 A virtual instructor for a virtual world

We implemented an English virtual instructor for one of the worlds used in the corpus collection we presented in Section 2. The English fragment of the corpus that we used has 21 interactions and a total of 1136 instructions. Games consisted on average of 54.2 instructions from the human DG, and took about 543 seconds on average for the human DF to complete the task.

On Figures 4 to 7 we show an excerpt of an interaction between the system and a user. 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.

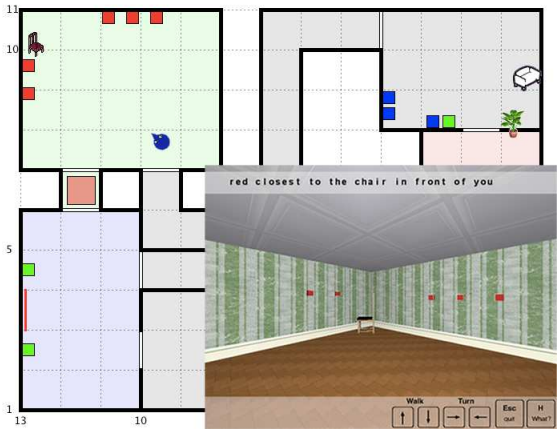

Figure 4: "red closest to the chair in front of you"

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

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

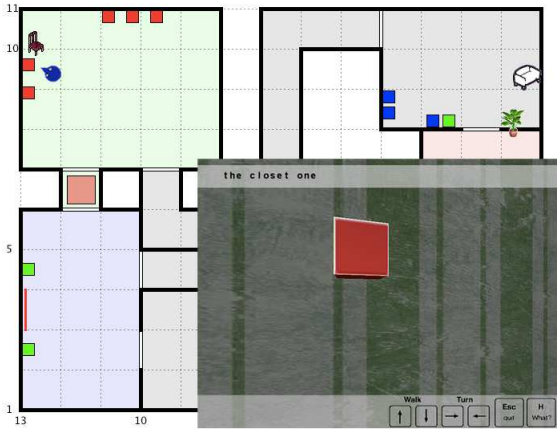

Figure 5: "the closet one"

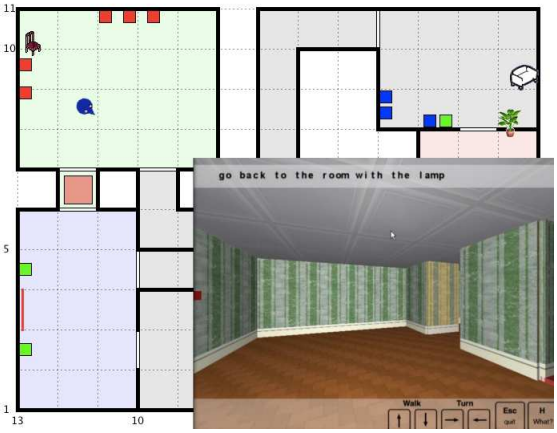

Figure 7: "go back to the room with the lamp"

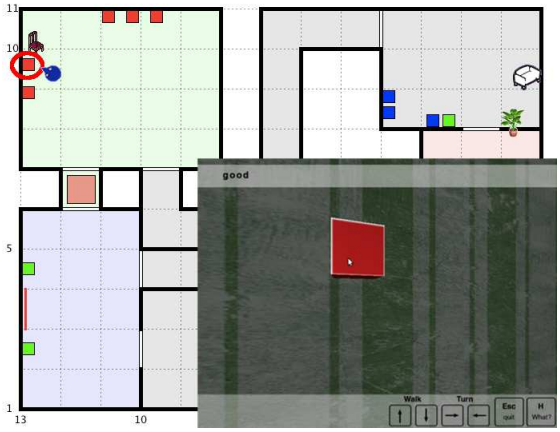

Figure 6: "good"

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 3.2, 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. As you can observe in Figure 3 the utterances in the candidate set can range from telegraphic style like "L" to complex sentences like "Go through the opening on the left with the yellow wall paper". Several kinds of instructions are displayed, acknowledgements such as "yes", pure moving instructions like "left" or "straight", instructions that refer to the local previous history such as "go back out the room" or "ok now go out the same door" and instructions that refer back to the global history such as "okay now go back to the original room".

