NPCEditor

- Overview
- Users
  - Launching NPCEditor
  - Making a Character Speak
  - Reviewing and Editing Character Dialog
  - Talking to a Character Inside NPCEditor
  - Building a new Plist
- Developers
  - plist Format
    - High Level Format
    - categories
    - questions
    - answers
    - map
- Message API
- Known Issues
- FAQ

Overview

NPCEditor controls the spoken behavior of the characters in the Toolkit, as well as the structure and logic of the interaction through its dialog manager. It contains a list of user "**Questions**" and character "**Answers**", and the links between them. NPCEditor uses a statistical text classifier to determine the best character response to novel user input, allowing users to converse with the characters with reduced authoring effort.

Quick facts:

- Location: /core/NPCEditor
- Language: Java/Groovy
- Distribution: Binary
- Platform(s): Multi-platform
- Main paper: NPCEditor: A Tool for Building Question-Answering Characters

Users

Launching NPCEditor

In the launcher, click 'Launch' in the NPCEditor row under the 'Agents' group. It will also run by default when you launch 'Run Checked'

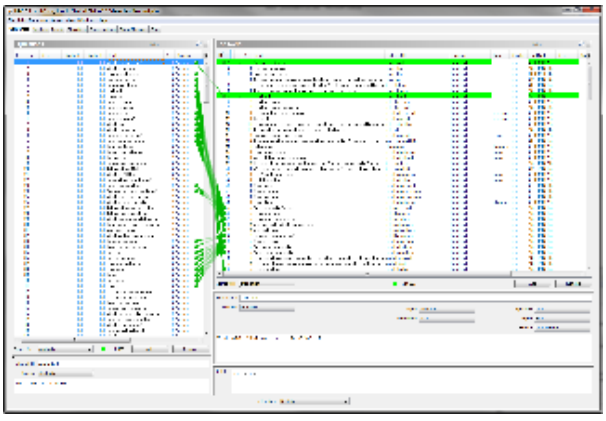
Making a Character Speak

The quickest way to make a character speak a line is to double click an entry in the right side of the **Utterances Tab**, under 'Answers'. The character should speak the line you've selected, with lip sync and gestures provided by NonVerbal Behavior Generator.

Reviewing and Editing Character Dialog

All of Brad's, or any character's, lines of dialog are stored in a plist generated by NPCEditor. NPCEditor uses a tabbed layout, with the two main tabs for users being the Utterances Tab and the **Chat Tab**. Dialog is found in the **Utterances Tab**.

Utterances Tab



The left hand side of the **Utterances Tab** lists the Questions, while the right is dedicated to the Answers. Questions are examples of user utterances that the system expects to receive. This can include questions, "What is your name?", or statements, "Tell me about yourself." When NPCEditor receives user input, whether typed or spoken through AcquireSpeech, it uses a statistical classifier to compare the input to known Questions. If there is an exact match, NPCEditor automatically responds with the highest scoring Answer. If there is not an exact match, the classifier determines the closest line in the Questions tab and responds with the highest scored linked Answer. These scores are determined by the links between Questions and Answers.

Character dialog is listed as Answers on the right of the tab. Each of a character's lines of dialog has its own entry.

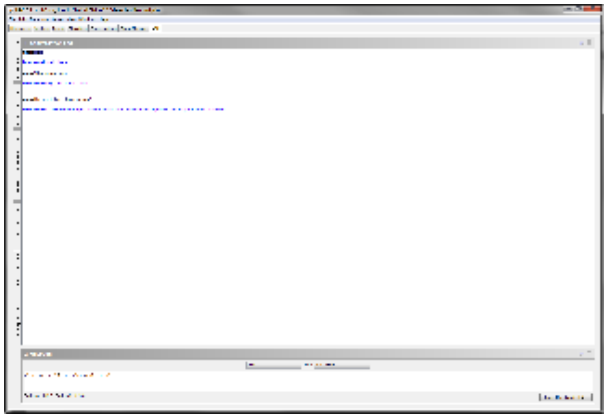
An Answer consists of a number of fields.:

- **External ID** is the file name of the sound file to be played back with a line of dialog.
- **Text** is the transcript of the line.
- **Domain** is used to control different dialog phases.
- **Speaker** and **Addressee** designate which character speaks this line and who is listening, whether another character of the user.
- **Type** is used to denote specific types of dialog, such as control lines, for example when Brad cannot understand the user's question.
- **Topic** and **Sound** are used to sort the dialog.

Each of these fields and the values within can be edited in the **Settings Tab**.

Talking to a Character Inside NPCEditor

Chat Tab



The Chat Tab allows you to see a log of the latest discussion with an NPCEditor controlled character, and interact with a character independent of other tools. You do not need a renderer or any other components. The body of the tab is by default a maximized chat interface. Users may enter what they wish to say to a character at the bottom, and the classifier will select the best response and print it to the **Conversation Log**.

