Talking NPCs in a Virtual Game World

Tina Klüwer, Peter Adolphs, Feiyu Xu, Hans Uszkoreit, Xiwen Cheng
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Projektbüro Berlin
Alt-Moabit 91c
10559 Berlin
Germany
{tina.kluewer,peter.adolphs,feiyu,uszkoreit,xiwen.cheng}@dfki.de

Abstract

This paper describes the *KomParse* system, a natural-language dialog system in the three-dimensional virtual world *Twinity*. In order to fulfill the various communication demands between non-player characters (NPCs) and users in such an online virtual world, the system realizes a flexible and hybrid approach combining knowledge-intensive domain-specific question answering, task-specific and domain-specific dialog with robust chatbot-like chitchat.

1 Introduction

In recent years multi-user online games in virtual worlds such as *Second Life* or *World of Warcraft* have attracted large user communities. Such virtual online game worlds provide new social and economic platforms for people to work and interact in. Furthermore, virtual worlds open new perspectives for research in the social, behavioral, and economic sciences, as well as in human-centered computer science (Bainbridge, 2007). Depending on the game type, non-player characters (NPCs) are often essential for supporting the game plot, for making the artificial world more vivid and ultimately for making it more immersive. In addition, NPCs are useful to populate new worlds by carrying out jobs the user-led characters come in touch with. The range of functions to be filled by NPCs is currently still strongly restricted by their limited capabilities in autonomous acting and communication. This shortcoming creates a strong need for progress in areas such as AI and NLP, especially their planning and dialog systems.

The *KomParse* system, described in this paper, provides NPCs for a virtual online world named *Twinity*, a product of the Berlin startup company *Metaversum*[1]. The *KomParse* NPCs offer various services through conversation with game users using question-answering and dialog functionality. The utilization of Semantic Web technology with RDF-encoded generic and domain-specific ontologies furthermore enables semantic search and inference.

This paper is organized as follows: Section 2 presents the NPC modelling and explains the application scenarios. Section 3 details the knowledge representation and semantic inference in our system. Section 4 explains the system architecture and its key components. Section 5 describes the *KomParse* dialog system. Section 7 gives a conclusion and closes off with our future work.

2 Application Scenario and NPC Modelling

The online game *Twinity* extends the *Second Life* idea by mirroring an urban part of the real world. At the time of this writing, the simulated section of reality already contains 3D models of the cities of Berlin, Singapore and London and it keeps growing. Users can log into the virtual world, where they can meet other users and communicate with them using the integrated chat function or talk to each other via Voice-over-IP. They can style their virtual appearance, can rent or buy their own flats and decorate them as to their preferences and tastes.

Out of many types of NPCs useful for this application such as pedestrians, city guides and personnel in stores, restaurants and bars, we start with two specific characters: a female furniture sales agent and a male bartender. The furniture seller is designed for helping users furnish their virtual apartments. Users can buy pieces of furniture and room decoration from the NPC by describing their demands and wishes in a text chat. During the di-

[1] http://www.metaversum.com/

Proceedings of the ACL 2010 System Demonstrations, pages 36–41,
Uppsala, Sweden, 13 July 2010.

Figure 1: The furniture sales NPC selling a sofa

Figure 2: Our bartender NPC in his bar in *Twinity*

alog with the NPC, the preferred objects are then selected and directly put into a location in the apartment, which can be further refined with the user interfaces that *Twinity* provides.

In the second scenario, the bartender sells virtual drinks. He can talk about cocktails with users, but moreover, he can also entertain his guests by providing trivia-type information about popular celebrities and various relations among them.

We chose these two characters not only because of their value for the *Twinity* application but also for our research goals. They differ in many interesting aspects. First of all, the furniture sales agent is controlled by a complex task model including ontology-driven and data-driven components to guide the conversation. This agent also possesses a much more fine-grained action model, which allows several different actions to cover the potential conversation situations for the selling task. The bartender agent on the other hand is designed not to fulfill one strict task because his clients do not follow a specific goal except ordering drinks. Our bartender has the role of a conversation companion and is able to entertain clients with his broad knowledge. Thus, he is allowed to access to several knowledge bases and is able to handle questions (and later conversations) about a much larger domain called the "gossip domain" which enables conversation about pop stars, movie actors and other celebrities as well as the relations between these people. In order to achieve a high robustness, we integrate a chatbot into the bartender agent to catch chitchat utterances we cannot handle.

