From Domain Specification to Virtual Humans: An integrated approach to authoring tactical questioning characters

Sudeep Gandhe, David DeVault, Antonio Roque, Bilyana Martinovski,
Ron Artstein, Anton Leuski, Jillian Gerten, David Traum

USC Institute for Creative Technologies, Marina del Rey, California
traum@usc.ict.edu

Abstract

We present a new approach for rapidly developing dialogue capabilities for virtual humans. Starting from domain specification, an integrated authoring interface automatically generates dialogue acts with all possible contents. These dialogue acts are linked to example utterances in order to provide training data for natural language understanding and generation. The virtual human dialogue system contains a dialogue manager following the information-state approach, using finite-state machines and SCXML to manage local coherence, as well as explicit modeling of emotions and compliance level and a grounding component based on evidence of understanding. Using the authoring tools, we design and implement a version of the virtual human Hassan and compare to previous architectures for the character.
Index Terms: spoken dialogue systems, virtual humans

1. Introduction

In this paper we report on the specific advances of our third generation architecture for Virtual Humans for Tactical Questioning. The aim of this work is to produce tools that non-experts in natural language dialogue can use to create characters to serve as virtual role-players in training environments for tactical questioning. *Tactical Questioning* is when military personnel, usually on patrol, hold conversations with individuals to receive information of military value. As well as simply extracting specific information it can also involve goals of building rapport and gathering general information about the local area. There are different questioning tactics that can be employed, from the "direct" approach, to offering incentives, or raising fear of the consequences of non-compliance. Different tactics have different chances of success with different interviewees, and part of the challenge is to learn when it is appropriate to use which tactics and what the consequences will be.

The work described in this paper directly builds on previous versions of the architecture, as reported in [1, 2]. Our first architecture, described in [1], required little technical knowledge to author, since one simply linked up questions (or other inputs) to responses. Scripted inputs were augmented with paraphrases and novel questions encountered in wizard and system testing, in order to reach good classifier performance on unconstrained spoken input. This kind of system has some limitations for a training system, however, since it is not flexible to the different personalities or attitudes the character may wish to exhibit and is not affected much by the trajectory of the dialogue. While there are dialogue models available for tracking the progression of a virtual human character's emotion and dialogue state (e.g. [3, 4]), these are designed for more complex domains, with negotiation of coordinated actions rather than just information exchange, and have heavier authoring demands. Instead we chose to make minimal extensions to the text-driven system, resulting in our second version, reported on in [2]. This system used sets of linked question-answer domains, one for each compliance level of the character (e.g. compliant, reticent, adversarial,..), as well as additional recognizers for relevant dialogue move, topic, and politeness. We added a dialogue manager [5] that can track several emotional and social variables, and decide on which of the possible responses to produce, depending on the chosen compliance level. The second version produced improvements in the consistency of responses within a session and global trajectory of the agent personality, as well as variability across sessions, but did not do a good job of maintaining intermediate-level coherence. Local coherence between initiative and response is well handled by the classifier technology, but a problem arises when the characters incur obligations or conditional obligations [6] that are not immediately addressed. For example, when asked about sensitive information, a character might first elicit assurances before providing the information. When assurances are given, the character should be able to provide the information without being queried specifically again, as was required in the system in [2]. Moreover, even if the question is asked again, it might be done in an elliptical or anaphoric manner, which the character must recognize as relevant to the previous questioning.

We thus needed to add additional resources for dialogue processing at the meaning level, while still keeping the domain authoring process non-technical enough so that domain experts with little computational linguistic training can easily use it. The specific enhancements described in this paper include:

- an integrated authoring environment to help manage the connections between domain, dialogue, and text levels
- an expansion of the dialogue manager to track additional aspects of local dialogue coherence, including deferred obligations and motivations for answering a question, and grounding and other subdialogues
- modules to increase the dialogue performance and decrease authoring burden using anaphora tracking and stylistic generation enhancements

Our authoring process starts with the domain knowledge construction layer, as described in Section 2. From this we are able to automatically construct the relevant dialogue acts that are used by the dialogue manager components for tracking context and deciding on system dialogue moves as described in sec-

This work 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 U. S. Government, and no official endorsement should be inferred.

tion 3. The authoring tools also allow direct linking of these acts to surface text for NLU and NLG, as described in section 4.