For example, as you speak to Brad with your microphone, you will see your speech and Brad's responses populate the log. If you wish to type an answer, you may do so here by selecting the user to speak as, typing your utterance, then clicking "Enter Question" in the bottom right. If Unity or another engine is running, Brad should speak and respond there, as well as in his window. Even without any other components running, you can test text interaction with a character through this interface.

Along the left hand side are other tools for seeing how the classifier within NPCEditor parses user input, analyzes it, and selects the best response.

Building a new Plist

There is no set way in how to author a character, but the general approach is:

1. Select a certain domain or a small amount of domains that your character should be able to talk about. For the Brad example character, this is the toolkit, the toolkit modules, ICT, USC and Brad himself.
2. Within these domains, identify the topics that are of interest, or that you anticipate your users will want to discuss. For instance, Brad will explain what the toolkit is, how you can use it, which modules are part of it, and where you can find more information.
3. For each topic, identify the statements you want your character to make. You can add these on the right hand side, in the Utterances tab. Try to create statements that are stand-alone, i.e. not referring to any words the user might have used in the question (also see guidelines below).
4. For each statement, define various questions for which the statement can be an answer. These questions can be defined in the left hand side in the Utterances tab. Link all appropriate questions to the statement by selecting the statement and all associated questions, and press CTRL + 6. You should now have sets of questions that link to exactly one statement.
5. For each statement, create a variation and link that to all existing questions for that statement. You now have a variety of topics that the user can discuss in several ways, and for which there are 2 variations for each answer. Be sure to save the plist (or turn on autosaving in Edit -> Properties).
6. For each answer statement (right side), ensure you select the correct domain 'Brad Smith' and correct speaker 'Brad'. Also be sure to save the plist.
7. Before interacting with the character, you should train the NPCEditor. You can do this in the Classifiers tab. Select the character from the Addressee column. If you have one character, there should be only one row. If there are more rows, you have defined multiple addressees. This is likely do the default Anybody addressee. In the Utterances tab, on the right hand, you should select all rows and make the addressee either your character name or Anybody (with only one character, the addressee does not matter that much). After you have made sure there is only one row in the Classifiers tab, select that row, check the 'Test on training data' box at the bottom and click Start Training. You don't have to train the NPCEditor any time you make a change, but if you make substantial changes, and the character is not reacting the way you expected it to, you should try retraining it.
8. You are now ready to test out the character for yourself. Just run the entire scenario (including render and Text To Speech, etc.), and start interacting with the character. You will find that some questions are not answered correctly, that some answer might not sound as well as you anticipated, and that you are thinking of both more answers and questions. You can make all the changes you want on the fly in the NPCEditor, without the need to restart it.
9. If you are happy with your interactions, you can start having others interacting with it. This will show you which areas need more work. It will also result in much data, especially in the form of questions, which you can use as new input for the NPCEditor. You could use the provided Logger to collect all messages (which include the messages from the speech client to the NPCEditor, and extract all questions asked into a separate text file (you would probably have to write a script for that). You can then import this text file into the NPCEditor, after which you can create new answers for these questions, or link questions to existing answers.
10. With this iterative process, the quality of your character should improve over time. Let us know how you are doing!

Author generic, stand alone answers

The less specific you make your answers, the more appropriate they are in different situations. For example, if you ask Brad "Did you go to USC?", he can answers "Yes, I went to USC.". This is a perfectly fine and appropriate response for a yes-no question, but not for a related question, like "Where did you go to school?" As an author, you can either create an additional response, like "For college, I went to USC", or you can link both questions to a more generic answer like "I went to USC."

In general there is a balance you need to keep between making answers interesting and having them work in many different contexts. It can be very compelling when a virtual character responds exactly to what you say, especially by repeating part of the question (i.e. "Yes, I did go to USC! You too?"), but having an inappropriate response completely breaks the illusion. The best strategy seems to stick to generic and broad answers most of the time, while anticipating some very specific answers only once in a while. The generic answers should be stand alone statements that sound good in relative isolation and relate broadly to the topic inquired about.

Provide multiple answers for each question

The NPCEditor usually tries to not say the exact same line twice in a row, but it can only do that when an appropriate alternative is available. You should therefore make sure that you explicitly provide several paraphrases for a statement. Otherwise, the character can either repeat the exact same line (which is unrealistic), or it can select a different response which is statistically close to the answer your provided, but semantically very different.

Don't have your character ask questions, unless you know you will get a very specific answer back

The NPCEditor version provided with the toolkit treats any input (usually a question) without any context. That means a character is unaware of what has been said in the conversation thus far, nor does it know what it has just said (apart from a simple 'do not repeat yourself is possible' rule). As a result, a character cannot ask a yes / no question, because it has no way to relate the answer ("yes", "no" or "maybe") to the question just asked. To the character, the answer is just isolated input; it cannot discern to which of potentially many yes/no questions it is an answer to. Of course, if you have only one yes/no question in your domain, you are safe to assume any yes/no related input is an answer to that particular question. In addition, if you know you will get a very specific answer back, you can respond appropriately to that specific answer. For instance, apart from a couple of rhetorical questions, Brad only asks the user how he or she is doing. This works in most cases, because it can appropriately respond to answers like "I am fine" and "Not so well". However, when a user just says "OK", we need to make a choice to either treat that as a response to how that person is doing, or as an unrelated acknowledgement to any of Brad's statements.