3 Knowledge Representation and Semantic Inference

Semantic Web technology is employed for modelling the knowledge of the NPCs. The Resource Description Format (RDF) serves as the base for the actual encoding. An RDF statement is a binary relation instance between two individuals, that is a triple of a predicate and two arguments, called the subject and the object, and written as *subj pred obj* (e.g. `f:Sofa_Alatea f:hasMainColour f:Burgundy`).

All objects and properties the NPC can talk about are modelled in this way. Therefore the knowledge base has to reflect the physical properties of the virtual objects in *Twinity* as faithfully as possible. For instance, specific pieces of furniture are described by their main color, material or style, whereas cocktails are characterized by their ingredients, color, consistence and taste. Furthermore, references to the 3D models of the objects are stored in order to create, find and remove such objects in the virtual world.

The concepts and individuals of the particular domain are structured and organized in domain-specific ontologies. These ontologies are modelled in the Web Ontology Language (OWL). OWL allows us to define concept hierarchies, relations between concepts, domains and ranges of these relations, as well as specific relation instances between instances of a concept. Our ontologies are defined by the freely available ontology editor Protégé 4.0[2]. The advantage of using an ontology for structuring the domain knowledge is

[2] `http://protege.stanford.edu/`, as accessed 27 Oct 2009

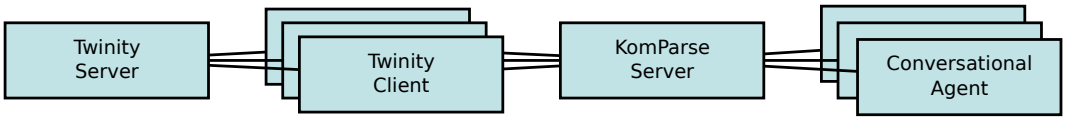

Figure 3: Overall System Architecture – Server/Client Architecture for NPC Control

the modular non-redundant encoding. When combined with a reasoner, only a few statements about an individual have to be asserted explicitely, while the rest can be inferred from the ontology. We employ several ontologies, among which the following are relevant for modelling the specific domains of our NPCs:

- An extensive furniture ontology, created by our project partner ZAS Berlin, defining kinds of furniture, room parts, colors and styles as well as the specific instances of furniture in *Twinity*. This knowledge base contains 95,258 triples, 123 furniture classes, 20 color classes, 243 color instances and various classes defining styles and similar concepts.
- A cocktail ontology, defining 13 cocktail classes with ingredients and tastes in 21,880 triples.
- A biographical ontology, the "gossip ontology", defining biographical and career-specific concepts for people. This ontology is accompanied by a huge database of celebrities, which has been automatically acquired from the Web and covers nearly 600,000 persons and relations between these people like family relationships, marriages and professional relations. (Adolphs et al., 2010)

The furniture ontology is the only knowledge base for the furniture sales agent, whereas the bartender NPC has access to both the cocktail as well as the gossip knowledge base.

We use *SwiftOwlim*[3] for storing and querying the data. *SwiftOwlim* is a "triple store", a kind of database which is specifically built for storing and querying RDF data. It provides a forward-chaining inference engine which evaluates the domain definitions when loading the knowledge repository, and makes implicit knowledge explicit by asserting triples that must also hold true according to the ontology. Once the reasoner is finished, the triple store can be queried directly using the RDF query language SPARQL.

[3] `http://www.ontotext.com/owlim/`

4 Overall System Architecture

Figure 3 shows the overall system architecture. *Twinity* is a server/client application, in which the server hosts the virtual world and coordinates the user interactions. In order to use *Twinity*, users have to download the *Twinity* client. The client allows the user to control the physical representation of the user's character in the virtual world, also called the "avatar". Thus the client is responsible for displaying the graphics, calculating the effects of physical interactions, handling the user's input and synchronizing the 3D data and user actions with the *Twinity* server.

