NPCEditor: Creating Virtual Human Dialogue Using Information Retrieval Techniques

Anton Leuski and **David Traum**
Institute for Creative Technologies
12015 Waterfront Drive
Playa Vista, CA 90094

Abstract

NPCEditor is a system for building a natural language processing component for virtual humans capable of engaging a user in spoken dialog on a limited domain. It uses statistical language classification technology for mapping from a user's text input to system responses. NPCEditor provides a user-friendly editor for creating effective virtual humans quickly. It has been deployed as a part of various virtual human systems in several applications.

1. Virtual Humans

Imagine talking to a computer system that looks and acts almost human—it converses, it understands, it can reason, and exhibit emotion. As an example, recall such computer characters created by Hollywood moviemakers as the librarian in *Time Machine*, the holographic professor in *I Robot*, and of course, the holodeck characters in numerous *Star Trek: The Next Generation* episodes. Once the realm of science fiction, limited, domain-specific versions of these kinds of characters are now achievable using AI and computer graphics technology. Such simulations, called *virtual humans* (VH), open up new horizons for entertainment, teaching, and learning. Virtual humans can serve as colleagues or adversaries in training simulations helping a student to study language and culture (Johnson, Vilhjalmsson, and Marsella 2005) or hone her negotiation skills (Traum et al. 2005). They can help to train physicians in treating psychological disorders (Kenny, Parsons, and Rizzo 2009). They work as virtual guides (Jan et al. 2009), museum docents (Swartout et al. 2010), or even engage the user in a gunfight (Hartholt et al. 2009). Also see the insert box.

A typical VH system is rather complex and may consist of several components including speech recognition, gesture recognition, language understanding, dialogue management, emotion reasoning, planning, inference, verbal and non-verbal output, body simulation, realistic graphics, and mixed reality displays (Gratch et al. 2002). So, before a virtual character is ready to interact, it has to be designed and VH system components have to be assembled and deployed. Therefore, we distinguish four types of VH technology users:

Designers *author* the VH system using available tools. They create scenarios, develop the look and behavior of the agents, including the interactive dialogue behavior.

Administrators *deploy* the system, maintain it in working order so that others can interact and view the VHs.

Interactors *talk to* the VH. There are really two types of interactors, *demoers* who work with the Administrators and are familiar with the system, and *players* who are not. Players are the primary target of the interaction. In the case of Demoers, it is the audience that is the primary target of interaction and demoers are presenting the system to the audience.

Audience members *observe* others interact with the virtual human. As said above, when interactors are demoers then the audience is the primary target of the interaction, however there may be an audience member acting as secondary target even when the interactors are players.

For virtual human systems to become widespread there are two main requirements. First, the advances in technology must reach the level of reliability and efficiency that makes the interactions with virtual humans seamless and realistic. Second, this technology has to be implemented and packaged in software that is accessible to the training or entertainment system designers who are not technical experts in the underlying AI technology.

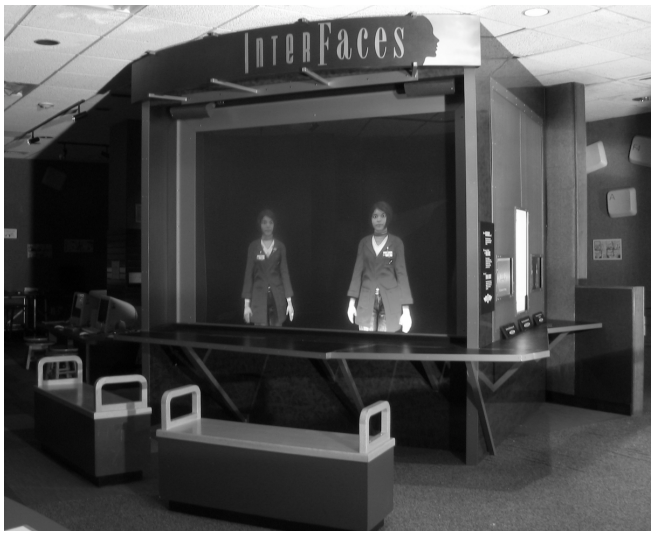
In this paper we focus on the natural language processing (NLP) parts of a virtual human system, including natural language understanding and generation, and dialogue management. We describe NPCEditor[1]—an NLP system that supports all of the above user types. At the core of the system is a statistical text classification algorithm developed specifically for the task of understanding the interactor's language. In the paper we summarize the algorithm and some experimental results that show its effectiveness. NPCEditor packages the text classifier in a GUI-based application that allows creation of useful VH systems with minimal training. In the paper we outline the system design and character creation and deployment process. More details on the user interface for the technology and how it is used can be found elsewhere (Leuski and Traum 2010). NPCEditor has been used extensively in a number of projects both internally in

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

Twins Ada and Grace, the Museum Guides

A well-informed guide or interpreter can have a tremendous influence on the quality of a museum visitors experience. The best guides not only provide information but also engage the visitor in an interactive exchange that can lead to deeper understanding and promote excitement about museum content. Unfortunately, human museum guides are often in short supply. Many studies have shown that people react to virtual humans in much the same way that they react to real people (Reeves and Nass 1996; Gratch et al. 2007). Could virtual humans be used to create museum guides that can engage visitors with museum content?

Ada and Grace are two virtual museum guides that we recently deployed at the Boston Museum of Science as a part of the *InterFaces* exhibit (Swartout et al. 2010). They are life-sized, photo-realistic characters that interact in natural language, complete with gestures, and other forms of non-verbal communication. Ada and Grace 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.

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.

our group and externally by other teams at the institute and outside organizations. We describe some of the VH systems that use NPCEditor and have been deployed at training locations, shows, virtual worlds, and museums.

2. Scientific Contribution: Cross-Language Retrieval for Dialogue Response Selection

There are many NLP technologies that might be applied to virtual human language interaction. The choice depends in large part on the required capabilities: Does the VH have a firm agenda or is it more flexible? Does it lead the interaction or react? Does it need to perform deep inference on the meaning of what is said, or can it stay close to the surface? Will the responses need to be computed on the fly based on current context, or can they be pre-computed or authored?