For comparison purposes, we have implemented versions of the same character, Hassan, in our three architectures. The scenario for Hassan takes place in contemporary Iraq. In a fictional storyline, the trainee talks to Hassan, a local politician, to determine who has placed an illegal tax on a local marketplace.

2. Domain Knowledge Level

Domain knowledge is created as a four level hierarchy. The highest level is the *characters*, the conversational participants in the domain, e.g. the trainee (called player) and Hassan, who can be speakers and addressees of utterances and dialogue acts. Each character knows about a set of *objects*. These objects can be of different types such as person (imam), location (market) or abstract concept (tax). Each object can be further described by *attributes*. Finally, attributes can take on *values*, some of which can be marked false - to be used as lies. A basic proposition is a triple <object,attribute,value>. Objects of type person can also have representations of the actions they can perform (e.g. offers, threats, admissions), their *goals*, and their *attitudes* toward other objects. These additional aspects are used to create policies for how to engage in questioning dialogues, as described in the next section. Another aspect crucial for the tactical questioning is social behavior, oriented more generally to the characters rather than specific objects. We have sets of these kinds of content, including *compliments* and *insults*.

We use a simple XML representation of these domain knowledge aspects, for ease of use across modules of the system at both domain specification and run-time. Figure 1 shows parts of the domain specification used for the Hassan scenario, including an attribute and an action for Hassan, and both a true and false (i.e. lie) value for an attribute about the tax

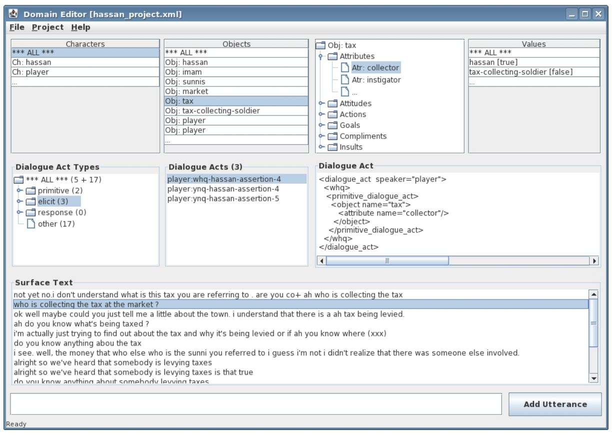
```
<domain name="hassan">
  <character name="hassan">
    <object name="hassan" type="person">
      <attribute name="role">
        <value>middle-man</value>
      </attribute>
      <actions>
        <offer name="cooporate"/>
      </actions>
    </object>
    <object name="tax" type="abstract">
      <attribute name="collector">
        <value>hassan</value>
        <value isTruth="false">
          tax-collecting-soldier
        </value>
      </attribute>
      ...
```

Figure 1: Aspects of the *Hassan* domain

In order to more easily author domain content, we have constructed a GUI, shown in Figure 2. The top pane shows the domain creation aspects, where, moving from left to right, authors can add or delete characters, objects, attributes (or other object contents) and values. The GUI will automatically construct XML like Figure 1 when the author adds fields to the GUI.

3. Dialogue Level

Once the domain is defined, it needs to be linked up with the language that will be used to refer to it. Dialogue acts form



*** ALL ***
Ch: hassan
Ch: player
...

*** ALL ***
Obj: hassan
Obj: imam
Obj: sunnis
Obj: market
Obj: tax
Obj: tax-collecting-soldier
Obj: player
Obj: player
...

Obj: tax
Attributes
Atr: collector
Atr: instigator
...
Attitudes
Actions
Goals
Compliments
Insults

*** ALL ***
hassan [true]
tax-collecting-soldier [false]
...

*** ALL *** (5 + 17)
primitive (2)
elicit (3)
response (0)
other (17)

player:whq-hassan-assertion-4
player:ynq-hassan-assertion-4
player:ynq-hassan-assertion-5

```
<dialogue_act  speaker="player">
  <whq>
   <primitive_dialogue_act>
     <object name="tax">
        <attribute name="collector"/>
      </object>
    </primitive_dialogue_act>
  </whq>
</dialogue_act>
```

not yet no.i don't understand what is this tax you are referring to . are you co+ ah who is collecting the tax
who is collecting the tax at the market ?
ok well maybe could you just tell me a little about the town. i understand that there is a ah tax being levied.
ah do you know what's being taxed ?
i'm actually just trying to find out about the tax and why it's being levied or if ah you know where (xxx)
do you know anything abou the tax
i see. well, the money that who else who is the sunni you referred to i guess i'm not i didn't realize that there was someone else involved.
alright so we've heard that somebody is levying taxes
alright so we've heard that somebody is levying taxes is that true
do you know anything about somebody levying taxes

