RiveScript

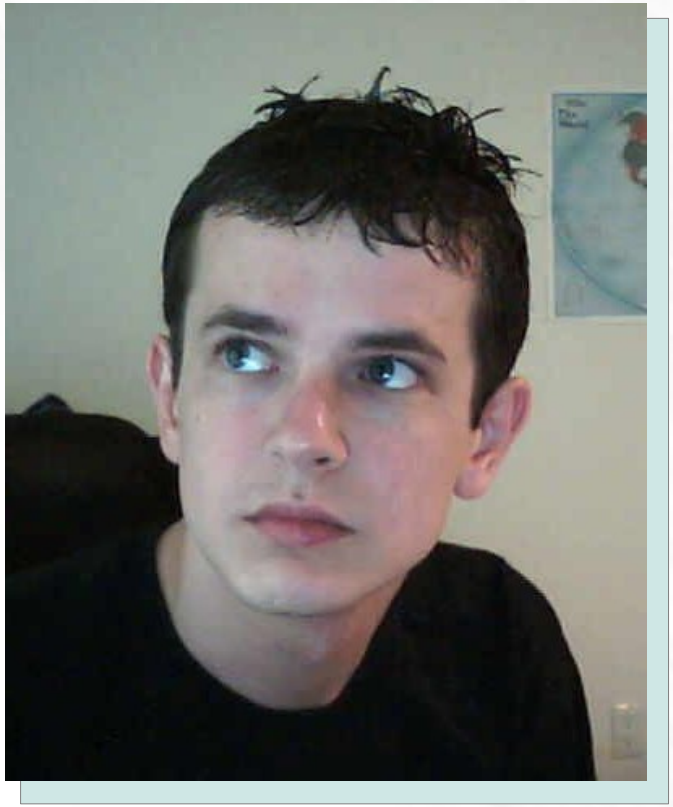
Chatbots 3.2 Conference

Presented by
Noah Petherbridge
noah@kirsle.net

Agenda

- Introduction & Background
- Reinventing AIML
- Chatbot::Alpha
- RiveScript Features
- RiveScript vs AIML
- Syntax Examples

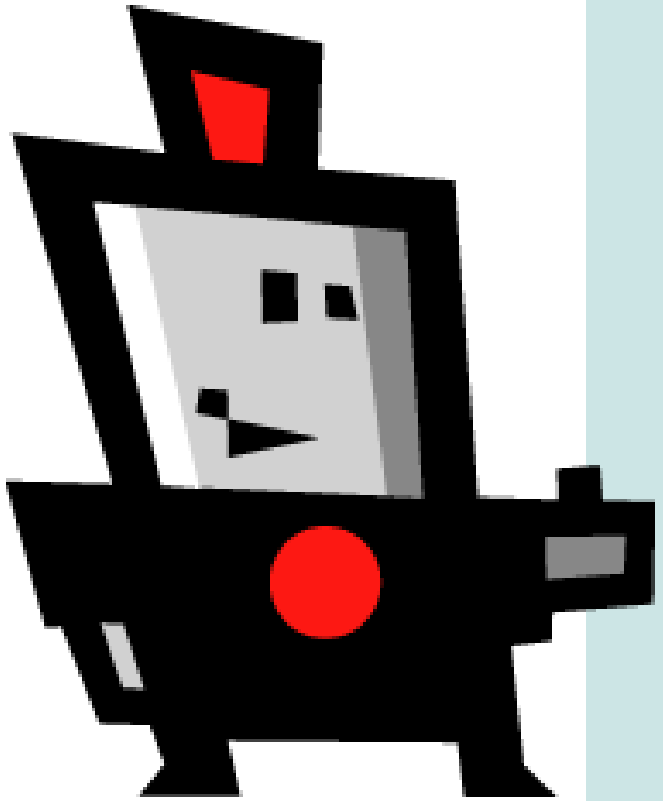
Who I Am



- Software developer
  (Perl, Java, HTML/JS...)
- Works at DreamHost
  (web hosting)
- Chatterbot programmer
  (as a hobby)

My Inspiration

- **SmarterChild** by ActiveBuddy was the first chatterbot I ever saw.

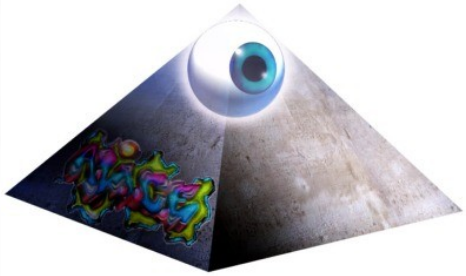


Chatterbot Experience

RunABot ca. 2001

- Uses a variant of AIML
- ALICE or Eliza as the base brain
- Free AIM bots, paid MSN bots
- Free bots were downgraded to Eliza whenever server was under heavy load
- I was 13—a paid subscription wasn't an option.