NPCEditor has been used primarily to construct *question-answering characters*. These characters play the role of interviewees and respond to questions in character. There are many kinds of interviews (doctor-patient, police-suspect, reporter-witness, information seeker-expert, and so forth) and thus question-answering characters have broad applicability. For example, imagine that you are playing the role of a detective in the game of "Clue."[2] An owner of a large house has been murdered and you interrogate the guests of the house. The house guests and witnesses are played by virtual humans. Each character should be capable of answering a number of questions on a limited set of topics that are potentially relevant to the event and it should be able to deflect all other questions.

A question answering virtual human is characterized by a collection of responses relevant to a particular topic. This approach gives complete control over the virtual persona's knowledge and expressions to the scriptwriter who creates the responses. It allows the writer to specify the character of the virtual persona, what information it can deliver and the form of that delivery. When an interactor comes up to the virtual character and asks it a question, the system driving the character analyzes the interactor's question and selects the appropriate response from the collection.

This approach also simplifies the overall VH system: a bare-bones system would consist of an ASR module for speech processing, the NPCEditor system to process the interactor's utterances and select the character response, and a rendering engine, capable of presenting the animated character on the screen and playing back prerecorded responses.

Automatic question answering has been studied extensively in recent years. It is generally defined as an information retrieval (IR) problem where a user places her request in a form of a question and expects a relevant and succinct response, e.g. "How tall is mount Everest?"—"Mount Everest is 29029 feet tall." One example of such a system is START from MIT (Katz 1988). It uses well-defined information databases and carefully crafted question parsing rules to find the required answer. Web-based question answering systems and systems studied in the context of the question-answering track at the Text REtrieval Conference (TREC) attempt to answer user's questions by finding and parsing relevant paragraphs in large text collections (Voorhees 2003).

In contrast to the fact-based question answering scenario where the goal is to provide the most *relevant* answer, we focus on the answer's *appropriateness*. In our example about an investigation, an evasive, misleading, or an "honestly" wrong answer from a witness character would be appropriate but might not be relevant. Alternatively, different characters may have different knowledge about the event and respond differently to the same question. We try to highlight that distinction by talking about question-answering *characters* as opposed to question-answering systems or agents. Another difference is that question-answering systems rely on the question text to be lexically and grammatically correct and well-formed, while our system is primarily used to reply to spontaneous spoken language, which is much more likely to include disfluencies and non-standard constructions. A third difference is that the input for our system comes from an automatic speech recognition (ASR) module that sometimes introduces errors into the transcription. These errors can affect the interpretation performance significantly. A virtual human should be robust to both disfluencies in conversational English and to the errors introduced by the ASR module.

Similar requirements exist for automatic phone reservation and call routing systems (Gorin, Riccardi, and Wright 1997). For example, Chu-Carroll and Carpenter describe a system that answers a phone call, asks a caller some questions, and routes the call to the appropriate destination (Chu-Carroll and Carpenter 1999). The system uses a vector-based text classification approach to analyze the caller's responses and map them to the destinations in the organization. Our NPCEditor system maps text of the question directly to texts of the answers and uses a novel text classification approach based on statistical language modeling that significantly outperforms vector-based approaches (Leuski et al. 2006).

Text classification has been studied for several decades, and numerous approaches exist (Lewis et al. 1996). It is the task of assigning pieces of text to one or more classes based on the training data. The traditional text classification approach for our task is to define each answer as a class and define the corresponding questions as the training text pieces. Generally, a text string is represented as a feature vector where individual words serve as feature elements. When a new question arrives, it is tokenized, converted into a feature vector representation, compared to the vectors of the known questions, and the answer corresponding to the best matching group of questions is returned. The disadvantage of this approach is that it completely ignores the content of an answer. The main difference between our text classification and a traditional text classification approach is that we match a user's question to known answers and not to the known questions. Our experiments show that taking the answer text into account during classification improves the effectiveness of the classification significantly.

Let us explain the theory behind our classification approach. Our task can be described as follows: given a question about a particular topic, find an answer about the

[2] "Clue" is official trademark of Hasbro Inc.

same topic. The key achievement of IR is an ability to match two strings of text based on content similarity. That is how a search system works—a text representation is computed for documents and a query, a matching algorithm is applied, and the best match is returned to the person who entered the query. One technique for text content representation that has recently gained wide usage in IR (Ponte and Croft 1997) uses the notion of a statistical language model: a probability distribution $P(W)$ over all possible word strings $W = w_1, ..., w_n$. Here is how it works: a content topic can be described by different text strings, some are more likely to be used than others. For example, the prhase "green and round" is much more likely to be used when describing an apple than the phrase "blue and square." So if we can define the probability of observing each possible text string in connection with a given topic, we will have a very detailed representation for the topic. It is reasonable to assume that language models of similar topics will also be similar. If we can estimate a language model from the question string and another language model for an answer string, we can compare the content of the question and the answer by comparing the corresponding language models.

Before we describe the details of the method, we have to make an observation: We cannot compare question and answer language models directly because the former is the probability distribution over questions and the latter is the probability over answers. These distributions are likely to be different even when matching to the same topic. For example, due to grammar rules, some strings (e.g. strings containing "wh" words) are much more likely to appear as questions than answers. Moreover questions and answers are "generated" by different entities—the interactor and the character (or, the scriptwriter) who may have different styles of expression reflected in the bias of the language model. These differences allow us to talk about questions and answers as samples from two different languages.

Here is where some training data can be useful. If for each answer in a character database we have some questions that can be answered by that answer, we can train the system to "translate" from the language of questions to the language of answers. When we see a new question Q we use that training data to estimate the language model of the answer to the question $P_Q(A)$ and then compare that language model to language models of the character answers and return the best match. This approach is very similar to the cross-language information retrieval task, e.g., where a search system has to find Chinese documents in response to an English query (Grefenstette 1998). The training data that pairs sample questions with the answers serves as "parallel corpora" and the translation rules are derived implicitly from that mapping.

There are different ways to compare two probability distributions. NPCEditor uses the Kullback-Leibler (KL) divergence $D(P_Q(A)||P(A))$ defined as

$$D(P_Q(A)||P(A)) = \int_A P_Q(A) \log \frac{P_Q(A)}{P(A)} \qquad (1)$$

which can be interpreted as the relative entropy between two distributions. Note that the Kullback-Leibler divergence is a dissimilarity measure, we use $-D(P_Q(A)||P(A))$ to rank the answers.

