NPCEditor: A Tool for Building Question-Answering Characters

Anton Leuski, David Traum

Institute for Creative Technologies
13274 Fiji Way
Marina del Rey, CA 90292, USA
{leuski,traum}@ict.usc.edu

Abstract

NPCEditor is a system for building and deploying virtual characters capable of engaging a user in spoken dialog on a limited domain. The dialogue may take any form as long as the character responses can be specified a priori. For example, NPCEditor has been used for constructing question answering characters where a user asks questions and the character responds, but other scenarios are possible. At the core of the system is a state of the art statistical language classification technology for mapping from user's text input to system responses. NPCEditor combines the classifier with a database that stores the character information and relevant language data, a server that allows the character designer to deploy the completed characters, and a user-friendly editor that helps the designer to accomplish both character design and deployment tasks. In the paper we define the overall system architecture, describe individual NPCEditor components, and guide the reader through the steps of building a virtual character.

1. Introduction

Computer-driven interactive virtual characters or virtual humans are increasingly being recognized as useful tools for training and education. As a part of a virtual training environment, these virtual humans can play roles of the user's teammates, tutors, test subjects, or adversaries. For example, they can help a student to learn a new language (Johnson et al., 2005) or polish her negotiation skills (Traum et al., 2005). As the demand for the virtual training environments increases, so grows the need to construct the virtual humans quickly and efficiently.

One of the main properties of virtual humans is the ability to support a natural language interaction with the user, understand human speech and react appropriately. In this paper we discuss NPCEditor[1] – a tool that allows easy construction, maintenance and run-time deployment of language processing capabilities for virtual characters.

The design of a virtual human can vary greatly depending on the task the character has to perform(Traum, 2008). The main initial focus of NPCEditor has been on *question answering characters* (Leuski et al., 2006b). A question answering virtual character can be defined as a virtual persona who answers questions in character, as if being interviewed by a human user. Of course there are many types of interview (job interview, reporter interview, police interview,...), as well as questioning in casual conversation. Moreover, different people and personality types react differently to being interviewed. Thus, in practice, there is quite a bit of difference between a question-answering character and a question-answering system in which retrieval of information is the main goal, and care must be taken in the authoring process to create consistent and compelling personalities. As an example of question-answering characters, consider the recently deployed characters Ada and Grace from the *InterFaces* exhibit at the Boston Museum of Science.[2] Ada and Grace act as museum guides who can answer questions on a number of topics, including their personal background, general facts about virtual human technology, and the exhibits on computers, robotics, and communication in the Cahners ComputerPlace exhibit hall of the museum.

Visitor: good morning

Ada: Hello!

Visitor: what are your names

Ada: My name's Ada.

Grace: And I'm Grace. We're your Virtual Museum Guides. With your help, we can suggest exhibits that will get you thinking! Or answer questions about things you may have seen here.

Ada: What do you want to learn about?

Visitor: what can you tell me about?

Grace: Let's see. What are you interested in? Do you want to learn more about computers, robots, cell phones, hands-on activities...

Visitor: I'm interested in robots

Grace: Here at ComputerPlace you can command a robot to do a Mars mission, and program another one at Robot Park. You can remote control a robot soldier, or figure out how robots climbs walls. You can even make a robotic dog dance, or use one to navigate a virtual world. And don't forget the one that remembers meeting you. What do you want to do first?

Visitor: where is the mars yard

Ada: The Mars Yard is just to your right, and includes Mission Control.

Visitor: can we colonize mars

Ada: Mars isn't exactly paradise. It has little oxygen and no food for humans to survive. It can get extremely cold and has harmful radiation. But, Grace and I, being computers, we can visit just fine.

[1] *NPC* stands for Non-Player Character

[2] http://www.mos.org/interfaces/

Visitor: where do you go to school

Ada: Even though we can't go to school, we are the product of a lot of education.

Grace: The museum gathered up what they wanted to us to know about computer place and sent it over to the University of Southern California. There the Institute for Creative Technologies added V Human knowledge to the stack, then uploaded it all into our database.

Ada: Not exactly a college education, but scientists from both Boston and Southern California have loaned us theirs, until we figure out how to join all of you in the classroom.