Each NPC comprises two major parts: whereas its avatar is the physical appearance of the NPC in the virtual world, the "conversational agent" provides the actual control logic which controls the avatar autonomously. It is in particular able to hold a conversation with *Twinity* users in that it reacts to a user's presence, interprets user's utterances in dialog context and generates adequate responses.

The *KomParse* server is a multi-client, multi-threaded server written in Java that hosts the conversational agents for the NPCs (section 5). The NPC's avatar, on the other hand, is realized by a modified *Twinity* client. We utilize the Python interface provided by the *Twinity* client to call our own plugin which opens a bidirectional socket connection to the *KomParse* server. The plugin is started together with the *Twinity* client and serves as a mediator between the *Twinity* server and the *KomParse* server from then on (fig. 3). It sends all in-game events relevant to our system to the server and translates the commands sent by the server into *Twinity*-specific actions.

The integration architecture allows us to be maximally independent of the specific game platform. Rather than using the particular programming language and development environment of the platform for realizing the conversational agent or reimplementing a whole client/server protocol for connecting the avatar to the corresponding agent, we use an interface tailored to the specific needs of our system. Thus the *KomParse* system

can be naturally extended to other platforms since only the avatar interfaces have to be adapted.

The integration architecture also has the advantage that the necessary services can be easily distributed in a networked multi-platform environment. The *Twinity* clients require a Microsoft Windows machine with a 3D graphics card supporting DirectX 9.0c or higher, 1 GB RAM and a CPU core per instance. The *KomParse* server requires roughly 1 GB RAM. The triple store is run as a separate server process and is accessed by an XML-RPC interface. Roughly 1.2 GB RAM are required for loading our current knowledge base.

5 Conversational Agent: KomParse Dialog System

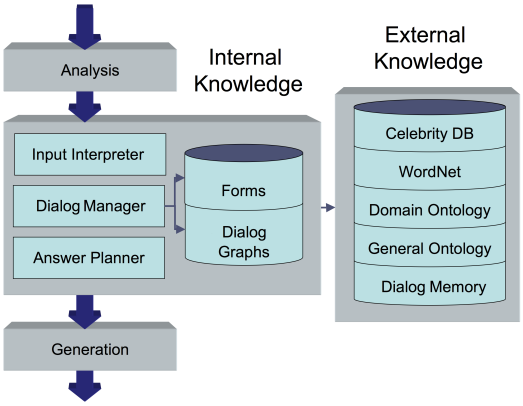

Figure 4: Dialog System: Conversational Agent

The *KomParse* dialog system, the main functionality of the conversational agent, consists of the following three major components: input analyzer, dialog manager and output generator (fig.4).

The *input analyzer* is responsible for the linguistic analysis of the user's textual input including preprocessing such as string cleaning, part-of-speech tagging, named entity recognition, parsing and semantic interpretation. It yields a semantic representation which is the input for the dialog manager.

The *dialog manager* takes the result of the input analyzer and delivers an interpretation based on the dialog context and the available knowledge. It also controls the task conversation chain and handles user requests. The dialog manager determines the next system action based on the interpreted parameters.

The *output generator* realizes the action defined by the dialog manager with its multimodal generation competence. The generated results can be verbal, gestural or a combination of both.

As mentioned above, our dialog system has to deal with two different scenarios. While the focal point of the bartender agent lies in the question answering functionality, the furniture sales agent is driven by a complex dialog task model based on a dialog graph. Thus, the bartender agent relies mainly on question answering technology, in that it needs to understand questions and extract the right answer from our knowledge bases, whereas the sales agent has to accommodate various dialog situations with respect to a sales scenario. It therefore has to understand the dialog acts intended by the user and trigger the corresponding reactions, such as presenting an object, memorizing user preferences, negotiating further sales goals, etc.

