Chatbot Creation Options

Abstract

Chapter 2 of Library Technology Reports *(vol. 49, no. 8), "Streamlining Information Services Using Chatbots," introduces AIML and ChatScript, the two most viable languages for creating a chatbot. While their basic structure and syntax are markedly different, either may be used effectively, and both offer their own advantages.*

There are a number of coding options available to use in creating your own bot. The markup or scripting language you choose will depend on your skill and experience, the amount of time you have available, and the functionality you're trying to create. At present, the best choices are AIML (Program Z or Program O) and ChatScript. We'll examine each in turn.

AIML (Artificial Intelligence Markup Language)

AIML is the starting place for many who are interested in chatbots or natural language processing. AIML was created in 1995 by Dr. Richard Wallace and is the basis for numerous chatbots, including the original Emma the Catbot, the University of Nebraska's Pixel, Adeena Mignona's Zoe, and Steve Worswick's Mitsuku.

AIML's great virtue is its simplicity; it's easy to learn and to implement. AIML is an XML dialect, so if you're familiar with XML or HTML, you'll be able to learn AIML quickly. You can write AIML using Notepad, WordPad, or a spreadsheet-style AIML editor like Simple AIML Editor from RIOT Software. AIML is based on pattern matching. Essentially, the data making up an AIML bot's "brain" take the form of a very large decision tree. User input is first preprocessed and then matched in order against the nodes of the tree. When input finds a match, the bot will execute an action, such as responding or opening a web page.

Simple AIML Editor
http://riotsw.com/sae.html

AIML does have some drawbacks, however. AIML's pattern matching is relatively weak, which means the content you create has the potential to match a range of input and return incorrect or meaningless responses. While authoring content is easy, a large amount of content is needed to create a convincing bot, somewhere in the range of 60,000+ categories. Each question or concept in the bot's knowledge base requires multiple categories to match permutations of the question and to ensure a correct response. For instance, there are many ways to ask, "What time does the library open?":

"When do you open?"
"When are you opening today?"
"What time do you open?"
"Will you be open today?"

You can easily add to this list.

A category is required to match each variation:

```
<Code Sample>
<category>
<pattern>WHAT TIME DOES THE
   LIBRARY OPEN</pattern>
<template><srai>HOURS</srai>
   </template>
</category>
```

```
<category>
<pattern>WHEN DO YOU OPEN
   </pattern>
<template><srai>HOURS</srai>
   </template>
</category>

<category>
<pattern>WHEN ARE YOU OPENING
   TODAY</pattern>
<template><srai>HOURS</srai>
   </template>
</category>

<category>
<pattern>WHAT TIME DO YOU OPEN
   </pattern>
<template><srai>HOURS</srai>
   </template>
</category>

<category>
<pattern>WILL YOU BE OPEN TODAY
   </pattern>
<template><srai>HOURS</srai>
   </template>
</category>
<End Code Sample>
```

In order to understand how an input will match or fail, you need to be familiar with all of the categories dealing with each question or concept in the bot's knowledge base. One cannot look at an individual AIML category and know what input it will match. Writing code to distinguish between fine shades of meaning can be tricky, and the time required to maintain and debug an AIML knowledge base can be considerable. Before you become too discouraged, keep in mind that AIML has been used to create successful library bots. There are ways to reduce development and maintenance time, such as by modifying an existing AIML set, infoTabby or A.L.I.C.E., and by writing your code to be as efficient as possible. We'll cover the details of writing efficient AIML in chapters 3 and 4.

Over the years, a variety of AIML interpreters, or engines, have emerged using Java, Ruby, Python, C++, C#, Pascal, and others. For our purposes, the two best choices for creating a library bot are AIML Program Z and AIML Program O.

AIML Program Z

AIML Program Z is the proprietary language of Pandorabots, one of the oldest and largest chatbot hosting services in the world. As of February 2012, Pandorabots' free service was home to more than 166,000 botmasters and 201,000 AIML chatbots.[1] If you would like to start with AIML and would like to avoid the technical issues involved with setting up your own chatbot server, Pandorabots is a fine choice. Free and paid hosting are both available. The free server is a good place to start and experiment. The paid server has great reliability. Either will make setting up your bot and adding it to a web page easy. The Pandorabots site has excellent documentation and support, including an extensive series of YouTube tutorials covering AIML basics, AIML predicates, ReversedAIML, adding SitePal avatars, AIML targeting, bot properties, and other topics. Pandorabots' excellent account interface gives access to conversation logs, including a time stamp and the user's IP address, the bot's AIML set, a training interface, Pandorawriter (which converts dialogue into AIML), and the HTML files used to embed the bot into web pages (see figure 2.1). Editing the AIML set, uploading or downloading files, and reviewing conversations are all made quite easy.