Chatterbot Experience



AliceBot Program D ca. 2002

- Free AIM and IRC bot engine
- Faster response times
- AIML 1.1
- But no regard for AIM rate limits or the AIM “warning” system.

Chatterbot Experience

- I wanted an MSN Messenger bot!
- Some of my friends only used MSN and I wanted them to see what my hobbies are.

Chatterbot Experience

- Found **WiredBots** (now defunct), ca. 2003
- Perl code templates for AIM and MSN bots
    - My first exposure to Perl
- Very minimal templates
    - No A.I. engine built in
- No AIML modules for Perl

Chatterbot Experience

- Taught myself Perl largely by editing and tinkering with the WiredBots template
- Started rewriting my Perl bots from scratch
- Started to miss AIML

Reinventing AIML

- Made various attempts at developing an A.I. engine I could use in Perl.
  - Many failed attempts to parse AIML code
    - Parsing XML in Perl isn't very easy!
  - Tried understanding Alicebot Program V
  - Failing at getting AIML to work for me, started developing my own alternatives.

Reinventing AIML

- Requirements for the new Perl AI engine:
  - Had to be easy to parse: no XML!
  - Had to be at least as easy to edit by hand as AIML is
  - Had to support most of AIML's features
    - But with RunABot's <topic> behavior :)
  - Ideally, should be easy to implement in other programming languages too

Chatbot::Alpha

- First iteration was named Chatbot::Alpha
- Basic syntax inspired by BuddyScript:

```
+ hello bot
- Hello human!
```

Chatbot::Alpha

- Inspired by BuddyScript, but **not** like it!
- It's a line-by-line, command-driven language.
- Commands are single symbols:

```
+ trigger     @ redirect (<sr/>)
- response    * conditions
```

- Easy to write, easy to parse!
- Feature set very close to AIML's

But we can do better than that!

Chatbot::Alpha

- Perl is a powerful language and great at processing text with its regular expressions
- Why settle for `<pattern>MY NAME IS *</pattern>` when we can have "`my (name|alias) is *`"?
- More replies in less space without tons of `<srai>`'s everywhere.

Introducing...

RiveScript

www.rivescript.com

RiveScript

- RiveScript is my second attempt after Chatbot::Alpha
- Fixes the limitations of Chatbot::Alpha
- Matches and surpasses the feature set of AIML 1.1
- Regular expressions made easy – less need for `<srai>`-style pattern aliases.

“RiveScript?”

- Brief history behind the name:
  - My first RunABot chatbot was named Chaos
  - Chaos moved to my WiredBots Perl bots
  - Created website “Chaos A.I. Technology,” later renamed to “AiChaos” (now defunct)
  - Rive – *v.* – *to rend or tear apart.*
  - With a little stretch of the imagination, “chaos” is similar to “rive”

RiveScript Features

- Simplified regular expression patterns
- Random responses a first-class feature (no need for <random>)
- Weighted random responses
- Full logic conditionals (==, !=, <, <=, >, >=)
- RunABot-style topics
- Support for inline Perl code
- Self-contained configuration

Self-Contained Configuration

- Bot variables, substitutions and “person substitutions” defined in RiveScript code instead of external XML files like Alicebots.

```
! bot name   = Aiden
! bot age    = 5
! bot master = Noah

! sub what's = what is
! sub who's  = who is
! sub :)     = smile
```

```
! person you = I
! person am  = are
! person I   = you
! person are = am
```

RiveScript vs. AIML

- RiveScript can do everything AIML can do.

```
<think><set name="name"><formal/></set></think>
<set name=<formal>>

<get name="name"/>          <uppercase>..</uppercase>
<get name>                  {uppercase}..{/uppercase}

<bot name="name"/>          <lowercase>..</lowercase>
<bot name>                  {lowercase}..{/lowercase}

<star index="1"/>           <that index="1"/>
<star1>                     <reply1>
```

- (Key: AIML, RiveScript)

RiveScript vs. AIML

- Not going to cover the similarities with AIML
- Going to show what RiveScript *looks like*
- Going to show areas where RiveScript excels
- Visit RiveScript.com for everything else

Trigger Examples

- **Atomic Triggers**
  The simplest example
- A simple trigger, a simple response.

```
+ hello bot
- Hello human!

+ how are you
- I am great, you?

+ aha
- Indeed.
```

Trigger Examples

- **Wildcards**
  Just like in AIML, but with more features.
- * matches anything
- # matches digits only
- _ matches letters only

```
+ say *
- Umm... “<star>”

+ my name is _
- Nice to meet you,\s
^ <formal>!

+ i am # years old
- A lot of people are\s
^ <star> years old.
```