Figure 2: A tool for designing the domain, dialogue acts and the utterances that map to those dialogue acts.

the middle level in this link, having domain aspects as their contents and being identified directly as the interpretations of language utterances. In 3.1, we describe the basic meaning of dialogue acts and how they are created. In 3.2, we describe several new aspects of the dialogue manager, which makes use of these dialogue acts to update the information state and decide what content to express.

3.1. Dialogue Acts

Our dialogue manager reasons about several standard types of dialogue acts, including *assertions*, *yn-questions*, *wh-questions*, *compliments*, *offers*, *threats*, *insults*, and *elicitations* and *responses* for most acts. Figure 3 shows our XML representation of some of these acts, which contain a speaker (one of the characters), an act-type, and contents.

Most dialogue acts are automatically created from the domain representation described in Section 2. E.g. all <object,attribute,value> triples for a character can serve as the contents of an *assert* for that character. Likewise, any <object,attribute> pair for that character can serve as a WHQ for other characters.

The middle pane of the authoring GUI shown in figure 2 allows selection from among the full set of dialogue acts and all contents understood by the character. The left pane allows selection of the type of dialogue act; the middle pane lets one select individual dialogue acts; the right pane shows the full XML content. We also generate some generic dialogue acts that are customary in human-human conversations like *greeting* and *closing*, that are not tied to specific domain content. Besides the automatically generated acts that come from content specification, the user can create special-purpose dialogue acts directly at this level, if necessary, using the GUI.

3.2. Dialogue Manager

The main responsibilities of the dialogue manager are to update the information state and dialogue history and to select content to be generated. The dialogue manager gets input dialogue acts from the NLU and outputs dialogue acts to the NLG. It decomposes the dialogue acts in order to update the information state. Our previous dialogue manager [5] made use of hand-authored rules for tracking affective variables and offers and threats made. It used these to compute a *compliance level*,

hassan.assert

```
<dialogue_act speaker="hassan">
  <primitive_dialogue_act>
    <assertion>
      <object name="tax">
        <attribute name="collector">
          <value>hassan</value>
        </attribute>
      </object>
    </assertion>
  </primitive_dialogue_act>
</dialogue_act>
```

Indeed, you might say that I collect the taxes.

---

player.offer

```
<dialogue_act speaker="player">
  <primitive_dialogue_act>
    <offer name="give-money"/>
  </primitive_dialogue_act>
</dialogue_act>
```

We can offer you financial reward.

---

hassan.elicit-offer

```
<dialogue_act speaker="hassan">
  <elicit>
    <primitive_dialogue_act>
      <offer name="give-money"/>
    </primitive_dialogue_act>
  </elicit>
</dialogue_act>
```

I might tell you what you want if there was something in it for me.

Figure 3: Sample dialogue acts automatically generated from the *Hassan* domain along with example utterances.

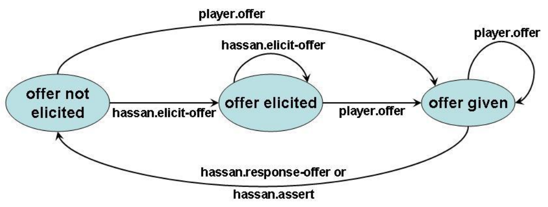
which would dictate how the character would respond. In that system the response was still calculated using text-to-text mappings, requiring complete input-text to output-text mappings for all compliance levels. By moving to the use of dialogue acts and domain content, we are able to reason at a more abstract level about how to respond. Two new components of the dialogue manager are described below.

