The Nudge Support Toolset

Michael A. Gohde

December 6, 2016

1 Overview

The task of writing stories for Nudge is inherently difficult. The Nudge Support Toolset exists to alleviate some of this difficulty by providing a large set of utilities designed to compile, validate, and install stories.

This document exists to detail the function and operation of each utility in the Nudge Support Toolset.

2 story2xml.py

story2xml exists to translate well structured, human-readable stories into an intermediate XML format for use by all of the other tools in the set.

2.1 Input file format

```
Title: <Insert story title here>

First story node title:
  <Insert story node text here>

  Responses:
      Response 1 text -> prb1% to dst1, prbN% to dstN
      Response N text -> prb3% to dst3
```

Each story2xml input file must follow a common set of conventions:

1. The story's title must be written on its own line and prefixed with "Title:"

2. Each story "block" or "decision" (hereafter referred to as a "node") must start with a title followed by a colon.

3. The text content in each node (ie. the story text) must not start with a word followed by a colon.

4. All possible user actions are defined in a block prefixed by the keyword "Responses:". This block must be indented more than all of the other text in its node.

2.2 A complete input file example

```
Title: Example story

D1_0:
    You are confronted with a serious question:
    To cheese it or not to cheese it?

    Responses:
        Cheese it! -> 50% to D2_0, 50% to D2_1
        Don't cheese it! -> 100% to D2_2

D2_0:
    Note: This is a comment.
    Note: All comment text is discarded.

    You were able to cheese it!

    Responses:
        Proceed -> 100% to D3_0

D2_1:
    Comment: This is also a comment.
    You were unsuccessful at cheesing it!

    Responses:
        Proceed -> 100% to D3_0

D2_2:
    You proceed not to cheese it.

    Responses:
        Proceed -> 100% to D3_0

D3_0:
    Regardless of whether you cheesed it,
    something happened.

    Responses:
        End -> 100% to END
```

2.3 Running story2xml

story2xml accepts one argument on the command line: the name of a text file containing a properly formatted story. Once run, story2xml will print an XML-formatted story to stdout, so it may be useful to redirect its output to a different file.

Example usage:

```
story2xml mystory.txt >mystory.xml
```

2.4 Error messages

When compiling a story, story2xml attempts to perform a few tests to ensure that the provided source file is logically sound. If an error is encountered, story2xml will print a message and information that can be used to locate and correct the error.

List of error messages:

Error: for response (response text) on line (line number), destination probabilities exceed 100%

Error: for response (response text) on line (line number), destination probabilities sum to a value below 100%

Warning: for line (line number), unknown token (string)