Due to the lack of orientation consideration in our system, some orientation dependent utterances are inappropriate in this particular context. For instance, "*left*" is incorrect given that the player does not have to turn left but go straight in order to go through the correct door. However, most of the instructions, even if quite different among themselves, could have been successfully used in the context of Figure 7.

5 Evaluation and error analysis

In this section we present the results of the evaluation we carried out on the virtual instructor presented in Section 4 which was generated using the dialogue model algorithm introduced in Section 3.

We collected data from 13 subjects. The participants were mostly graduate students; 7 female and 6 male. They were not English native speakers but rated their English skills as near-native or very good.

The evaluation contains both objective measures which we discuss in Section 5.1 and subjective measures which we discuss in Section 5.2.

5.1 Objective metrics

The objective metrics we extracted from the logs of interaction are summarized in Table 1. The table compares our results with both human instructors and the three rule-based virtual instructors that were top rated in the GIVE-2 Challenge. Their results correspond to those published in (Koller et al., 2010) which were collected not in a laboratory but connecting the systems to users over the Internet. These hand-coded systems are called NA, NM and Saar. We refer to our system as OUR.

	Human	NA	Saar	NM	OUR
Task success	100%	47%	40%	30%	70%
Canceled	0%	24%	n/a	35%	7%
Lost	0%	29%	n/a	35%	23%
Time (sec)	543	344	467	435	692
Mouse actions	12	17	17	18	14
Utterances	53	224	244	244	194

Table 1: Results for the *objective* metrics

In the table we show the percentage of games that users completed successfully with the different instructors. Unsuccessful games can be either canceled or lost. We also measured the average time until task completion, and the average number of utterances users received from each system. To ensure comparability, we only counted successfully completed games.

In terms of task success, our system performs better than all hand-coded systems. We duly notice that, for the GIVE Challenge in particular (and probably for human evaluations in general) the success rates in the laboratory tend to be higher than the success rate online (this is also the case for completion times) (Koller et al., 2009). Koller et al. justify this difference by stating that the laboratory subject is being discouraged from canceling a frustrating task while the online user is not. However, it is also possible that people canceled less because they found the interaction more natural and engaging as suggested by the results of the subjective metrics (see next section).

In any case, our results are preliminary given the amount of subjects that we tested, but they are indeed encouraging. In particular, our system helped users to identify better the objects that they needed to manipulate in the virtual world, as shown by the low number of mouse actions required to complete the task (a high number indicates that the user must have manipulated wrong objects). This correlates with the subjective evaluation of referring expression quality (see next section).

We performed a detailed analysis of the instructions uttered by our system that were unsuccessful, that is, all the instructions that did not cause the intended reaction as annotated in the corpus. From the 2081 instructions uttered in total (adding all the utterances of the 13 interactions), 1304 (63%) of them were successful and 777 (37%) were unsuccessful.

Given the limitations of the annotation discussed in Section 3.1 (wrong annotation of correction utterances and no representation of user orientation) we classified the unsuccessful utterances using lexical cues into 1) correction like "no" or "wrong", 2) orientation instruction such as "left" or "straight", and 3) other. We found that 25% of the unsuccessful utterances are of type 1, 40% are type 2, 34% are type 3 (1% corresponds to the default utterance "go" that our system utters when the set of candidate utterances is empty). In Section 7 we propose an improved virtual instructor designed as a result of this error analysis.