Normally a topic is represented by a single text string (W). It is impossible to determine the language model from such a sample explicitly. The goal is to estimate the probability of such a string, $P(W)$, as accurately as possible. The problem of estimating the joint probability $P(w_1, ..., w_n)$ of several words occurring together to form a string of text W has received a lot of attention in recent years among researchers in the IR community. The main challenge is to take into account the interdependencies that exist among the individual words while still making the computation feasible. Several different methods were suggested starting from the most trivial technique where all words are assumed to be distributed identically and independently from each other—the unigram model:

$$P(w_1, ..., w_n) = \prod_{i=1}^{n} P(w_i) \qquad (2)$$

Other approaches include Probabilistic Latent Semantic Indexing (PLSI) (Hofmann 1999) and Latent Dirichlet Allocation (LDA) (Blei et al. 2003), where the authors model text collections by a finite set of k topics and the overall text probability is viewed as a mixture of the individual topic language models.

Lavrenko (Lavrenko 2004) suggests a more general approach where the word interdependencies are defined by an unknown vector parameter θ and the words are taken as conditionally independent. This step allows him to relax the independence assumption of the unigram model so that the probability distribution depends on the co-occurrence of words. Lavrenko treats the vector θ as a random variable and suggests several techniques for estimating its probability distribution from experimental data. He calls it a Relevance Model approach and shows how PLSI and LDA can be viewed as special cases of the approach. His experiments and studies conducted by other researches (e.g., (Wei and Croft 2006)) established the relevance model approach as one of the top performing techniques in information retrieval.

While Lavrenko suggests several methods for estimating the probability distribution of vector θ, one of these techniques is most often used in the experiments. Its advantages are the small number of parameters to estimate (i.e., one) and relative computational efficiency. Specifically, given a collection of known questions $\mathcal{Q}$, the language model of a new question $Q = q_1, ..., q_n$ is defined as follows:

$$P(q_1, ..., q_n) = \frac{1}{|\mathcal{Q}|} \sum_{s \in \mathcal{Q}} \prod_{i=1}^{n} p_s(q_i) \qquad (3)$$

where $|\mathcal{Q}|$ is the number of questions in the database and $p_s(q_i)$ is the probability of observing word q_i in string s. There are several ways of estimating the latter value. We use Maximum Likelihood Estimation (MLE) with Jelinek-Mercer smoothing (Bahl, Jelinek, and Mercer 1990):

$$p_s(q) \cong \pi_s(q) = \lambda_\pi \frac{\#_s(q)}{|s|} + (1 - \lambda_\pi) \frac{\sum_s \#_s(q)}{\sum_s |s|} \qquad (4)$$

$\#_s(q)$ is the number of times word q appears in string s, $|s|$ is the number of words in string s, and λ_π is a tunable parameter that can be determined from the training data.

An astute reader may notice that equation 3 is similar to the unigram model: it is an average of unigram models of individual questions in the training data. It allows us to take into account the word co-occurrence in the training data and incorporate this into model estimation.

Equation 3 assumes that all words q_i come from the same vocabulary. We can show that in the case of two different vocabularies, the conditional probability $P(a|Q)$ of observing a word a in answer language given an interactor's utterance Q can be estimated as:

$$P(a|Q) = \frac{P(a, q_1, ..., q_n)}{P(q_1, ..., q_n)} = \frac{\sum_s \pi_{\mathcal{A}_s}(a) \prod_{i=1}^{m} \pi_{\mathcal{Q}_s}(q_i)}{\sum_s \prod_{i=1}^{m} \pi_{\mathcal{Q}_s}(q_i)} \quad (5)$$

The matching criteria in Equation 1 can be written as

$$D(P_Q(A)||P(A)) = \sum_a P(a|Q) \log \frac{P(a|Q)}{\pi_A(a)} \quad (6)$$

In summary, given a character database $\{\mathcal{Q}_s, \mathcal{A}_s\}$ and a question Q, we use Equations 3, 4, and 5 to compute Equation 6 for each answer A in the database and return the answer with the highest value $-D(P_Q(A)||P(A))$. See (Leuski et al. 2006; Leuski and Traum 2008) for more details.

The final parameter is the classification threshold on the KL-divergence value: only answers that score above the threshold value are returned from the classifier. The threshold is determined by tuning the classifier on a randomly chosen subset of the training data.

Non-lexical Features

So far we have described how a textual answer is selected in response to a textual question. There are several other cases in which the NPCEditor uses the same classification algorithm to go beyond this scenario. First, in some applications we may use the cross-language information retrieval approach to convert between text and a semantic language. In some systems, we use the NPCEditor to recognize features such as speech acts or impact on interpersonal variables (Roque and Traum 2007), while in other systems, the NPCEditor can be used to interpret the meaning of an utterance in a semantic representation, rather than selecting the answer to respond (Gandhe et al. 2008). Likewise, the NPCEditor can be used to translate a semantic representation of a response into text (Leuski and Traum 2008).

In some applications additional context information might be available as well as text. For example, in the Gunslinger system (Hartholt et al. 2009) the interactor meets with three different virtual humans. The system uses NPCEditor, an ASR module, and a vision component, which (among other things) detects where the interactor is looking. NPCEditor annotates the ASR output with a token corresponding to the interactor's gaze target. The classifier treats such annotations as words in a piece of text associated with the question but separate from the actual question text. Thus a question becomes a multi-field data structure. One of these fields contains the original text, the other fields contain label tokens. These label tokens have special vocabulary different from the question text vocabulary, so a separate language model is estimated for each field. The question-answer similarity score becomes a weighted sum of similarities between the answer language model and the language models for each field in the question data structure:

$$D(P_Q(A)||P(A)) = \sum_i \alpha_i \sum_{w \in V_i} P(w|Q_i) \log \frac{P(w|Q_i)}{\pi_{A_i}(w)} \quad (7)$$