A question answering character is defined by the set of the questions it can answer and the set of responses it can give. At the core of NPCEditor is a statistical text classification algorithm that maps a user's questions onto the character's answers. Using the text classification algorithm allows the language processing system reliably handle user utterances that either are not present in the training data or contain speech recognition errors. We have designed the text classification algorithm specifically for this language understanding task and our experiments show it to be extremely robust to the errors in the input (Leuski et al., 2006a; Leuski and Traum, 2008).

Our design goal was to hide the algorithmic complexity from the character designer. At an abstract view, the NPCEditor system consists of two lists: a list of questions and a list of answers. The character designer specifies those utterances, links questions to appropriate responses, and pushes a single button. The system trains a classifier and registers the classifier on the network allowing incoming and outgoing connections from the rest of the virtual human modules.

NPCEditor has so far been used by over a dozen users to construct over 20 different agents, including multiple versions of some of them, which have different specific domain knowledge. NPCEditor is available as part of the ICT virtual human toolkit[3].

In this paper we describe NPCEditor in more detail from the point of view of a virtual human system designer. First we give a brief overview of the system architecture and components in Section 2. Next we describe each of the major components in detail. In a companion paper (Leuski and Traum, 2010), we describe more details of the classification algorithm, evaluation, and use in multiple virtual human agents.

2. System Design & Architecture

NPCEditor supports design and development of a natural language understanding (NLU) component of a virtual human. The component accepts an input from the user of the virtual human system and returns an appropriate response. Generally the input comes in a form of a text string from an automatic speech recognition engine, but this input can come from an instance messaging or an email server. Also,

[3]See http://vhtoolkit.ict.usc.edu/index.php/ for details.

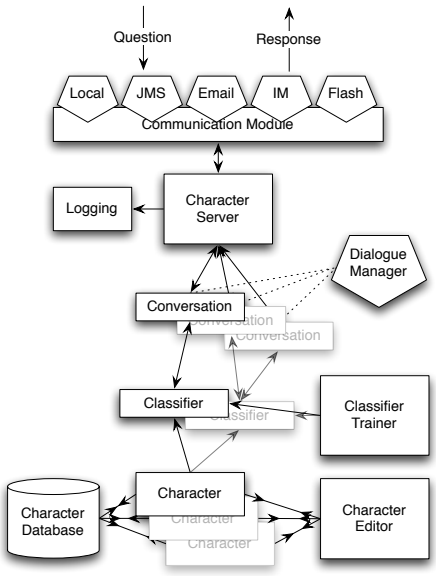

Figure 1: NPCEditor system design.

the text can be combined with additional non-textual information. We will discuss this in more details in Section 5. The natural language component contains a database of possible responses and its task is to select the response given the input. The response format is similar to the the input format – it is a text string optionally combined with other structured information such as a unique identifier or a character name. Note that a single input may result in multiple responses.

There are two tasks NPCEditor helps a character designer to achieve: The first one is to define the character's natural language processing component, and the second is to execute the component and monitor its activity. Thus there are two main parts in NPCEditor: the *character database* that stores the information about the virtual human and the required language data and a *character server* that monitors the network, accepts incoming messages, processes the requests, and sends out character responses.

Figure 1 shows the block diagram of the NPCEditor system. At the center of the system is the character database that stores the information about the virtual characters. A character designer can store multiple characters in the same database so the user may have a conversation with several virtual humans at the same time as in the example in Section 1. Each virtual human character is associated with a set of responses it can produce. The designer enters sample questions and links them to the responses. The *classifier trainer* component generates text *classifiers* that map from the user's questions to the character's responses. The designer also selects one of the provided *dialogue manager* components. A dialogue manager is a rule-based subsystem that uses the classification results and the dialogue history to select the actual response. Finally, the character designer sets up the character server by registering network identities for each character with the *communication module* and enabling the conversation logging. Multiple people