5.2 Subjective metrics

The subjective measures were obtained from responses to the GIVE-2 questionnaire that was presented to users after each game. It asked users to rate different statements about the system using a continuous slider. The slider position was translated to a number between -100 and 100. As done in GIVE-2, for negative statements, we report the reversed scores, so that in Tables 2 and 3 greater numbers indicates that the system is better (for example, Q14 shows that OUR system is less robotic than the rest). In this section we compare our results with the systems NA, Saar and NM as we did in Section 5.1, we cannot compare against human instructors because these subjective metrics were not collected in (Gargett et al., 2010).

The GIVE-2 Challenge questionnaire includes twenty-two subjective metrics. Metrics Q1 to Q13 and Q22 assess the effectiveness and reliability of instructions. For almost all of these metrics we got similar or slightly lower results than those obtained by the three hand-coded systems, except for three metrics which we show in Table 2. We suspect that the low results obtained for Q5 and Q22 relate to the unsuccessful utterances identified and discussed in Section 5.1 (for instance, corrections were sometimes contradictory causing confusion and resulting in subjects ignoring them as they advanced in the interaction). The high unexpected result in Q6, that is indirectly assessing the quality of referring expressions, demonstrates the efficiency of the referring process despite the fact that nothing in the algorithms is dedicated to reference. This good result is probably correlated with the low number of mouse actions mentioned in Section 5.1.

NA	Saar	NM	OUR
Q5: I was confused about which direction to go in			
29	5	9	-12
Q6: I had no difficulty with identifying the objects the system described for me			
18	20	13	40
Q22: I felt I could trust the system's instructions			
37	21	23	0

Table 2: Results for the significantly different *subjective* measures assessing the effectiveness of the instructions (the greater the number, the better the system)

Metrics Q14 to Q20 are intended to assess the naturalness of the instructions, as well as the immersion and engagement of the interaction. As Table 3 shows, in spite of the unsuccessful utterances, our system is rated as more natural and more engaging (in general) than the best systems that competed in the GIVE-2 Challenge.

NA	Saar	NM	OUR
Q14: The system's instructions sounded robotic			
-4	5	-1	28
Q15: The system's instructions were repetitive			
-31	-26	-28	-8
Q16: I really wanted to find that trophy			
-11	-7	-8	7
Q17: I lost track of time while solving the task			
-16	-11	-18	16
Q18: I enjoyed solving the task			
-8	-5	-4	4
Q19: Interacting with the system was really annoying			
8	-2	-2	4
Q20: I would recommend this game to a friend			
-30	-25	-24	-28

Table 3: Results for the *subjective* measures assessing the naturalness and engagement of the instructions (the greater the number, the better the system)

6 Portability to other virtual environments

The hand-coded systems, which we compared to, 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. These hand-coded systems are designed to work on different GIVE virtual worlds without the need of training data, hence their algorithms 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. In fact, 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 hand-coded GIVE systems. 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, which was indeed the case for the GIVE framework.

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.

7 Future work and conclusions

In this paper we presented a novel algorithm for automatically prototyping virtual instructors from human-human corpora without manual annotation. Using our algorithms and the GIVE corpus we have generated a virtual instructor for a game-like virtual environment. A video of our virtual instructor is available in `http://cs.famaf.unc.edu.ar/~luciana/give-OUR`. We obtained encouraging results in the evaluation with human users that we did on the virtual instructor. In our evaluation, our system outperforms rule-based virtual instructors handcoded for the same task both in terms of objective and subjective metrics. We plan to participate in the GIVE Challenge 2011[2] in order to get more evaluation data from online users and to evaluate our algorithms on multiple worlds.

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. Thus we plan to investigate the possibility of considering the context of an utterance as a set of features, including plan, orientation, discourse history and so forth, in order to extend the algorithms presented in terms of context building and feature matching operations.