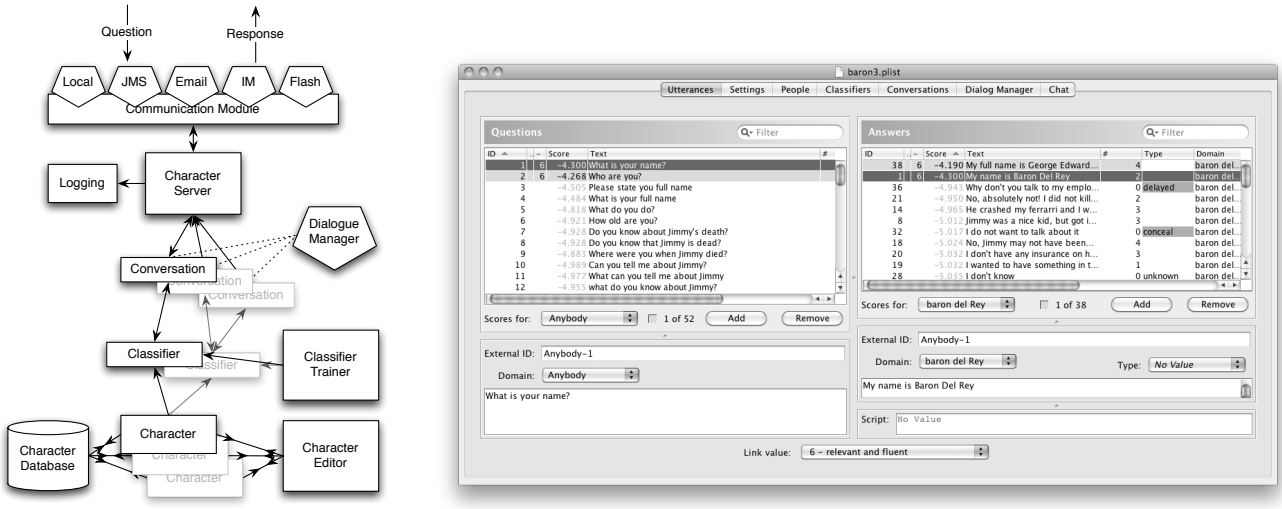
here Q_i and A_i are the ith field of the question and the answer, the outer summation goes over every field of interest, while the inner summation iterates over vocabulary for the ith field. The parameters α_i allow us to vary the importance of different fields and can be determined from the training data. Thus NPCEditor can be trained to respond differently to the same question,—e.g., "What is your name?",—depending on who is the interactor is looking at. NPCEditor's user interface allows the designer to define arbitrary annotation classes or categories and specify some of these categories as annotations to be used in classification.

Dialogue Management

The text classification algorithm 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. The dialogue manager is tasked with choosing the one response that is returned back to the user. NPCEditor contains a rule-based dialogue manager that interacts with the rest of the system via a simplified API:

- The classifier selects multiple answers: the dialogue manager returns the least recently used answer, breaking ties by the classifier score.
- The classifier selects no answer: the classifier believes that there is no appropriate response in the character database for the user's question. We call such a question "off-topic". The dialogue manager returns one of the answers that the designer specifies as an "off-topic" answer, e.g., "Say this again?" or "I do not know anything about it."
- The system returns several off-topic responses in a row: the dialogue manager tries to bring the user back to domain of conversation by prompting her with a question, e.g., "Why don't you ask me about my technology?" We encourage the character designers to add a variety of off-topic and prompt utterances to the character database.

For simple question-answering characters, variations of the above approach are sufficient to generate useful behavior. However, this approach is less suited to dialogues where the decision of what to say is based more on context and/or character/scenario goals than reactions to a new question. NPCEditor can also support more complex dialogue phenomena, as long as the basic paradigm of selecting preauthored outputs can be maintained. This "advanced" func-



(a) (b)

Figure 1: a) NPCEditor system design. b) Character editor screen.

tioning is still rather primitive, however, requiring the designer to program the dialogue manager rather than providing graphical support and easy to follow design guidelines (see Section 6 for planned future directions).

An advanced character designer can create her own response handling strategies using the Groovy scripting language[3]. For example, in the Gunslinger project the interactor's experience follows a predetermined script that goes through several states. There is a greeting and smalltalk state, where the interactor meets two virtual characters and gets to know them. It is followed by another state, where the characters confide their problem to the interactor. It is followed by another state when the third character comes on stage, and so on. The interactor's behavior and how she responds to the characters determines the story development, the ordering of states, and the topic of conversations. For each dialogue state the Gunslinger designers specify a subset of the character lines that are appropriate for that state. They train a text classifier for each state and switch between the classifiers as the interactor's experience transitions between states.

The dialogue manager also keeps a model of the interactor and updates it as the interaction progresses. For example, at the beginning of the experience the characters are polite to the interactor and would pause to listen when the interactor tries to speak. Later on, if the interactor says something that upsets them, the characters may become more hostile and hush the interactor if she tries to interrupt them.

3. Practical Contribution: System for Character Development and Deployment

While the cross-language information retrieval models described in the previous section have been shown to be effective (see Section 4), it can still be daunting for a system developer to master the equations, create the requisite training data, and train a classifier. It may also be a challenge for a system administrator to connect this module to other parts of a VH system, so that interactors can successfully communicate with the virtual human. One of the goals of NPCEditor is to preserve the effectiveness of the technical approach while hiding the complexity from the users. To this end, NPCEditor creates a unified development and run-time interface, that allows easy character authoring and deployment. To create a character, a designer only has to populate the language database and push a single button to train the classifier. To integrate the module into a VH system an administrator has to fill out a form as simple as an account form in an email client.

NPCEditor is written in Java and should run on any platform that supports Java 6. It has been extensively tested on Microsoft Windows and Mac OS X. The initial version was developed in 2004-2005 and subsequently maintained and extended by the first author.

Figure 1a shows the block diagram of the NPCEditor system. The character database stores information about the virtual characters. A character designer can store multiple characters in the same database so the interactor(s) may have a conversation with several virtual humans at the same time. 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 *classifiers* using the methods described in Section 2 that map from the interactor'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 charac-

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

ter server by registering network identities for each character with the *communication module* and enabling conversation logging. The *character server* monitors the network, accepts incoming messages, processes the requests, and sends out character responses. Multiple people can interact with the virtual human at the same time. For this purpose the server maintains a list of *conversations*.