3.2.1. Response Generation

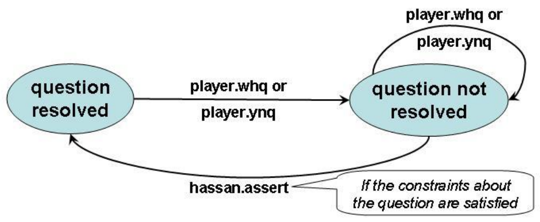
The response generation component keeps track of the dialogue history and outputs a dialogue act in response to an input dialogue act. The system uses the information-state approach to dialogue modeling [7]. The information state update rules are written as augmented transition networks or state charts. We are using SCXML, a W3C standard [8], for describing the state charts. SCXML allows for explicit data models that can be manipulated by executable code. This code can be triggered on entry or exit from a state or transition. A set of networks can also be run in parallel fashion. As pointed out by [9], all these features make it viable to implement the information-state based dialogue model with SCXML.[1]

We have a set of networks for each type of subdialogue. These networks model Hassan's conversational obligations, following [6]. Figure 4(a) shows a sample network that handles dialogue acts for the offer subdialogue. The outgoing arcs from the currently active states denote all possible dialogue acts that

[1]We used the apache commons SCXML implementation. Available at http://commons.apache.org/scxml



(a) offer subdialogues



(b) question-answer subdialogues

Figure 4: State charts for *Hassan* domain.

can be generated as a response by the system or can be handled as input from the user. Some of these transitions can be conditional and depend on the data model configuration (i.e. information-state).[2] The character's compliance level is also used to decide what kind of reply to make. E.g., when adversarial, the character may choose to lie in response to questions, if a lie is available.

			<question_resolved, offer_not_elicited>
1	P	whq	Ok I'm trying to understand where the local taxation is coming from?
			<question_not_resolved, offer_not_elicited>
2.1	H	grounding	So you want to talk about the taxes.
2.2	H	elicit-offer	I might tell you what you want if there was something in it for me.
			<question_not_resolved, offer_elicited>
3	P	offer	We can offer you financial reward.
			<question_not_resolved, offer_given>
4	H	assert	Please understand, I collect taxes for my Imam. All in service to Allah.
			<question_resolved, offer_not_elicited>
5	P	whq	And what is his name?
			<question_not_resolved, offer_not_elicited>
6	H	elicit-offer	My friend, if people find out that I tell you this, it would be a problem for me.

Figure 5: Example dialogue, P is the player (human trainee) and H is Hassan. The currently active states are shown within <>.

As an example, in the dialogue from Figure 5, the player asks a sensitive question (utterance 1), the constraints for which are not yet satisfied. At this point Hassan chooses to elicit an offer. The particular offer to be elicited depends on the constraints for the question under discussion. After utterance 3 the constraints are met. Hassan can either choose to answer the question (*hassan.assert*) or respond to the offer (*hassan.response-offer*). In compliant mode hassan chooses to go with more in-

[2]The constraints and policies for these networks are currently hand-authored, but our future plans are to include a capability to author these in the GUI. Consistent with our design approach, the domain author will be expected to select from a set of reasonable subdialogues for a given type of content rather than engineer subdialogue networks from first principles (though that option will be available if needed).

formative option as in utterance 4.

3.2.2. Grounding

A grounding component tracks the extent to which the dialogue participants have achieved mutual understanding of the material being discussed. The grounding module manages behavior such as explicit and implicit confirmations and requests for repetitions. The grounding module uses domain knowledge when making decisions about the extent to which material must be grounded. Further details about the grounding module's integration into this system are presented in [10].

4. Textual Level

The remaining pieces of the system involve converting from surface text to dialogue acts and back again. The authoring GUI shown in Figure 2 supports this via links between natural language texts in the bottom pane, and dialogue acts in the middle pane. For each dialogue act from the character, the author can add one or more options for the character to realize this act. Likewise, for the player dialogue acts, the author can link possible ways for the player to produce this act.

4.1. Natural Language Understanding & Generation

The NLU uses a statistical language modeling text classification technique [1] to map the text produced by the speech recognition to dialogue acts. It requires a training corpus of sample utterances linked to dialogue acts, which can be produced in the authoring GUI as described above. This technique generalizes from the set of authored texts to find the closest dialogue act match (or *unknown* if no candidate exceeds a threshold). Using the GUI and automatically created dialogue acts helps lighten the authoring burden and ensure consistency and coverage of the training set.

The NLG uses the same statistical classification techniques, but mapping from dialogue acts to surface text. The NLG output can be modified by a style generator described below.