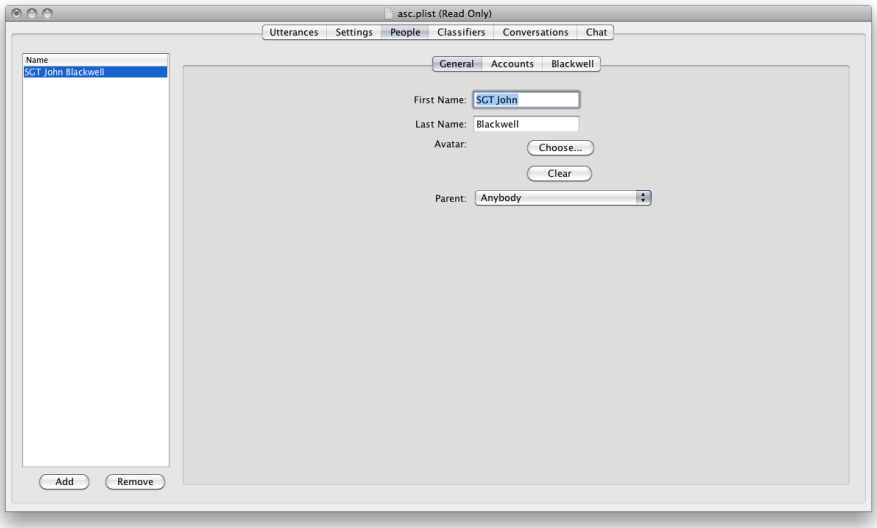

Figure 2: The People panel supports editing of the character general information and network settings.

can interact with the virtual human at the same time. For this purpose the server maintains a list of *conversations*.

NPCEditor provides monitoring and control functionality over the individual system components using a GUI *character editor*. The editor window consists of several tabbed panels each corresponding to a particular function. The character designer starts by defining characters in the People editor panel (Figure 2), where she specifies the character properties and network settings. Then the designer spends most of her time in the Utterances panel (Figure 3). Here the designer enters characters responses and sample questions, annotates the utterances, and links questions to responses. The annotation labels are defined in the Settings editor panel (see Figure 4). The Classifier panel (Figure 5) lists text classifiers for every character and provides controls for training the classifier parameters. The Conversations panel tracks all active conversations and allows the designer or system operator to monitor them. The designer uses the Chat panel (see Figure 6) to pose questions to the characters in the database and observe how the classifiers rank the available responses. It is useful for debugging of the characters.

3. Characters

Figure 2 shows the People tab of the editor window. Here the designer enters all the characters known to the system. Each character has several general properties including the name, graphical avatar, and a set of network addresses that can be used to identify the character in a network environment. We distinguish between *questioner* and *responder* characters. In addition to the general properties the responder characters have a set of answers and some training questions linked to those responses. The questioner characters serve as identifiers of entities (real or virtual) residing outside of the character database. The goal here is to allow the character designer to tune the responses depending on who is asking the question. Suppose a virtual human responder character A has two different answers to question "Who created you?" – one special response for the character's designer B ("You did") and the other for anybody else ("The folks at ICT has put me together."). Then person B and her network address has to be defined as a questioner character in NPCEditor. A "catch-all" questioner character with the name "Anybody" is provided by default.

Each responder character has a parent and optional children. Thus the responder characters form a tree. This relationship defines how individual answers are assigned to the characters: a character's answer set contains all answers explicitly assigned to the character plus all the answers from the character's parent. The default responder character "Anybody" serves as the root of the tree. This technique allows the designer to share responses among characters without re-entering them for each virtual human. This child-parent relationship is also specified in the People panel.

The character data is normally stored in an XML file, but it can be imported from or exported to a variety of other formats including plain text and Excel formats. The data storage subsystem has a plugin architecture with a defined Java interface and a programmer can add additional importing or exporting capabilities to NPCEditor without rebuilding the whole application.

4. Utterance Editor

Figure 3 shows an NPCEditor window with the utterance editor panel selected. There are two main areas here: the question editor is on left and the answer editor is on the right. Both the question and the answer editors follow the master-detail interface pattern: each lists all the utterances in a table and provides controls for editing the selected utterance. Specifically, the designer defines the utterance text, speaker, assigns a text-based identifier and annotation labels. In the screenshot the answer with ID 4 is selected (blue highlighting) and the appropriate data appears in the bottom right part of the screen. The character designer can