The task model for sales conversations is inspired by a corpus resulting from the annotation of a Wizard-of-Oz experiment in the furniture sales agent scenario carried out by our project partner at ZAS (Bertomeu and Benz, 2009). In these experiments, 18 users spent one hour each on furnishing a virtual living room in a *Twinity* apartment by talking to a human wizard controlling the virtual sales agent. The final corpus consists of 18 dialogs containing 3,171 turns with 4,313 utterances and 23,015 alpha-numerical strings (words). The following example shows a typical part of such a conversation:

USR.1: And do we have a little side table for the TV?
NPC.1: I could offer you another small table or a sideboard.
USR.2: Then I'll take a sideboard thats similar to my shelf.
NPC.2: Let me check if we have something like that.

Table 1: Example Conversation from the Wizard-of-Oz Experiment

The flow of the task-based conversation is controlled by a data-driven finite-state model, which is the backbone of the dialog manager. During a sales conversation, objects and features of objects mentioned by the NPC and the user are extracted from the knowledge bases and added into the underspecified graph nodes and egdes at runtime. This strategy keeps the finite-state graph as small as possible. Discussed objects and their features are stored in a frame-based sub-component named "form". The form contains entries which correspond to ontological concepts in the furni-

ture ontology. During conversation, these entries will be specified with the values of the properties of the discussed objects. This frame-based approach increases the flexibility of the dialog manager (McTear, 2002) and is particularly useful for a task-driven dialog system. As long as the negotiated object is not yet fully specified, the form represents the underspecified object description according to the ontology concept. Every time the user states a new preference or request, the form is enriched with additional features until the set of objects is small enough to be presented to the user for final selection. Thus the actual flow of dialog according to the task model does not have to be expressed by the graph but can be derived on demand from the knowledge and handled by the form which in turn activates the appropriate dialog subgraphs. This combination of graph-based dialog models and form-based task modelling effectively accounts for the interaction of sequential dialog strategies and the non-sequential nature of complex dialog goals.

Given a resolved semantic representation, the dialog manager triggers either a semantic search in the knowledge bases to deliver factual answers as needed in a gossip conversation or a further dialog response for example providing choices for the user in a sales domain. The semantic search is needed in both domains. In case that the semantic representation can neither be resolved in the task domain nor in the gossip domain, it gets passed to the embedded A.L.I.C.E. chatbot that uses its own understanding and generation components (Wallace and Bush, 2001).

5.1 Semantic Representation

The input understanding of the system is implemented as one single understanding pipeline.The understanding pipeline delivers a semantic representation which is the basis for the decision of the dialog manager which action to perform next.

This semantic representation can be extracted from the user input by our understanding component via a robust hybrid approach: either via a number of surface patterns containing regular expressions or via patterns reflecting the syntactic analysis of a dependency parser (de Marneffe and Manning, 2008).

The representation's structure is inspired by our knowledge representation design described in section 3 as well as by predicate logic. The core of the representation is a predicate-argument structure limited to two arguments including message type and the whole syntactic information found by the analysis pipeline. The field "Message Type" can have one of the following values: wh-question, yes/no-question, declarative. Predicates can often be instantiated with the lemmatized matrix verb of the successfully analysed piece of the input. If the input contains a wh-question, the questioned fact is marked as an unfilled argument slot. The general structure can be simplified described as:

```
<PREDICATE, ARG1, ARG2, [message-type]>
```

The following examples show the structure used for different input:

- *"Who is the boyfriend of Madonna?"*

  ```
  <hasBoyfriend, Madonna, ?, [wh]>
  ```

- *"I want to buy a sofa."*

  ```
  <buy, I, "a sofa", [declarative]>
  ```

5.2 Information Extraction

Both scenarios make use of state-of-the-art information extraction approaches to extract the important pieces from the user input. While the bartender depends on relation extraction to detect the fact or relation questioned by the user (Xu et al., 2007), the sales agent uses information extraction methods to recognize user wishes and demands. As a result, the questioned fact or the demanded object feature equals the ontology structure containing the knowledge needed to handle the user input. The input *"Do you have any red couches?"* for example needs to get processed by the system in such a way that the information regarding the sofa with red color is extracted.