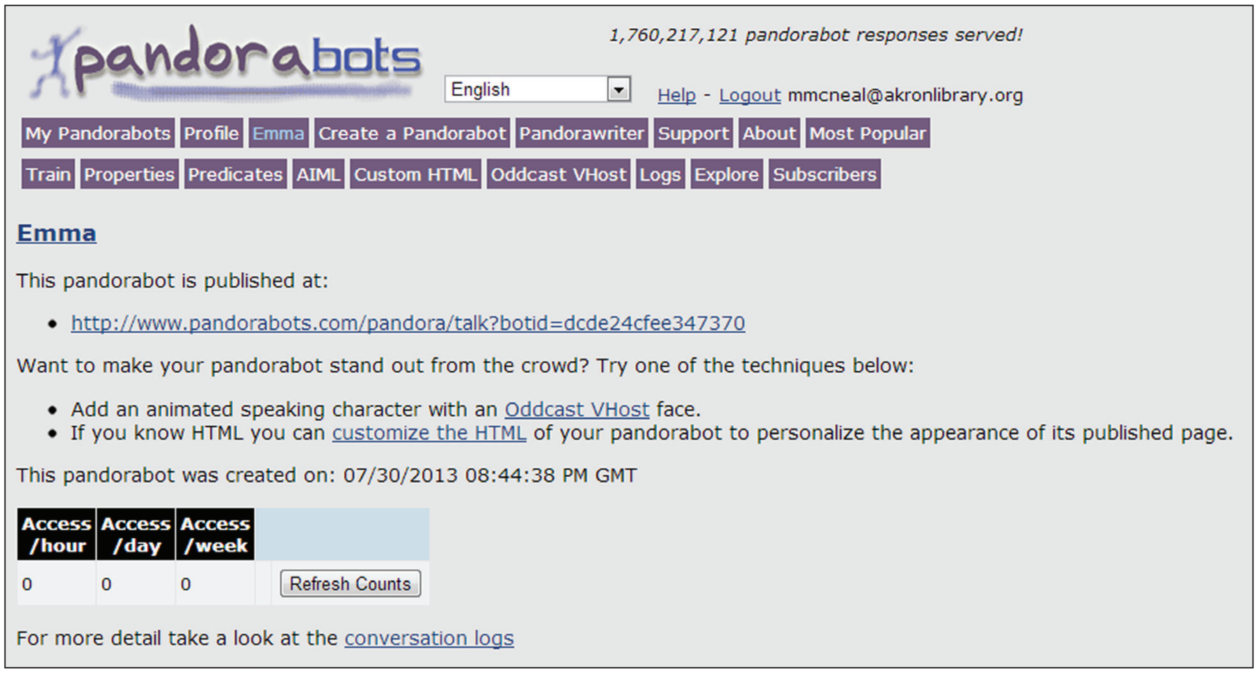
Pandorabots
www.pandorabots.com

You can create multiple bots, so it's possible to stage or test your changes before making them live to the public. You can start with no AIML content or choose among several pre-authored AIML sets: Standard AIML (2001), A.L.I.C.E. (2002), Annotated A.L.I.C.E. AIML (2003), Christian Drossmann's standalone German AIML, or Sandro Pons's Italian AIML set. Pandorabots works closely with SitePal, so adding avatars and text-to-speech is simple if you wish to give your bot a face and voice.

In April 2012, Pandorabots released CallMom, a mobile, voice-activated virtual assistant app for Android that can be configured to use any existing Pandorabot. You can download CallMom through Google Play at no charge. CallMom is open source and supports a number of additional AIML tags. We will discuss CallMom and mobile apps in a later chapter.

CallMom on Google Play
https://play.google.com/store/apps/details?id=com.pandorabots.callmom

Because of AIML's simplicity, the small financial investment needed to implement it, and the availability of the free hosting provided by Pandorabots, we will be using AIML Program Z for the examples in later chapters of this report.


1,760,217,121 pandorabot responses served!

English
Help - Logout mmcneal@akronlibrary.org
My Pandorabots | Profile | Emma | Create a Pandorabot | Pandorawriter | Support | About | Most Popular
Train | Properties | Predicates | AIML | Custom HTML | Oddcast VHost | Logs | Explore | Subscribers

Emma

This pandorabot is published at:

- http://www.pandorabots.com/pandora/talk?botid=dcde24cfee347370

Want to make your pandorabot stand out from the crowd? Try one of the techniques below:

- Add an animated speaking character with an Oddcast VHost face.
- If you know HTML you can customize the HTML of your pandorabot to personalize the appearance of its published page.