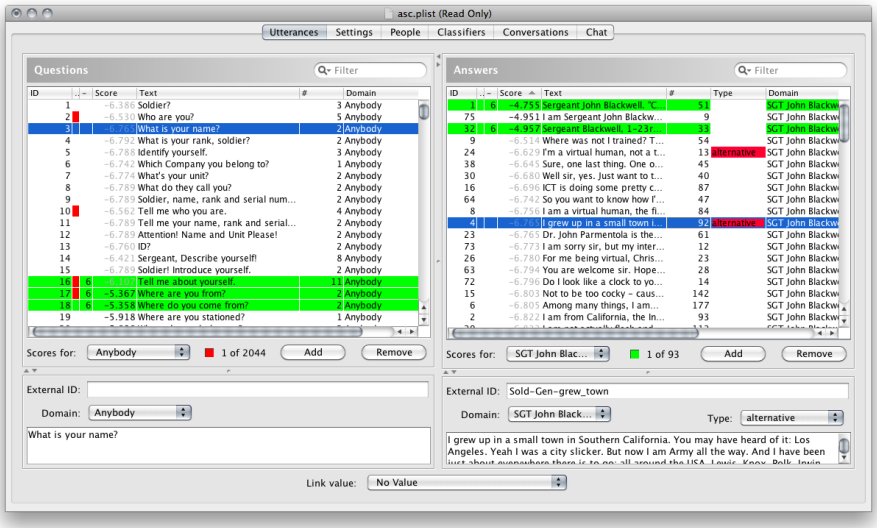

Figure 3: The Utterances panel shows the lists of questions and answers.

sort the table rows, reorder and hide some of the columns, and filter the list by using the GUI control above the table.

To link a question to an answer, the user selects the question and the answer in the corresponding lists and assigns the link value using the popup menu at the bottom of the window. Given a set of appropriate answers to a question, some answers might be more appropriate than others. For example, consider two answers to the question "What is your name?" –"Sergeant John Blackwell" and "Yes, my name is John Blackwell". While both answers are appropriate, using the latter one would manifest in a slight disfluency in the dialog. The system allows the designer to assign a degree of appropriateness to the question-answer link. Currently the system defines a six-point scale for the link strength ranging from "completely irrelevant" to "relevant and fluent" links, following the annotation scheme for dialogue response quality in (Gandhe et al., 2004). If an utterance is selected in one table the opposite table highlights utterances linked to it. For example, in Figure 3 the question 3 is selected (blue highlighting) and the answer table shows two utterances with IDs 1 and 32 as linked to the question. The linked utterances are highlighted in green. The color intensity of the green highlighting is proportional to the strength of the link. Also the link strength value appears in the third column of the table (the value is "6" for both answers). Additionally the number of links connected to the utterance appears in the fifth table column that follows the utterance text column. Question 3 has 2 linked answers and answer 1 has 51 linked questions. The total number of questions and answers appears at the bottom of each table.

NPCEditor enforces some general requirements on the utterances. For example, it warns the user if two utterances have the same identifier or text content. These problems are shown as red bars in the second column of each table. For example, question 2 appears to have the same content as some other questions in the database (those questions are not shown on the Figure) and there is a red bar in row 2. When the user moves the mouse cursor over the red bar, the editor brings up a popup window that details the problem. There is also an overall list problem indicator at the bottom of the table. The square located next to the utterance count is filled with red if the editor detects a problem with any of the utterances. For example, the screenshot shows that there are some problems with the questions in the database, while the answer list is problem-free.

We give such a detailed description to illustrate the point that creating a virtual character requires no special knowledge engineering and expertise. The process of mapping sample questions to answers is simple and straightforward. However, for characters with large answer sets – and the system has been used to build characters with more than a thousand answers and over twenty thousand sample questions – the mapping process can be rather tedious. One of the main impedances in speedy character development is searching for appropriate answers to link to a given sample question. We use the text classifier algorithm to help the character designer with this task. Note the score column (the fourth column) in the answer table shown in Figure 3. These are the similarity scores between each answer and the question selected in the question table ("What is your name?" in this example). The scores are assigned by the classification algorithm. The scores above the classification threshold are shown in black, while the rest of the scores have gray color. You may see the same score ranking in the Chat panel (see Figure 6).

The question does not need to be linked to any answers for the scores to be computed. For example, answer 75 is not linked to question 3 (there is no value in the third column) but appears to have a sufficiently large similarity score. If the character designer sorts the rows in the answer table by score value in decreasing order, the answers with the highest scores – the answers that the system believes to be appropriate to the question – will be shown at the top of the table. Now the designer only needs to validate the system decision by creating the links between the question