4.2. Anaphora Resolution

Tactical questioning involves extended dialogues in which information about particular objects and individuals arrives incrementally. In this setting, it is common for the trainee to use pronouns to refer to individuals under discussion, as illustrated by the use of *his* in utterance 5 in Figure 5. Our system resolves such pronouns, and uses the identified antecedent text (in this case *my Imam*) to create a more informative text for the NLU to interpret. In this example, the result is that NLU receives the text *And what is your imam's name?* rather than the trainee's actual spoken text *And what is his name?*. To identify a pronoun's antecedent, we use the machine learning approach of [11], which learns how to select the correct antecedent using a corpus of annotated versions of domain-specific dialogues such as that in Figure 5.

4.3. Style Generation

We have also further reduced the authoring burden while allowing productive expression of stylistic features by adding a hybrid statistical-grammar based style generator as a post-processor to the NLG. The Style Generator uses the compliance level to generate the styled text output. For example, utterance 6 in Figure 5 begins with the phrase *My friend*, which Hassan uses as a deliberate signal of politeness toward the trainee. Depending on Hassan's level of compliance, his output is dynamically filtered to include a variety of such signals of politeness (*My friend*, *Sir*, etc.) or rudeness (*Listen to me*, *Look*, *You don't understand*, etc.). The style generator is based on techniques described in [12].

5. Preliminary Evaluation

We conducted a preliminary analysis of the improvements made to Hassan over previous architecture as described in [2]. A set of subjects ran the scenario, and filled out post-session surveys rating the experience on a scale of 1 to 7. The results of those ratings were compared to ratings from the previous Hassan system. In the three key user-satisfaction measures the system showed improvement. For answer to the question "How would you rate your performance in questioning Hassan?" the improvement was statistically significant ($p<0.01$ on student's T-test). For the questions, "How satisfied were you with your interview with Hassan" and "How well do you think Hassan understood your speech," the average results were higher but the results were not statistically significant. We have started conducting larger evaluations with more specific questions to determine the source and magnitude of the improvements. We also plan to test the authorability, by having non-experts use our tools to develop new characters.

6. References

[1] A. Leuski, R. Patel, D. Traum, and B. Kennedy, "Building effective question answering characters," in *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, 2006, pp. 18–27.

[2] D. Traum, A. Roque, A. Leuski, P. Georgiou, J. Gerten, B. Martinovski, S. Narayanan, S. Robinson, and A. Vaswani, "Hassan: A virtual human for tactical questioning," in *The 8th SIGdial Workshop on Discourse and Dialogue*, 2007.

[3] J. Rickel, S. Marsella, J. Gratch, R. Hill, D. Traum, and W. Swartout, "Toward a new generation of virtual humans for interactive experiences," *IEEE Intelligent Systems*, vol. 17, pp. 32–38, 2002.

[4] D. Traum, W. Swartout, J. Gratch, and S. Marsella, "A virtual human dialogue model for non-team interaction," in *Recent Trends in Discourse and Dialogue*, L. Dybkjaer and W. Minker, Eds. Springer, 2008.

[5] A. Roque and D. Traum, "A model of compliance and emotion for potentially adversarial dialogue agents," in *The 8th SIGdial Workshop on Discourse and Dialogue*, 2007.

[6] D. Traum and J. Allen, "Discourse obligations in dialogue processing," in *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 1994, pp. 1–8.

[7] D. Traum and S. Larsson, "The information state approach to dialogue management," in *Current and New Directions in Discourse and Dialogue*. Kluwer, 2003.

[8] J. Barnett, M. Bodell, D. Burnett, J. Carter, and R. Hosn, "State Chart XML (SCXML): State machine notation for control abstraction," 2007.

[9] F. Kronlid and T. Lager, "Implementing the information-state update approach to dialogue management in a slightly extended scxml," in *Proceedings of the workshop on the Semantics and Pragmatics of Dialogues*, 2007.

[10] A. Roque and D. Traum, "Using degrees of grounding for dialogue management," 2008, submitted.

[11] C. Müller, "Resolving it, this, and that in unrestricted multi-party dialog," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, June 2007, pp. 816–823.

[12] David Devault and David Traum and Ron Artstein, "Making grammar-based generation easier to deploy in dialogue systems," 2008, submitted.