(the ^ command continues the previous line)

Arrays for Triggers

- You can pre-define certain arrays for use in triggers.

```
! array colors = red blue green yellow

// use in parenthesis to match in <star> tags
+ what color is my (@colors) *
- Your <star2> is <star1>, silly!

// what color was Napoleon's white horse?
+ what color was * (@colors) *
- <star1>'s <star3> was <star2>!
```

Response Examples

- **Random Responses**
  Just add multiple – lines!
- Responses are chosen with equal weight

```
+ hello
- Hello there!
- Hey!
- Hi!
- Hey there!
```

Response Examples

- **Weighted Random Responses**
  Just add a {weight} tag!

```
+ hello bot
- Hello human!{weight=10}
- Hey there!{weight=4}
- Hi!
```

“Hello human!” has a 10/15 chance
“Hey there!” has a 4/15 chance
“Hi!” has a 1/15 chance

Priority Triggers

- Weights can also be applied to the triggers, to make high priority triggers.

```
+ * or something{weight=50}
- Or something. <@>
```

- Weights in triggers just define priority in comparison to other triggers. The numbers are arbitrary.

Conditional Examples

```
+ i am # years old
- <set age=<star>>I will remember that.

+ what am i old enough to do
* <get age> >= 21 => You can drink alcohol.
* <get age> >= 18 => You can vote and gamble.
* <get age> >= 16 => You can drive a car.
* <get age> <  16 => You can't do anything fun.
- I don't know how old you are yet.
```

Conditional Examples

```
+ my name is _
* <formal> == <bot name> => Wow, we have\s
  ^ the same name!<set name=<formal>>
- <set name=<formal>>Nice to meet you,\s
  ^ <get name>!

+ do you know my name
* <get name> == undefined => {random}
  ^ No, I don't.|
  ^ No, what is your name?
  ^ {/random}
- Yes, your name is <get name>!
- Your name is <get name>, grasshopper!
```

Topics

- Topics in RiveScript are “RunABot-like”
- Topics are used to segregate the trigger set
- A user may **only** match triggers that exist in their current topic
- Topics may inherit or include triggers that belong to other topics
- Inherit: local triggers are higher priority
- Include: triggers all grouped together equally

A Topic Example

```
+ imaginative curse word
- How rude are you? I won't talk to you until
^ you apologize.{topic=sorry}

> topic sorry
   + *
   - I won't talk until you apologize.
   - Say you're sorry now!

   + sorry
   - Ok! I'll forgive you!{topic=random}
< topic
```

The “BEGIN” Topic

- A special kind of topic
- Allows you to pre- and post-process your transaction
- Optional

Begin Topic Example

```
> begin
  + request
  * <bot maint> == true => Sorry, I'm currently
    ^ undergoing maintenance!
  - {ok}
< begin
```

- The “request” trigger is the entry point.
- Include “{ok}” in the reply for a real reply to be fetched, otherwise return something else.
- String tags like {uppercase} can be applied to {ok} too, for post-process reply formatting!

Inline Object Macros

- Define your own dynamic “objects” using regular program code.
- The Perl and Java libraries both support Perl code for these objects.
- You can define your own languages to handle.
- Objects allow your bot to provide services (look up weather, movie times, etc.)

Object Example

```
> object md5sum perl
  my ($rs, $args) = @_;

  use Digest::MD5 qw(md5_hex);
  return md5_hex(join(" ", @{$args}));
< object

+ encode * in md5
- The MD5 sum is: <call>md5sum <star></call>
```

RiveScript Availability

- **Perl module**
  - use RiveScript;
  - The original RiveScript library, feature-complete and where new features come first
- **Java library**
  - import com.rivescript.RiveScript;
  - Feature-complete, but hasn't been as thoroughly tested as the Perl module.

Perl RiveScript Distribution

- Comes with the `rivescript` command line utility – for testing your RiveScript code.
- Can also be used from a third-party program in non-interactive mode – communicating using JSON. So you can get RiveScript access without needing to use Perl or Java for your project!

`rivescript` JSON Example

```
$ echo '{“username”: “Kirsle”, “message”: \
“Hello bot” }' | rivescript --json /opt/rs/brain
{
    “status”: “ok”,
    “reply”: “Hello, human!”,
    “vars”: {
        “topic”: “random”,
        “__lastmatch__”: “hello bot”
    }
}
```

Learn More

- Visit RiveScript.com
- Download a complete AIM and YMSG RiveScript chatterbot, **AiRS**
- Download the Perl or Java libraries to use in your own projects
- Try it out live: www.rivescript.com/tryonline

RiveScript

Noah Petherbridge

www.rivescript.com
noah@kirsle.net