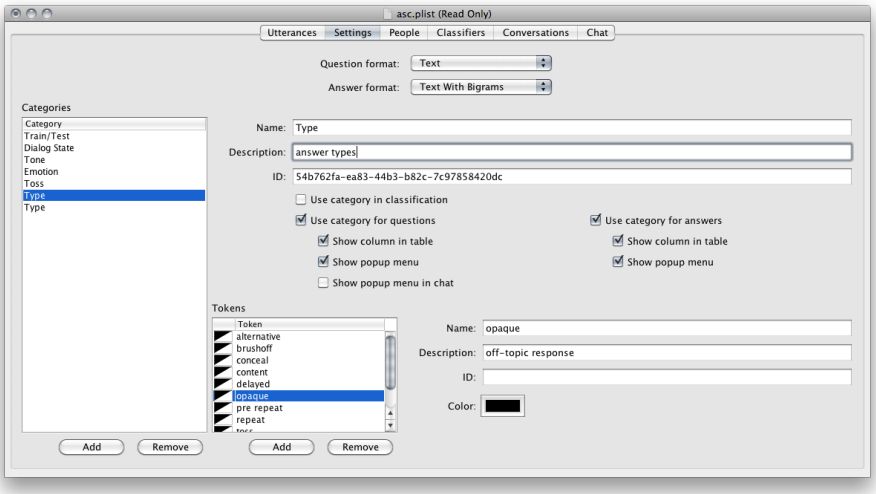

Figure 4: The Settings panel facilitates annotation creation and editing.

and the answers that are really appropriate.[4] The designer might not need to examine and evaluate the bottom portion of the answer list as those utterances are very likely to be inappropriate responses. We have observed this feature to reduce the annotator's load and significantly speed up the question-answer linking process. Note that the roles of the question and answer tables can be reversed: the scores in the question table are the similarity scores between each question and the selected answer. Ordering the question rows by that value will bring up questions that are likely to be linked to the given answer. The system can be told to constantly monitor the database and dynamically update the scores if needed. A note of caution is that for this procedure to be effective it requires some initial albeit small number of links between the questions and answers.

5. Annotations

The sixth column in the answer table of the Utterances panel (see Figure 3, the column headed by "Type") is an example of an annotation column. The editor allows the designer to define arbitrary annotation labels, assign color to the labels, group labels into categories, and assign labels to questions and answers presented as columns corresponding to the categories. A number of annotation categories are provided with the system by default and other categories can be added using controls in the Settings panel (Figure 4). The annotations play three important roles in the system: First, the labels provide the character designer with visual clues, helping her to navigate the database more effectively while linking sample questions to the answers – Figure 3 shows the red label "alternative" assigned to answers 4 and 24. Secondly, the labels can be used by the dialogue manager to identify special classes of answers, as discussed in Section 7. Finally, the labels can be used as non-lexical features by the classification process in addition to the question text. For example, the Gunslinger project (Hartholt et al., 2009) uses computer vision to determine where in the environment the user is looking and whether the user is holding a gun. This information is passed to the NPCEditor and is used to annotate the user's questions. When two characters are present in environment, the same question "What's your name?" might be answered by different characters depending on which character the user is looking at. The designer for Gunslinger specifies a Character category with two labels "harmony" and "utah" (the names of characters in Gunslinger scenario), adds a piece of code to the system that maps vision system messages to those annotation labels, defines two instances of the sample question, one annotated with "harmony" and one annotated with "utah", and links them to responses for the appropriate characters. The system learns from those examples, e.g., to trigger the Utah's response when the user is looking at the Utah character.

6. Text Classification

At the core of the NPCEditor is the ability to map users' questions onto character responses. Because of the variability of natural language, disfluencies in speech, and potential errors introduced by the automatic speech recognition, this mapping process must be very robust to the system input. We use a statistical text classifier that learns the question-to-answer mapping from the question-answer pairs in the character database. We developed a novel classification algorithm especially for the virtual human NLU problem. This approach relies on results from cross-language information retrieval (Lavrenko et al., 2002). It tokenizes the text of the utterances, computes probabilistic representations (language models) for the questions and for the answers, learns how to "translate" a question into an answer, – it computes the likelihood of observing a particular token in the answer given a set of tokens in the question, – compares the question language model to the language model

[4] If the desired answer is already at the top, the designer will not even have to link this answer to get this exact text to match – however it may still be desirable to link the answer so that similar questions will be reinforced and this connection will be more stable as other links are added.