This pandorabot was created on: 07/30/2013 08:44:38 PM GMT

Access /hour	Access /day	Access /week	
0	0	0	Refresh Counts

For more detail take a look at the conversation logs

Figure 2.1
An example of some of the documentation and support offered by Pandorabots. Copyright 2013 Pandorabots. Portions reproduced with permission.

AIML Program O

Program O is an open-source, web-based application written in PHP using a MySQL database. You can access and download Program O from the Program O website. Older versions of Program O lacked some of the functionality of Program Z, but this gap has been greatly reduced with the release of version 2.0. Program O has the same advantages and disadvantages as program Z; you will have to set up the engine on your own server and will need to have PHP and MySQL running on that server. A good knowledge of PHP and MySQL is helpful, but not entirely necessary. As you will be installing Program O on your own webspace, it is more complicated than using Pandorabots, but can offer greater flexibility for experienced webmasters. The University of Nebraska's Pixel is an excellent example of a library bot written in Program O; Morti is another well-written Program O chatbot.

Program O website
http://blog.program-o.com

Pixel
http://pixel.unl.edu

Morti
www.geekcavecreations.com/Morti

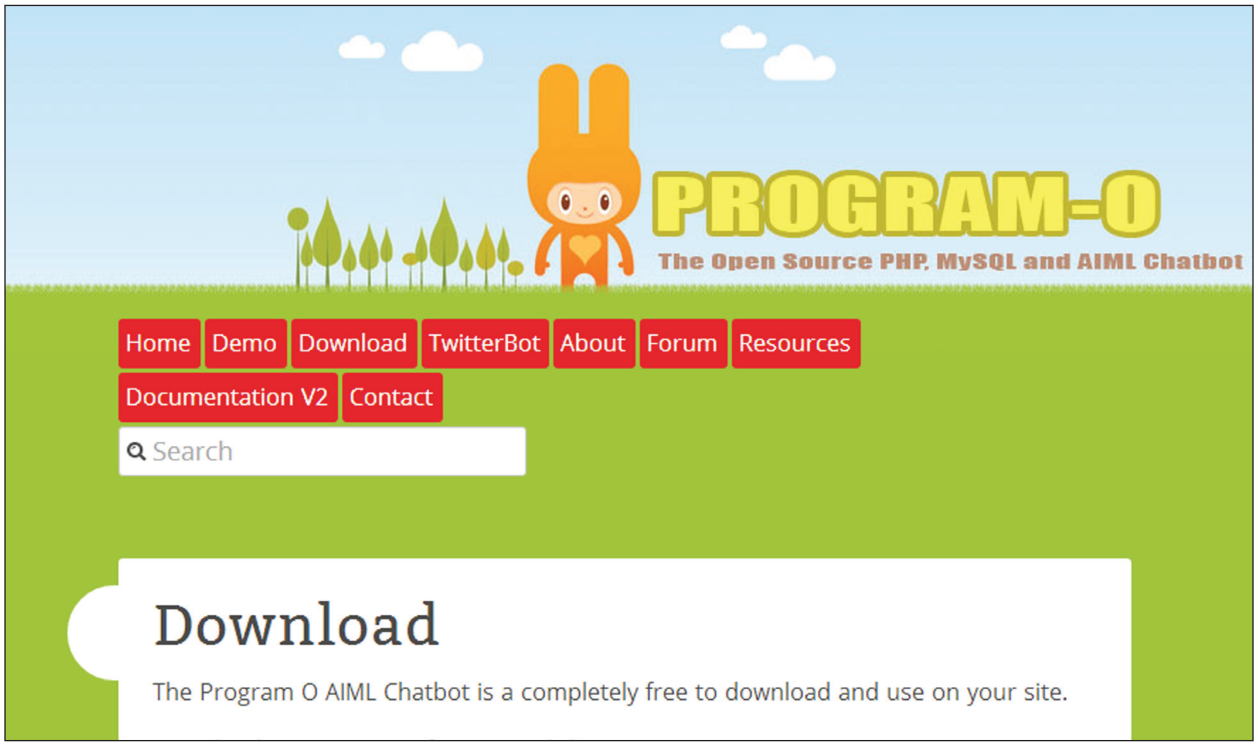
The Program O website contains demos, downloads, an installation guide, and support, including a FAQs page, forum, and contact information.