The overall architecture is modular and extensible. Several parts of the NPCEditor functionality can be extended via external plugins. These include supported network protocols, dialogue management, text pre-processing for classification, classification optimization function, and supported file formats.

NPCEditor provides monitoring and control functionality over the individual system components using a GUI *character editor*. An NPCEditor window consists of several tabbed panels each corresponding to a particular function. These panels are listed below, along with how they are used by the user classes defined in Section 1.

1. Utterances: a character designer specifies answers and sample questions, assigns them to individual characters, and links them to each other.
2. Settings: a designer creates and modifies annotation categories and labels, assigns colors to labels, and specifies whether a category should be used as a non-lexical feature for classification.
3. People: a designer specifies available characters and edits the character properties. An administrator uses the panel to specify how the characters are connected to the network.
4. Classifier: a designer has to train the classifier parameters after creating or modifying the character language database. The training process is as simple as pressing a single button on the Classifier panel. However the panel also provides some advanced classifier tuning capabilities for expert designers.
5. Conversations: an administrator can monitor the existing conversations.
6. Chat: an interactor can pose arbitrary questions to the characters in the database and observe how the classifier ranks the available responses. Also, the audience can see details of the system performance. An administrator and designer can debug the characters.

Figure 1b 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, a developer can define the utterance text, speaker, assign a text-based identifier and annotation labels. To link a question to an answer, the developer 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. More details about the interface and uses may be found in (Leuski and Traum 2010).

4. Evaluation

We have evaluated NPCEditor in a number of off-line and on-line experiments. We have tested the classification accuracy, robustness to errors in the classifier input, and user engagement in interactions with a virtual human. In this section we summarize some of these experiments. More details about the experimental setup and the results can be found elsewhere (Leuski et al. 2006; Artstein et al. 2009; Leuski and Traum 2008; Kenny, Parsons, and Rizzo 2009). Evaluations of the performance of characters built using the NPCEditor are briefly described in Section 5.

Classification Accuracy

In the first set of experiments we have evaluated the classification accuracy or how often the first answer returned by the system was appropriate. As the baseline we used a text classification approach based on Support Vector Machines (SVM). We represented questions as vectors of term features and the linked answers defined the question classes. We tokenized the questions and stemmed the tokens using the KStem algorithm (Krovetz 1993) in exactly the same way as we tokenize the text to compute language models. We used a $tf \times idf$ weighting scheme to assign values to the individual term features (Allan et al. 1998). Finally, we trained a multi-class SVM (SVM^{struct}) classifier with an exponential kernel (Tsochantaridis et al. 2004). We also experimented with a linear kernel function, various parameter values for the exponential kernel, and different term weighting schemes. The reported combination of the kernel and weighting scheme showed the best classification performance. Such an approach is well-known in the community and has been shown to work well in numerous applications (Joachims 1998). We believe it provides us with a strong baseline.

As the second baseline we used the language model approach described in Section 2, but we compared questions to questions instead of comparing them to answers. This is equivalent to single-language retrieval without the translation effect of the question-answer mapping. Specifically, in Equation 5 we use the likelihood over question terms and sum over all sample questions. Given an input question, this technique retrieves the most similar sample question, and we return the answer linked to that question.

To evaluate the systems we used the language database from the SGT Blackwell virtual human (see Section 5). The database contains 1,261 questions and 60 answer classes. We divided the collection of questions into training and testing subsets following the 10-fold cross-validation schema and calculated the effectiveness of each approach.

	accuracy	impr. over SVM	avg. prec.
SVM	53.13		
SLM	57.80	8.78	63.88
CLM	61.99	16.67	65.24

Table 1: Comparison of three different algorithms for answer selection on SGT Blackwell data. Each performance number is given in percentages.

We use two evaluation metrics to compare the systems'

performance. Firstly, we calculate the classification accuracy or how often the first answer returned by the system was appropriate. Table 1 shows the accuracy numbers for the two baselines (we call them "SVM" and "SLM") and the NPCEditor classification ("CLM"). The NPCEditor classification approach is 17% more accurate than the SVM baseline. It is also more accurate than the SLM baseline. The differences shown are statistical significant by t-test ($p < 0.05$).

Secondly, recall that both SLM and CLM methods are ranking techniques. They order the answers in the database by their expected appropriateness. Both methods may return several candidate answers depending on the threshold value. Thus, an interactor may get a different response if she repeats the question (if there is more than one good answer). We want to measure the quality of the ranked list of candidate answers. As our second evaluation metric we use interpolated average precision—a well-known IR measure that for every appropriate answer in the list of candidates computes the proportion of appropriate answers among all preceding answers and averages those values. We show the average precision numbers for the SLM and CLM runs. The average precision scores are significantly higher for the cross-language approach than for the single-language approach. It indicates that the CLM approach tends to rank appropriate answers above non-appropriate answers more often than the SLM approach.

We have repeated the experiment on 7 other virtual characters with smaller language databases. We observed that our system is more effective on problems with more answer classes.

Classifier Robustness

In the second set of experiments we evaluated the classifier robustness to input errors. Recall that in a typical VH system the text input to the classifier comes from an ASR module, which may contain speech recognition errors. We thus examined the impact of ASR quality on answer quality. We recruited 20 participants to interview the SGT Blackwell character. Each participant asked 20 questions. We computed the Word Error Rate (WER) for each ASR-transcribed question. A word error rate is the ratio of the total number of word errors in a string (substitutions, deletions, and insertions) to the number of words in the correct string. Note that WER can be greater than 100%. The average WER score was 37.33%.

We applied the NPCEditor classification approach to both ASR and human transcribed data and recorded the selected answers. We asked three human raters to judge the appropriateness of the selected responses using a 1-6 scale (Gandhe et al. 2004). The Cronbach's alpha value, measuring the inter-rater agreement, was above 0.91 indicating high consistency among the judges.