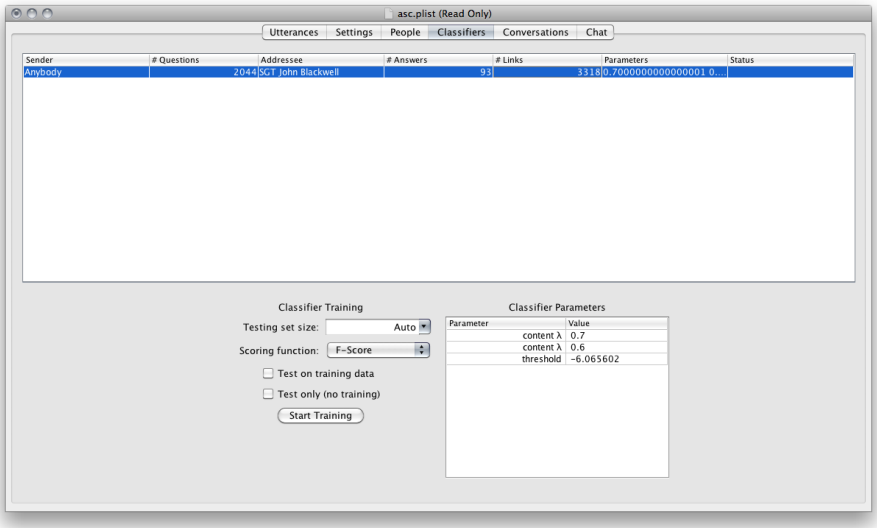

Figure 5: The classifier panel lists existing text classifiers, provides controls for classifier training, and allows an advance user to modify the classifier parameters.

of every known answer, and ranks the answers. We omit the full description of the approach due to space limitation in this paper. More details can be found in (Leuski and Traum, 2008). Our experiments show that the technique outperforms traditional text classification approaches and the classifier output is unaffected by the text quality if the proportion of speech recognition errors in the input stays below 50% (Leuski et al., 2006b).

The text classification algorithm used in NPCEditor has several tunable parameters. The classifier trainer module separates the sample questions into training and testing data sets and optimizes the parameters on the latter using the mapping information from the former. From the interface perspective the system user has a single button, which she needs to press before the virtual character is ready to be deployed. However additional control over the training process and the final classifier parameter values is provided for advanced users (see Figure 5).

7. Dialogue Manager

The text classification algorithm used in the system returns a ranked list of appropriate answers for a given question. This list can be empty when the classifier believes that no known answer is appropriate to the question. Alternatively, this list may contain multiple answers while only one answer has to be returned. Each conversation instance contains a dialogue manager module responsible for selecting the actual response that is sent back to the questioner. Figure 1 shows this relationship as a dotted line. The dialogue manager uses the list of responses returned by the text classifier and the information in the dialogue history to make its decision. For example, it can combine several answer utterances to form a single response.

Two different dialogue managers are provided with the system by default. The basic dialogue manager is a rule-based system with several hardcoded strategies. Its goal is to minimize answer repetition, handle cases when no appropriate answer exists, and process a couple special commands. We named this set of strategies "Blackwell" after the first character designed using NPCEditor. We review its capabilities in the rest of this section. The second dialogue manager allows an advanced designer to create her own response handling strategies. The rules are scripted using the *Groovy* language[5] and interact with the rest of the NPCEditor system via a simplified API. The default distribution includes sample dialogue manager scripts that illustrate the NPCEditor capabilities. Other dialogue managers can be added to the system via external plugins.

If the text classifier finds several possible answers for a user's question, the Blackwell dialogue manager considers the ranking of the responses. It starts at the top of the ranked list and follows it down until it either finds a response that has not been seen in the recent dialogue history (the default window is 4 dialogue turns) or it runs out of responses. Thus, repeating the same or a similar question often results in a different response, introducing variety into the conversation.

When the dialogue manager cannot find an alternative response it will repeat the highest ranked answer that it has given recently. In this case the dialogue manager can precede the answer with a short line indicating that the character is aware of the repetition, e.g., "Let me say this again..." followed by the answer. The character designer can add a number of such lines to the character database. The lines have to be annotated with the "pre-repeat" label from the "Type" category. The category is added to the database automatically by the Blackwell dialogue manager.

If the text classification returns no answers, it means that none of the existing answers have been judged as appropriate answers for the question.[6] We call such questions "off-

[5] http://groovy.codehaus.org/

[6] This can occur for many reasons, including very bad ASR results, or a question that is out of domain.

topics". The character designer should provide answers that deal with the questions, e.g., "Can you say this again?" or "I do not know anything about it", that either ask the user to restate the question or indicate that the question is inappropriate to the topic of the conversation. The off-topic responses are handled similarly to the "pre-repeat" lines – the designer adds the responses to the database and labels them appropriately. When the dialogue manager encounters an off-topic question, it selects one of the off-topic annotated response at random attempting to avoid repeating recently heard responses.

