Dialogue Framework

*RSVP Flavored AIML Specifications*

*Comprehensive documentation of how to write and extend AIML files.*

Version: v1.0

Last revised: 2014-12-11

Table of Contents

[1. AIML Basics 2](#_Toc414526258)

[1.1 What is AIML 2](#_Toc414526259)

[1.2 Elements of AIML 2](#_Toc414526260)

1. AIML Basics
   1. What is AIML

AIML, or Artificial Intelligence Markup Language, is and XML dialect for creating natural language software agents. In another word, it let you define the logic for a chatterbot to generate responses based on certain user inputs. AIML is used to write the world famous chatterbot ‘A.L.I.C.E.’ and also adopted in many of today’s popular chatterbots.

* 1. Elements of AIML

AIML contains several elements, which are XML nodes. The unit in which the logic of generating response to a certain input is contained inside a <category>. And inside each <category> there are two required nodes <pattern> and <template>, as well as one optional nodes <that>. They are described in further detail below.

<category>

<pattern>Hello</pattern>

<that>Hi</that>

<template>How are you</template>

</category>

When this category is loaded, the bot will respond ‘How are you’ when the received user input is ‘Hello’ and the bot’s last response was ‘Hi’.

* 1. Patterns

A pattern is a string of characters intended to match one or more user inputs. A literal pattern can be something like ‘Hello’ in the above case that only the input ‘Hello’ will be matched by this category. Standard AIML also supports wildcard matching. For example, one can use ‘\*’ to match anything. However, remember that AIML is targeting English so words are expected to be separated by spaces, which doesn’t fit Chinese well.

As a result, we extended standard AIML patterns and allows the usage of grammar patterns. To use grammar patterns, one need to solely use a <grammar> tag inside the <pattern> tag and place the grammar term that he/she is trying to match inside the tag. For example:

<category>

<pattern>

<grammar>basic.greeting</grammar>

</pattern>

<that>Hi</that>

<template>How are you</template>

</category>

Instead of matching the literal version of user input, now this category tries to match a public grammar term ‘basic.greeting’ which was already defined in the chatterbot’s grammar files, where ‘basic’ is the namespace and ‘greeting’ is the term. For instance, the grammar file could look like this:

Namespace basic

hi := strings{Hello}

public greeting : hi

Thus, if the user input was ‘Hello’, the grammar parser will parse the input beforehand and output the term ‘basic.greeting’, therefore it will be captured by the category listed above,

* Templates