This is done by the system in a data-driven way. The input analysis first tries to find a demanded object in the input via asking the ontology: Every object which can be discussed in the scenario is encoded in the sales agents knowledge base. This can be seen as a Named Entity Recognition step. In case of success, the system tries to detect one of the possible relations of the object found in the input. This is achieved by querying the ontology about what kind of relations the identified object can satisfy. Possible relations are encoded in the class description of the given object. As a result the system can detect a relation "hasColour" for the found object "sofa" and the color value "red". The found information gets inserted into the form which gets more and more similar or if possible equal to a search query via RDF.

Input	Extracted Information	Knowledge Base
Do you have any red couches?	#hasMainColour (#Sofa, #Reds)	{f:Sofa_Alatea f:hasMainColour f:Reds}, {f:Sofa_Franky f:hasMainColour f:Reds}, {...}
I would like a leather one	#hasMaterial(#Sofa, #Leather)	{f:Sofa_Alatea f:hasMaterial f:leather}, {f:Sofa_Franky f:hasMaterial f:leather}, {...}
Who is the boyfriend of Madonna?	#hasBoyfriend(#Madonna, ?)	{f:Madonna f:hasBoyfriend f:GuyRichie}, {f:Madonna f:hasBoyfriend f:VanillaIce}, {...}

Figure 5: Comparison of Input, Extracted Information and Knowledge Base

6 Conclusion and Future Work

The *KomParse* system demonstrates an attractive application area for dialog systems that bears great future potential. Natural language dialog with NPCs is an important factor in making virtual worlds more interesting, interactive and immersive. Virtual worlds with conversing characters will also find many additional applications in education, marketing, and entertainment.

KomParse is an ambitious and nevertheless pragmatic attempt to bring NLP into the world of virtual games. We develop a new strategy to integrate task models and domain ontologies into dialog models. This strategy is useful for task-driven NPCs such as furniture sellers. With the chatty bartender, a combination of task-specific dialog and domain-specific question answering enables a smart wide-domain off-task conversation. Since the online game employs bubble-chat as a mode of communication in addition to Voice-over-IP, we are able to test our dialog system in a real-time application without being hindered by imperfect speech recognition.

The system presented here is still work in progress. The next goals will include various evaluation steps. On the one hand we will focus on single components like hybrid parsing of input utterances and dialog interpretation in terms of precision and recall. On the other hand an evaluation of the two different scenarios regarding the usability are planned in experiments with end users. Moreover we will integrate some opinion mining and sentiment analysis functionality which can be helpful to better detect and understand the user's preferences in the furniture sales agents scenario.

Acknowledgements

The project *KomParse* is funded by the ProFIT programme of the Federal State of Berlin, co-funded by the EFRE programme of the European Union. The research presented here is additionally supported through a grant to the project TAKE, funded by the German Ministry for Education and Research (BMBF, FKZ: 01IW08003). Many thanks go to our project partners at the Centre for General Linguistics (ZAS) in Berlin as well as to the supporting company Metaversum.

References

Peter Adolphs, Xiwen Cheng, Tina Klüwer, Hans Uszkoreit, and Feiyu Xu. 2010. Question answering biographic information and social network powered by the semantic web. In *Proceedings of LREC 2010*, Valletta, Malta.

William Sims Bainbridge. 2007. The scientific research potential of virtual worlds. *Science*, 317.

Nuria Bertomeu and Anton Benz. 2009. Annotation of joint projects and information states in human-npc dialogues. In *Proceedings of the First International Conference on Corpus Linguistics (CILC-09)*, Murcia, Spain.

Marie C. de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, Manchester, UK.

Michael F. McTear. 2002. Spoken dialogue technology: enabling the conversational user interface. *ACM Comput. Surv.*, 34(1).

Richard Wallace and Noel Bush. 2001. Artificial intelligence markup language (aiml) version 1.0.1 (2001). Unpublished A.L.I.C.E. AI Foundation Working Draft (rev 006).

Feiyu Xu, Hans Uszkoreit, and Hong Li. 2007. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 584–591, Prague, Czech Republic, June. Association for Computational Linguistics.