The Blackwell dialogue manager supports three types of off-topic responses. The first type (the "opaque" label in the "Type" category) indicates that the character might have misheard the question and asks the user to repeat it, e.g., "What was that?" The second type (the "unknown" label) normally indicates that the character understood the question, but it does not have a response for it, e.g. "I do not know anything about it." The final type ("conceal") corresponds to cases when the character does not want to continue the discussion on the current topic, e.g., "I do not want to talk about it." If the user asks several off-topic questions in a row, the dialogue manager would progressively select the responses from the first, second, and the third type.

In the event that the user persists in asking questions for which the character has no informative response, the dialogue manager takes initiative and attempts to nudge the user back into the conversation domain by sending the "conceal" off-topic response followed by a suggestion of a relevant question, e.g., "I cannot answer that. But you should ask me about my technology." We call these lines "prompts" and the designer adds those lines to the database in the fashion similar to the off-topic responses. We encourage the character designers to define at least a dozen or so off-topic and prompt lines among the character answers. Our experience shows that a large number of short and sometimes witty off-topic responses makes the interaction much more appealing.

The Blackwell dialogue manager also illustrates how command statements can be handled by the system. Specifically, it includes support for two commands: a request for an alternative answer and a request to repeat the last response. The dialogue manager instantiates two special answer utterances in the character database. The first answer is labeled "repeat" and it should be linked to questions like "Say this again?" and "What was that?" If the text classifier selects that answer as its top choice, the dialogue manager simply returns the last given answer. The second answer is labeled "alternative" and the designer links it to questions like "Do you have anything else to say?" In that case the dialogue manager will behave exactly as if the user has repeated the last question and attempt to find an alternative response to it.

8. Character Server

Once the designer defines the characters, specifies the language data, trains the classifiers, and selects the dialogue manager, the system is ready for deployment. The designer defines and activates one or more network connections that link NPCEditor to other software. The connections (or accounts) are defined in the People panel (see Figure 2) and the interface is similar to setting up an account in an email client. NPCEditor supports a number of network protocols including email, instance messaging and several low level messaging protocols that use either Flash Communication Server or Java Messaging Service APIs. Additional network protocols can be added to NPCEditor via externals plugins for the communication module.

Once the communication module receives a question over one of the channels, it passes the question to the character server. The server constantly maintains a list of active conversations between the questioner and responder characters. Each conversation is a record associated with a particular questioner-responder pair, a text classifier, and history list of all questions asked by the questioner and answers provided by the responder. Given the sender and addressee information the server finds an existing conversation and adds the question to that conversation. If the conversation record does not exist, the server selects the appropriate text classifier and instantiates the conversation.

Note that a single classifier can be associated with multiple conversations – when two questions for character A arrive from different addresses B and C that are not present in the character database, the server creates two temporary identities for B and C and maps them internally to the Anybody questioner character when selecting the classifier. Thus there will be two conversations (B with A and C with A) that use the same classifier (Anybody-vs-A). The character server also has optional logging and question recording capabilities allowing a system operator to monitor the conversations and record them for further analysis.

A transcript of conversations as well as the most recent classification results can be seen in the Chat panel (Figure 6), which also allows you to bypass the network connections and type directly to the character.

9. Conclusions and Future Work

In this paper we described NPCEditor – a tool that allows easy construction, maintenance, and run-time deployment of language processing capabilities for virtual characters. As mentioned above, NPCEditor is available without charge for academic research use, through the Virtual Human Toolkit, and commercial licenses can also be made available. We are continuing to maintain and develop NPCEditor. The system was originally designed to support stateless or semi-stateless interview scenarios where any question can be asked at any time. While this conversation style is feasible for simple role-playing characters or virtual humans in museum kiosks, it is not appropriate for other kinds of interaction in which the characters take the initiative and more complex contextual or state-based reasoning is required. Current work involves adding better support for these kinds of environments. Finally, we should note that NPCEditor has also been used as components in more complex systems (e.g. (Gandhe et al., 2008)), where it performs NLU and/or NLG tasks, mapping from text to a semantic representation (or vice versa), but does not perform dialogue management functions.