To judge the impact of ASR errors on classification appropriateness, we computed the cumulative average appropriateness score (CAA) as a function of WER: for each WER value t we average the appropriateness scores for all questions-answer pairs with WER score less than or equal to t, as shown in equation 8, where p is a question-answer pair, $A(p)$ is the appropriateness score for p, and $E(q_p)$ is the WER score for the ASR output of a spoken question q_p. CAA(t) is thus the expected value of the appropriateness score if the WER is at most t.

$$CAA(t) = \frac{1}{|S(t)|} \sum_{p \in S(t)} A(p), S(t) = \{p | E(q_p) \leq t\} \quad (8)$$

We computed two sets of $CAA(t)$ values: one using the appropriateness scores A for ASR-transcribed questions and the other using the appropriateness score for the human transcribed questions. We examined the differences between these scores at different values of WER. We observed that the differences are small and not statistically significant until WER reaches 60%. After that point the CAA score is significantly lower on the ASR transcribed data (by t-test with $p < 0.05$). We concluded that the classifier performance is not significantly affected by the input errors if the amount of error does not exceed 60%.

Interaction Quality

Kenny and his colleagues (Kenny, Parsons, and Rizzo 2009) study virtual humans for clinician training. They have built a virtual human using NPCEditor that plays a role of a patient with a psychiatric problem and they wanted to assess whether a virtual patient would respond to the clinician interview as a real patient would. They wanted to see if 1) clinicians could elicit proper responses from questions relevant for an interview from a virtual patient and 2) to evaluate psychological variables such as openness and immersion of the participant and believability of the character as a patient. They have engaged 15 test subjects from a medical school including medical students, psychiatry residents and fellows. Each subject conducted a 15 minute interview with the virtual patient trying to diagnose her condition and filled out a set of questionnaires before and after the interview. The researchers analyzed the data from the interview transcripts and from the questionnaires and found that the subjects were generally immersed in the interviews, they described the virtual patient character as believable and engaging, and they did ask and received responses covering all aspects of a typical patient interview. The study showed a feasibility of using virtual patients for training.

5. Applications

NPCEditor has been used as the language processing component for over a dozen virtual humans at ICT (some with multiple versions), and several dozen elsewhere. Over a dozen different developers have so far used the system to create or extend characters. The system has been deployed and administered in museums, in virtual worlds, at trade shows and conferences, and in mobile vans by people not involved in their development. Thousands of people have interacted with these systems, and even more have seen them as audience to live interactions. In this section we describe some of the installations, highlighting their unique features.

SGT Blackwell Originally created as a showcase of VH technology, SGT Blackwell was introduced at the 2004 Army Science Conference. He is a life-size 3D US Army

soldier projected onto a transparent screen in a mixed-reality environment. Conference attendees acted as audience with ICT demoers who spoke to SGT Blackwell. The original domain had 83 responses on different topics covering his identity, origin, language and animation technology, design goals, our university, the exhibition setup, and some miscellaneous topics, such as "what time is it?" and "where can I get my coffee?" After a lot of positive feedback from the attendees, several subsequent versions were built, using the NPCEditor. An extended version with additional domain items was used for demos both at ICT and by the office of the US Army's Director for Research and Laboratory Management, with external administrators and demoers. It was also selected as a part of the Smithsonian's National Design Triennial, *Design Life Now* exhibit in 2006. The system was installed at the Cooper-Hewitt Museum in New York from December 2006 to July 2007 (and later at two other museums), where SGT Blackwell was administrated by Museum staff and interacted directly with over 100,000 visitors. Limited versions of SGT Blackwell were also created specially for the Director for Research and Laboratory Management to interact with at the opening and closing of the 2006 Army Science conference, as well as a cameo appearance with SGT Star at the 2008 conference. An example of dialogue with SGT Blackwell can be found in (Leuski et al. 2006). Preliminary evaluation of Blackwell in the Cooper-Hewitt can be found in (Robinson et al. 2008).

SGT Star Interactive SGT Star was funded by the US Army Accessions Command, who wanted a mobile, life-sized, face to face version of their web character from goarmy.com[4]. SGT Star is like SGT Blackwell a life-size rendering of an Army soldier that answers questions on topics including Army careers, training, education and money for college[5]. He can also handle queries about the technology behind his development and explain how his creation fits in with plans for future Army training environments. There are approximately 320 answers in his repertoire. The original version was used by Accessions command at trade shows and has since been ported to several "Army Adventure Vans" in which Army educational personnel interact with SGT Star about Science and Army careers. The character database was constructed by a linguist in our lab, with consultation from scriptwriters and Army SMES and is administered and interacted with by the vans' Army staff. More about SGT Star, including a longitudinal evaluation at several conventions can be found in (Artstein et al. 2009).

Virtual Patients for Clinical Training Since 2006, the NPCEditor has been used to create virtual characters exhibiting psychological conditions who can interact verbally and non-verbally with a clinician in an effort to teach the clinician interpersonal skills such as interviewing and diagnosis. Three virtual patient characters were developed by a separate team at the institute without the direct involvement of the NPCEditor creators. Each character database contained up to 200 responses. Users were medical and psychology students (Kenny, Parsons, and Rizzo 2009) .

Army Communication Skills Training Since 2006 NPCEditor has been successfully used by the Program Executive Office (PEO) Simulation, Training, and Instrumentation (STRI), US Army, as a natural language understanding and processing component in a number of interactive training systems that teach soldiers communication and culture-specific conversational skills. We have received very positive feedback about NPCEditor from designers and developers of the training systems. These systems have been fielded in 19 training installations. As of this writing, more than 3,000 soldiers (commissioned and noncommissioned) have received training using the system. An independent analysis has shown that the US Army has achieved significant savings (as much as $30 million) in training systems research and development costs by reusing this existing system and have realized greater flexibility in the ability to respond to theater driven changing training requirements[6]. The designers and administrators are Army or contracted personnel outside ICT, and the interactors are soldiers, using the systems for training.