In the near future we plan to build a new version of the system that improves based on the error analysis that we did. For instance, we plan to take orientation into account during selection. As a result of these extensions however we may need to enlarge the corpus we used so as not to increase the number of situations in which the system does not find anything to say. Finally, if we could identify corrections automatically, as suggested in (Raux and Nakano, 2010), we could get an increase in performance, because we would be able to treat them as corrections and not as instructions as we do now.

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.

[2] `http://www.give-challenge.org/research`

References

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*.

Dusan Jan, Antonio Roque, Anton Leuski, Jacki Morie, and David Traum. 2009. A virtual tour guide for virtual worlds. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, pages 372–378, Berlin, Heidelberg. Springer-Verlag.

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.

Alexander Koller, Kristina Striegnitz, Donna Byron, Justine Cassell, Robert Dale, Sara Dalzel-Job, Johanna Moore, and Jon Oberlander. 2009. Validating the web-based evaluation of nlg systems. In *Proceedings of ACL-IJCNLP 2009 (Short Papers)*, Singapore.

Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. Report on the second NLG challenge on generating instructions in virtual environments (GIVE-2). In *Proceedings of the International Natural Language Generation Conference (INLG)*, Dublin.

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.

Antoine Raux and Mikio Nakano. 2010. The dynamics of action corrections in situated interaction. In *Proceedings of the SIGDIAL 2010 Conference*, pages 165–174, Tokyo, Japan, September. 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.

A Automatically annotated fragment of the GIVE corpus

Utterance: make a left and exit the room
Reaction: ⟨move(b2-room-1-9,room-1-9), move(room-1-9,room-1-8), move(room-1-8,room-1-7), move(room-1-7,room-1-6), move(room-1-6,room-1-3), move(room-1-3,room-1-4), move(room-1-4,room-1-5), move(room-1-5,d3-room-1-5)⟩

Utterance: go forward and turn 90 degrees
Reaction: ⟨move(d3-room-1-5,d3-room-2), move(d3-room-2,room-2)⟩

Utterance: go into the room on the right
Reaction: ⟨move(room-2,d6-room-2), move(d6-room-2,a2-d6-room-3), move(a2-d6-room-3,room-3)⟩

Utterance: push the green button to the left of the red button
Reaction: ⟨move(room-3,b6-room-3), manipulate-stateless(b6), move(b6-room-3,room-3)⟩

Utterance: go into the room on your right
Reaction: ⟨move(room-3,d11-room-3), move(d11-room-3,d11-room-7), move(d11-room-7,room-7)⟩

Utterance: turn 90 degrees right and push the red button
Reaction: ⟨move(room-7,b11-room-7), manipulate(b11), move(b11-room-7,room-7)⟩

Utterance: on your right, push the yellow button
Reaction: ⟨move(room-7,b10-room-7), manipulate-stateless(b10), move(b10-room-7,room-7)⟩

Utterance: turn 180 degrees and push the red button next to the plant
Reaction: ⟨move(room-7,b12-room-7), manipulate-stateless(b12), move(b12-room-7,room-7)⟩

Utterance: turn 180 degrees and push the blue button in the middle of the yellow and blue button
Reaction: ⟨move(room-7,b8-b9-room-7), manipulate-stateless(b9), move(b8-b9-room-7,room-7)⟩

Utterance: turn 90 degrees left
Reaction: ⟨⟩

Utterance: go into the room on the right
Reaction: ⟨move(room-7,d10-room-7), move(d10-room-7,d10-room-6), move(d10-room-6,room-6)⟩

Utterance: turn right and proceed down the room
Reaction: ⟨⟩

Utterance: push the red button next to the blue button on your right
Reaction: ⟨move(room-6,b13-b14-room-6), manipulate(b14), move(b13-b14-room-6,room-6)⟩

Utterance: turn left 120 degrees left
Reaction: ⟨⟩

Utterance: and walk through the hall
Reaction: ⟨move(room-6,d9-room-6), move(d9-room-6,d9-room-5), move(d9-room-5,room-5)⟩