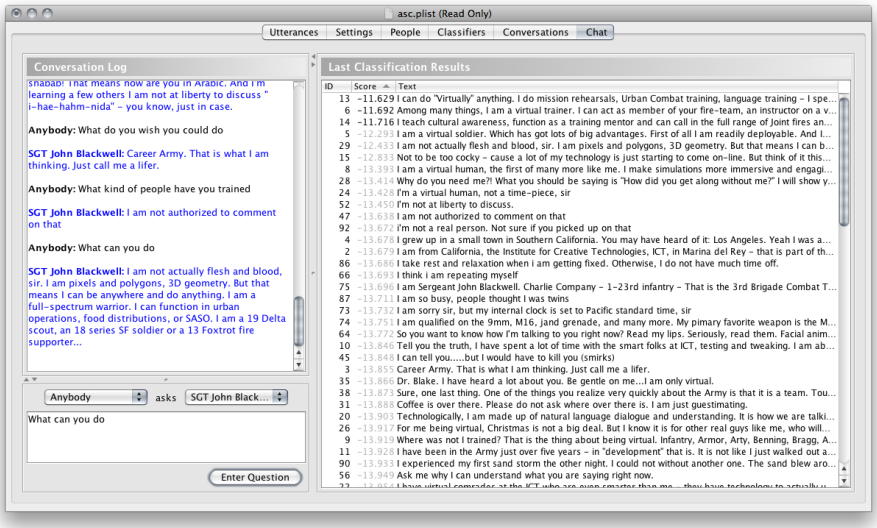

Figure 6: The Chat panel includes controls for entering questions, shows the answer ranking for the last question and the whole conversation transcript.

Acknowledgments

We would like to thank the users of NPCEditor for many helpful suggestions that led to specific improvements in usability and functionality. The effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

10. References

Sudeep Gandhe, Andrew Gordon, Anton Leuski, David Traum, and Douglas W. Oard. 2004. First steps toward linking dialogues: Mediating between free-text questions and pre-recorded video answers. In *Proceedings of the 24th Army Science Conference*, Orlando, Florida, USA, December.

Sudeep Gandhe, David DeVault, Antonio Roque, Bilyana Martinovski, Ron Artstein, Anton Leuski, Jillian Gerten, and David Traum. 2008. From domain specification to virtual humans: An integrated approach to authoring tactical questioning characters. In *Proceedings of Interspeech*, Brisbane, Australia, September.

Arno Hartholt, Jonathan Gratch, Lori Weiss, Anton Leuski, Louis-Philippe Morency, Stacy Marsella, Matt Liewer, Marcus Thiebaux, Prathibha Doraiswamy, and Andreas Tsiartas. 2009. At the virtual frontier: Introducing Gunslinger, a multi-character, mixed-reality, story-driven experience. In *IVA '09: Proceedings of the 9th International Conference on Intelligent Virtual Agents*, pages 500–501, Berlin, Heidelberg. Springer-Verlag.

W. L. Johnson, H. Vilhjalmsson, and M. Marsella. 2005. Serious games for language learning: How much game, how much AI? In *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, Amsterdam, The Netherlands.

Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual relevance models. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 175–182, Tampere, Finland.

Anton Leuski and David Traum. 2008. A statistical approach for text processing in virtual humans. In *Proceedings of the 26th Army Science Conference*, Orlando, Florida, USA, December.

Anton Leuski and David Traum. 2010. Practical language processing for virtual humans. In *Twenty-Second Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-10)*.

Anton Leuski, Brandon Kennedy, Ronakkumar Patel, and David Traum. 2006a. Asking questions to limited domain virtual characters: how good does speech recognition have to be? In *Proceedings of the 25th Army Science Conference*.

Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. 2006b. Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, Sydney, Australia, July.

David Traum, William Swartout, Jonathan Gratch, Stacy Marsella, Patrick Kenney, Eduard Hovy, Shri Narayanan, Ed Fast, Bilyana Martinovski, Rahul Bhagat, Susan Robinson, Andrew Marshall, Dagen Wang, Sudeep Gandhe, and Anton Leuski. 2005. Dealing with doctors: Virtual humans for non-team interaction training. In *Proceedings of ACL/ISCA 6th SIGdial Workshop on Discourse and Dialogue*, Lisbon, Portugal, September.

David Traum. 2008. Talking to virtual humans: Dialogue models and methodologies for embodied conversational agents. In Ipke Wachsmuth and Günther Knoblich, editors, *Modeling Communication with Robots and Virtual Humans*, pages 296–309. Springer.