Virtual World Guides Since 2008, NPCEditor has been used to develop several AI avatars in online virtual worlds including Second Life and Active Worlds. These characters are used for aides in educational settings as well as guides of the virtual space. These characters have been designed and administrated at ICT, but the interactors were people in the virtual worlds who came to visit the areas and interact with the characters. In contrast to the other characters described in this section, the online virtual world characters do not use speech recognition but the native virtual world chat and IM facilities. Probably the most advanced of these is LT Moleno, a staff duty officer, who patroled the US Army Welcome island in Second Life for over a year. He answered questions about the island and conducted interactive tours of the island facilities. Over 4,000 visitors interacted with LT Moleno. More details on LT Moleno can be found in (Jan et al. 2009).

Gunslinger The Gunslinger project (Hartholt et al. 2009) is a mixed-reality interactive-entertainment experience that combines physical props with virtual humans. The participant physically walks into a saloon room situated somewhere in Wild West and interacts with three life-sized virtual human characters projected onto screens built into the saloon walls. The characters listen and talk to the participant and to each other. They observe and react to participant's location in the room and his actions. For example, they notice when the participant takes out his gun and may comment on it. NPCEditor handles both the language understating and dialogue management parts of the system. It uses a state-based dialogue manager which tracks the participant's progress through the scenario and handles normal conversation flow. The dialogue manager allows different responses to interactor interruption and allows the characters to exhibit initiative and start their own discussion topics.

[4]The website version was developed by Next IT Corporation and does not share any technology with the ICT version.

[5]A video of an early version of the SGT Star character can be found at our website: `http://projects.ict.usc.edu/nld/group/videos/early-version-sgt-star`.

[6]Personal communication. The report is not publicly available.

The character database and the dialogue manager script were created by the Gunslinger group at the institute. The system is setup for a single interactor, however hidden cameras at the set allow an audience to observe the interaction remotely.

InterFaces In the fall of 2008 the InterFaces exhibit opened up at the Boston Museum of Science[7]. The stars of the exhibit are Ada and Grace—two virtual docents who have approximately 400 answers for questions about computers, robots, and communications, as well as themselves and exhibits in the museum's Cahners Computer Place (Swartout et al. 2010)[8]. The target audience for the exhibit are children from ages 7 to 14. NPCEditor drives language understanding and dialogue management for both characters. The system is operated by the museum volunteers and one of the volunteers generally serves as the designated interactor.

Virtual Human Toolkit NPCEditor is being used to create virtual humans in more and more diverse applications. It is now part of a Virtual Human Toolkit that is a collection of modules, tools and libraries that allow developers to create their own virtual humans. The toolkit is available without cost for academic research purposes[9]. In September 2008 we conducted a 3 day workshop, where approximately 30 attendees, mostly graduate students from universities across the country, designed and built 6 different characters for a game of "Clue" over two afternoons. Each character had approximately 30 to 40 responses. This illustrates how quickly a novice character designer can develop a useful virtual human.

6. Future Work

One of the weakness of the classification-based language processing is the requirement to collect sufficient language data. The character designer has to define all possible answers, specify sample questions for the answers, and link them together. We have experimented with characters that have as many as 400 answers and 2,000 questions in their language databases. Our partners in the Army have developed characters with 2,000 answers and more than 20,000 questions. The classification approach scales well for larger datasets. However, creating and expanding the language resources while maintaining consistency in the data can become very tedious. We plan to incorporate ideas from active learning into NPCEditor to help with linking of questions and answers. We are exploring techniques for automatic question and answer generation from descriptive text. We are also interested to investigate solutions for mining relevant language data from chat archives or world wide web.

We have shown that the classification approach is robust to the errors in the input text when conditions permitting. However, the ASR performance degrades in noisy environments or when dealing with heavily accented speech. We are looking into alternative representations for the input text to improve the classifier robustness even further. For example, our recent experiments show that incorporating phonetic information into the input for the classifier provides small but significant improvement in quality. We are also exploring how to integrate multiple ASR hypotheses into the input representation.

NPCEditor started as a text classification tool for context-free question-answering dialogue where any question can be asked at any time and the answer depends only on the question content. This type of interaction works well for kiosk-like applications in museums or show exhibits. As we are exploring other VH application domains there is a growing need to support more sophisticated dialogue behaviors. There are two way that we can approach this. Firstly, we can integrate complex dialogue handling into NPCEditor itself. Currently NPCEditor allows a designer to incorporate context information by defining multiple states for a single character, specify a classifier for each state, and indicate when the inter-character state transitions occur. It integrates the classifier technology with a dialogue scripting module facilitating creation of characters with complex behavior. One of our goals is to continue the development of NPCEditor, refining the support for complex dialogue strategies. We are looking into building components to visualize the dialogue state network, tools for debugging the dialogue transitions and state information.

Secondly, we can leave the dialogue management to a specialized component. We have built systems where we use the NPCEditor language classification capability to convert from the text input into a semantic representation and pass this information to a specialized dialogue manager (Leuski and Traum 2008; Gandhe et al. 2008). Currently this connection is one-way: there is no feedback from the dialogue manager for the classifier. We are experimenting with techniques for integrating dialogue context information into the classification process with the hope that it will increase the accuracy of the classification.

7. Conclusions

In this paper we presented NPCEditor, a system for building and deploying virtual characters capable of engaging a user in spoken dialog on a limited domain. NPCEditor has been used mainly for question answering characters where an interactor asks questions and the character responds. However other types of dialogue have also been successfully implemented, for example, the Gunslinger system, in which characters take the initiative and question the user. The dialog may have other forms as long as character responses can be fully specified a priori.

NPCEditor contains a state of the art cross-language information retrieval-based classifier that is robust to noisy input from speech recognition results. It contains a development environment that includes a user-friendly GUI to support several classes of user, from developer to interactor and audience. NPCEditor has been successfully evaluated in the laboratory and field-tested and proved to be an effective and versatile system in a number of different applications.

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

[8] A video of the characters is available at our website: http://projects.ict.usc.edu/nld/group/videos/jillian-talking-ada-and-grace

[9] For more information, see our website: http://vhtoolkit.ict.usc.edu/

Acknowledgments

We would like to thank the users of NPCEditor for many helpful suggestions that led to specific improvements in usability and functionality. We also would like to thank the reviewers for their invaluable suggestions and comments helping us to improve the paper. 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.

References

Allan, J.; Callan, J.; Croft, W. B.; Ballesteros, L.; Byrd, D.; Swan, R.; and Xu, J. 1998. Inquery does battle with TREC-6. In *Sixth Text REtrieval Conference (TREC-6)*, 169–206.