Program AB/AIML 2.0

Program AB is a compiler supporting AIML 2.0 and is written in Java. AIML 2.0 was released in January 2013 by the ALICE A.I. Foundation[2] and represents an attempt to address some of the limitations of earlier AIML versions while maintaining the simplicity of the language. Some of the new features are

- Wildcards that match 0 or more words
- Highest priority matching to assign top matching priority to selected words
  - AIML sets to match input with sets of words and phrases (similar to ChatScript concepts, see below)
  - AIML Maps to map set elements to members of other sets
  - looping
  - local variables

It is also possible to define and use custom tags. The full AIML 2.0 working draft and specifications are available online. At this time, implementation of AIML 2.0 is limited to Program AB and the CallMom Basic app. A new Pandorabots server that will support all



Download

The Program O AIML Chatbot is a completely free to download and use on your site.

Figure 2.2
The Program O download page. Copyright 2013 Elizabeth Perreau and The Program O Project. Used with permission.

the features of AIML 2.0 is under construction.

AIML 2.0 Working Draft
https://docs.google.com/document/d/1wNT25hJRyupcG51aO89UcQEiG-HkXRXusukADpFnDs4/pub

AIML Summary

AIML Pros

- Simple
- Easy to learn
- Easy to implement
- Pre-authored AIML sets are available (including the infoTabby AIML set designed for public and academic library use). For more information and to download the AIML sets visit the infoTabby website.

infoTabby
www.infoTabby.org

AIML Cons

- Relatively weak pattern matching
- Can be time-consuming and difficult to maintain
- A large number of categories are needed to create a robust bot.
- While AIML 2.0 addresses these issues, options for implementation are currently limited but growing.

ChatScript

ChatScript is a scripting language/chatbot engine developed by Bruce Wilcox and released as open source in 2010. In only a few years it has been very successful. Bruce's bot "Suzette" won the 2009 Chatterbot Challenge as "best new bot," and went on to win the 2010 Loebner prize. Bruce's bot "Rosette" won the 2011 Loebner prize, and "Angela" placed second in 2012. ChatScript has been used commercially in ESL bots by SpeakGlobal (Japan) and as the AI component of the popular "Angela loves Tom" app by Outfit7.[3]

You can download ChatScript from SourceForge. The zip file contains the ChatScript engine and excellent documentation, including a user manual and tutorial.

ChatScript download on SourceForge
http://sourceforge.net/projects/chatscript

ChatScript is powerful and efficient and has much stronger pattern matching than AIML. To give you an idea of ChatScript's power, one of the AIML files in the original infoTabby set contains seventy-two categories covering over six hundred lines of code. This file has been recreated in a single ChatScript topic with seventeen lines of code. ChatScript is organized into rules, which are gathered under topics. Unlike AIML, which finds the best pattern match for an input, ChatScript first finds the best topic match, then executes a rule contained in that topic. This might sound a bit confusing, but it really isn't. Let's take a brief look at a ChatScript topic and follow how it matches input. For instance, a library bot should be able to answer questions about e-books. Here's how we can start to program this in ChatScript:

```
concept: ~ebook NOUN_
   SINGULAR(ebook epub kindle_book
   nook_book Overdrive)
topic: ~policies_ebook keep repeat
   (~ebook)
t: We have many eBooks for you
   to download and enjoy. Follow
   this link for more information:
   http://www.ourlibrary.org/
   overdrive.html

u: (need * card * checkout) You
   need to have a library card to
   check out eBooks. Do you have a
   card?
a: (~yes) Great! Follow this link
   to our eBooks: http://www.our
   library.org/overdrive.html

a: (~no) I can help you get a card
   online right now. Just follow
   this link and complete the
   form: http://www.ourlibrary.
   org/selfreg.htm

u: (~renew * ~ebook) I'm sorry,
   eBooks may not be renewed.
u: (~help * ~ebook) Here's our
   eBook help page: http://www.
   ourlibrary/ebookFAQ.html

u: (~ebook) gambit(~)
```

Some users are concise and might enter a single word only, for instance e-books. This input would match the topic, and the bot would respond with the line beginning with t: (the "gambit").

```
t: We have many eBooks for you
   to download and enjoy. Follow
   this link for more information:
   http://www.ourlibrary.org/
   overdrive.html
```

This line would also match user input such as

- Do you have e-books?
- Can I get Nook books from you?
- I'm looking for epub books.

If our user asks, "Do I need a card to check out an e-book?" the input would first match the topic, then trigger the rule:

```
u: (need * card * checkout) You
   need to have a library card to
   check out eBooks. Do you have a
   card?
a: (~yes) Great! Follow this link
   to our eBooks: http://www.
   ourlibrary.org/overdrive.html

a: (~no) I can help you get a card
   online right now. Just follow
   this link and complete the
   form: http://www.ourlibrary.
   org/selfreg.htm

a: (*) If you're not sure, it
   would be best for you to speak
   with one of our human staff.
   Please call them during regular
   business hours at 555-555-1234.
```

Note that the bot answers the user's question, then gives further assistance by asking if the user has a card. Appropriate responses are then given by the bot based on the user's response. Let's look at a few more features.