There are ways around this particular problem that we are experimenting with within ICT. If you are interested, do not hesitate to contact us.

When using Text To Speech, check how your answers sound

You can do this by double clicking on the answer in the right hand panel of the NPCEditor, or by selecting the row and clicking Send, at bottom right. You might notice that a particular TTS engine does not pronounce a certain word you expected it. If that's the case, you can either use a different word, or be more phonetic in your spelling. For instance, the Brad character has "ICT" spelled as "I C T" in order to force the TTS engine to pronounce each individual letter, rather than trying to create a sound as if it was a word.

For each set of answers, gather around 30 questions

Initially you can just have a couple of questions for each set of answers, but user testing will result in many different paraphrases for these questions. Collecting all these paraphrases and linking them to the right set of answers ensures that your character will get more and more robust.

Developers

NPCEditor provides a way to modify your dialogue manager using a groovy script. If you look under the Dialog Manager tab in NPCEditor, you can point to a script of your own and use it.

Groovy is a scripting language which uses java like syntax. You can find more about it at http://groovy.codehaus.org/

See these slides to get started: 2013-03.NPCEditor-secrets.key.pdf

plist Format

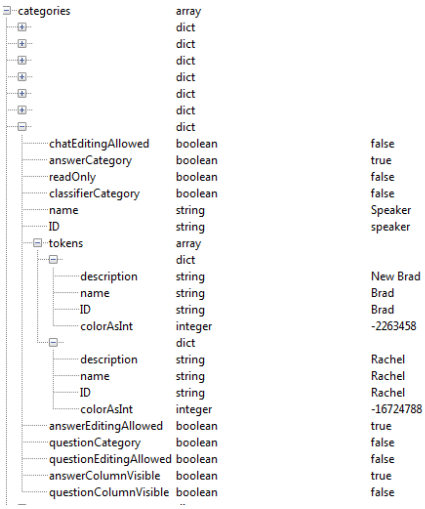
The NPCEditor uses a .plist file format for loading.

High Level Format

Root	dict
question_format	string
recordingQuestions	boolean
answer_format	string
answers	array
questions	array
speakers	array
searcherSessions	array
trainingOnTestQuestions	boolean
version	integer
keepingLinkEstimatesUpdated	boolean
map	array
categories	array
dialog_manager	dict
evaluationFrameworkProvider	dict
loggingConversations	boolean

categories

The character names and ids can be found in this section

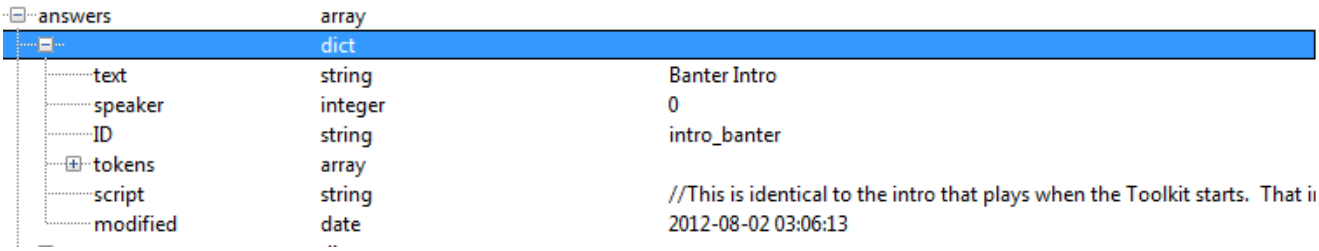


questions

Lists all the user questions

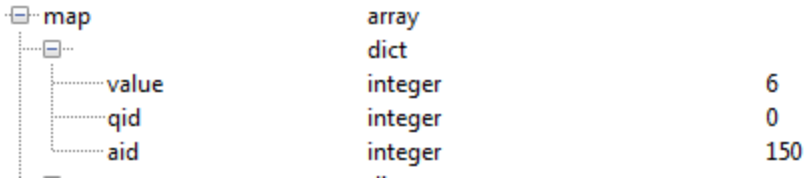
answers

Lists all the system answers that the virtual humans respond with. Speaker id is the index of categories speaker tokens array



map

Contains an array of dicts that maps a question from the questions section of the plist to an answer from the answer section of the plist using array indices.



Message API

Input:

- vrSpeech

Output:

- vrExpress

Known Issues

- Heavy memory usage under Windows XP 64 bit.

FAQ

See the NPCEditor section in the Main FAQ.