Artstein, R.; Gandhe, S.; Gerten, J.; Leuski, A.; and Traum, D. R. 2009. Semi-formal evaluation of conversational characters. In *Languages: From Formal to Natural*, 22–35.

Bahl, L. R.; Jelinek, F.; and Mercer, R. L. 1990. A maximum likelihood approach to continuous speech recognition. In *Readings in speech recognition*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 308–319.

Blei, D. M.; Ng, A. Y.; Jordan, M. I.; and Lafferty, J. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Chu-Carroll, J., and Carpenter, B. 1999. Vector-based natural language call routing. *Journal of Computational Linguistics* 25(30):361–388.

Gandhe, S.; Gordon, A.; Leuski, A.; Traum, D.; and Oard, D. W. 2004. First steps toward linking dialogues: Mediating between free-text questions and pre-recorded video answers. In *Proceedings of the 24th Army Science Conference*.

Gandhe, S.; DeVault, D.; Roque, A.; Martinovski, B.; Artstein, R.; Leuski, A.; Gerten, J.; and Traum, D. 2008. From domain specification to virtual humans: An integrated approach to authoring tactical questioning characters. In *Proceedings of Interspeech Conference*.

Gorin, A.; Riccardi, G.; and Wright, J. 1997. How may i help you? *Speech Communication* 23:113–127.

Gratch, J.; Rickel, J.; Andre, E.; Cassell, J.; Petajan, E.; and Badler, N. 2002. Creating interactive virtual humans: Some assembly required. *IEEE Intelligent Systems* 54–63.

Gratch, J.; Wang, N.; Okhmatovskaia, A.; Lamothe, F.; Morales, M.; van der Werf, R.; and Morency, L.-P. 2007. Can virtual humans be more engaging than real ones? In Jacko, J., ed., *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments*, volume 4552 of *Lecture Notes in Computer Science*, 286–297. Springer Berlin / Heidelberg.

Grefenstette, G. 1998. *Cross-Language Information Retrieval*. Norwell, MA, USA: Kluwer Academic Publishers.

Hartholt, A.; Gratch, J.; Weiss, L.; Leuski, A.; Morency, L.-P.; Marsella, S.; Liewer, M.; Thiebaux, M.; Doraiswamy, P.; and Tsiartas, A. 2009. At the virtual frontier: Introducing Gunslinger, a multi-character, mixed-reality, story-driven experience. In Ruttkay et al. (2009), 500–501.

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd International ACM SIGIR Conference*, 50–57.

Jan, D.; Roque, A.; Leuski, A.; Morie, J.; and Traum, D. R. 2009. A virtual tour guide for virtual worlds. In Ruttkay et al. (2009), 372–378.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, Lecture Notes in Computer Science. chapter 19, 137–142.

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

Katz, B. 1988. Using english for indexing and retrieving. In *Proceedings of the 1st RIAO Conference on User-Oriented Content-Based Text and Image Handling (RIAO '88)*.

Kenny, P. G.; Parsons, T. D.; and Rizzo, A. A. 2009. Human computer interaction in virtual standardized patient systems. In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part IV*, 514–523. Berlin, Heidelberg: Springer-Verlag.

Krovetz, R. 1993. Viewing morphology as an inference process. In *Proceedings of the 16th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 191–202.

Lavrenko, V. 2004. *A Generative Theory of Relevance*. Ph.D. Dissertation, University of Massachusetts at Amherst.

Leuski, A., and Traum, D. 2008. A statistical approach for text processing in virtual humans. In *Proceedings of the 26th Army Science Conference*.

Leuski, A., and Traum, D. 2010. NPCEditor: A tool for building question-answering characters. In *Proceedings of The Seventh International Conference on Language Resources and Evaluation (LREC)*.

Leuski, A.; Patel, R.; Traum, D.; and Kennedy, B. 2006. Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*.

Lewis, D. D.; Schapire, R. E.; Callan, J. P.; and Papka, R. 1996. Training algorithms for linear text classifiers. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 298–306.

Ponte, J. M., and Croft, W. B. 1997. Text segmentation by topic. In *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, 120–129.

Reeves, B., and Nass, C. 1996. *The Media Equation*. Cambridge: Cambridge University Press.

Robinson, S.; Traum, D.; Ittycheriah, M.; and Henderer, J. 2008. What would you ask a conversational agent? observations of human-agent dialogues in a museum setting. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.

Roque, A., and Traum, D. 2007. A model of compliance and emotion for potentially adversarial dialogue agents. In *Proceedings of the 8th SIGdial Workshop*.

Ruttkay, Z.; Kipp, M.; Nijholt, A.; and Vilhjálmsson, H. H., eds. 2009. *Intelligent Virtual Agents, 9th International Conference, IVA 2009, Amsterdam, The Netherlands, September 14-16, 2009, Proceedings*, volume 5773 of *Lecture Notes in Computer Science*. Springer.

Swartout, W. R.; Traum, D. R.; Artstein, R.; Noren, D.; Debevec, P. E.; Bronnenkant, K.; Williams, J.; Leuski, A.; Narayanan, S.; and Piepol, D. 2010. Ada and grace: Toward realistic and engaging virtual museum guides. In *IVA*, 286–300.

Traum, D.; Swartout, W.; Gratch, J.; Marsella, S.; Kenney, P.; Hovy, E.; Narayanan, S.; Fast, E.; Martinovski, B.; Bhagat, R.; Robinson, S.; Marshall, A.; Wang, D.; Gandhe, S.; and Leuski, A. 2005. Dealing with doctors: Virtual humans for non-team interaction training. In *Proceedings of ACL/ISCA 6th SIGdial Workshop on Discourse and Dialogue*.

Tsochantaridis, I.; Hofmann, T.; Joachims, T.; and Altun, Y. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first international conference on Machine learning*.

Voorhees, E. M. 2003. Overview of the trec 2003 question answering track. In *Proceedings of The Twelfth Text Retrieval Conference*, 54–69.

Wei, X., and Croft, W. B. 2006. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, 178–185. New York, NY, USA: ACM.