Concepts

ChatScript comes preloaded with a large database of concepts; you can create additional concepts easily. Concepts are sets of synonyms. Once a concept has been created, it will match user input containing words within the concept. In the preceding example there is a concept for e-books:

```
concept: ~ebook NOUN_
   SINGULAR(ebook epub kindle_book
   nook_book Overdrive)
```

This concept will match user input with any of the words ebook, epub, kindle-book, nook-book, Overdrive and will also match their plural forms. In AIML it would be necessary to write categories for each of these words. In ChatScript, we can create a single concept to cover all of them. Concepts can also be nested within other concepts, as in this example:

```
concept: ~downloadable (~ebook
   ~Freegal ~audiobook ~podcast)
```

Canonical Forms

ChatScript can match both the original and canonical forms of a word if you use the canonical form in a pattern. For example, if you use the word book in a pattern, ChatScript will match either book or books in the user's input. Plural nouns canonize to their singular form, verbs canonize to their infinitive forms. Adjectives and adverbs canonize to their base forms.

WordNet Ontologies

WordNet is a lexical database of the English language created and maintained at the Cognitive Science Laboratory of Princeton University. It groups words into sets of synonyms—synsets—each expressing a distinct concept. WordNet gives short, general definitions and records the semantic relations between synsets. You can use WordNet ontologies in ChatScript by naming the word and meaning you want. This makes distinguishing between finer shades of meaning much easier.

WordNet
http://wordnet.princeton.edu

Wildcards

ChatScript uses an assortment of wildcards. The unrestricted wildcard * will match zero or more words. You can also specify a ranged wildcard. *~3 will match zero to three words. You can also set a specific length wildcard—*1, *2, etc.—which will match only that number of words. It's even possible to set backward wildcards—*-2, *-3—which will match only that number of words before the wildcard position.

Variables

ChatScript lets you use an assortment of variables, match variables, user variables, and system variables. Match variables will temporarily retain input, as in this example:

```
s: (my name is _*) Nice to meet
   you, '_0.
```

If the input is My name is Kristina, the output would be Nice to meet you, Kristina. User variables are preceded by the dollar sign: $username. Match variables can be retained as user variables:

```
s: (my name is _*)
$username='_0
Nice to meet you, $username.
```

User variables will last indefinitely or until they are assigned a new value. System variables are predefined and start with %. One does not normally assign values to system variables. One example of their use:

```
?: (what * date) Today is %day,
   %monthname %date.
```

Facts and Tables

One of the most powerful features of ChatScript is the ability to create facts—subject, verb, object triples—and to organize them into tables. The data saved as facts and in tables may then be recalled using queries. For example, you can organize your fine information into a table:

```
table: ~fine_rates (^item
   ^dailyfine)
^createfact(^item finerate
   ^dailyfine)

DATA:
book        $0.10
cd          $0.10
dvd         $0.50
magazine    $0.10
videogame   $0.50
ipad        $5.00
kindle      $5.00
nook        $5.00
```

Rates for each kind of item may be recalled by using a query.

Implementation

Since ChatScript is both a scripting language and an engine, implementation can be somewhat more difficult than AIML, depending on your intended use. There are currently no hosting services available for ChatScript, although this may change in the near future. ChatScript is well suited for stand-alone applications, such as information kiosks or desktop help icons, or as the intelligent component of apps. In fact, ChatScript's ability to run without an external server is a decided advantage in these cases. Dave Morton has several ChatScript GUIs available to download on his website, Geek Cave Creations.

Geek Cave Creations
www.geekcavecreations.com

ChatScript Summary

ChatScript Pros

- Strong pattern matching
- Powerful, flexible, and efficient
- Excellent documentation and support available
- Proven performance in commercial use
- Best choice for certain applications as it may be used without an external server

ChatScript Cons

- More difficult to learn than AIML
- Programming experience helpful, but not necessary
- There are no hosting services analogous to SitePal, so while it is possible to create a bot with an avatar and text-to-speech capability, you will have to do it yourself
- More difficult to embed in a web page

Notes

1. "Welcome to Pandorabots," accessed October 2, 2013, www.pandorabots.com/botmaster/en/home.
2. "Program AB," Google Code website, accessed September 12, 2013, https://code.google.com/p/program-ab.
3. "ChatScript," SourceForge website, accessed October 2, 2013, http://sourceforge.net/projects/chatscript.