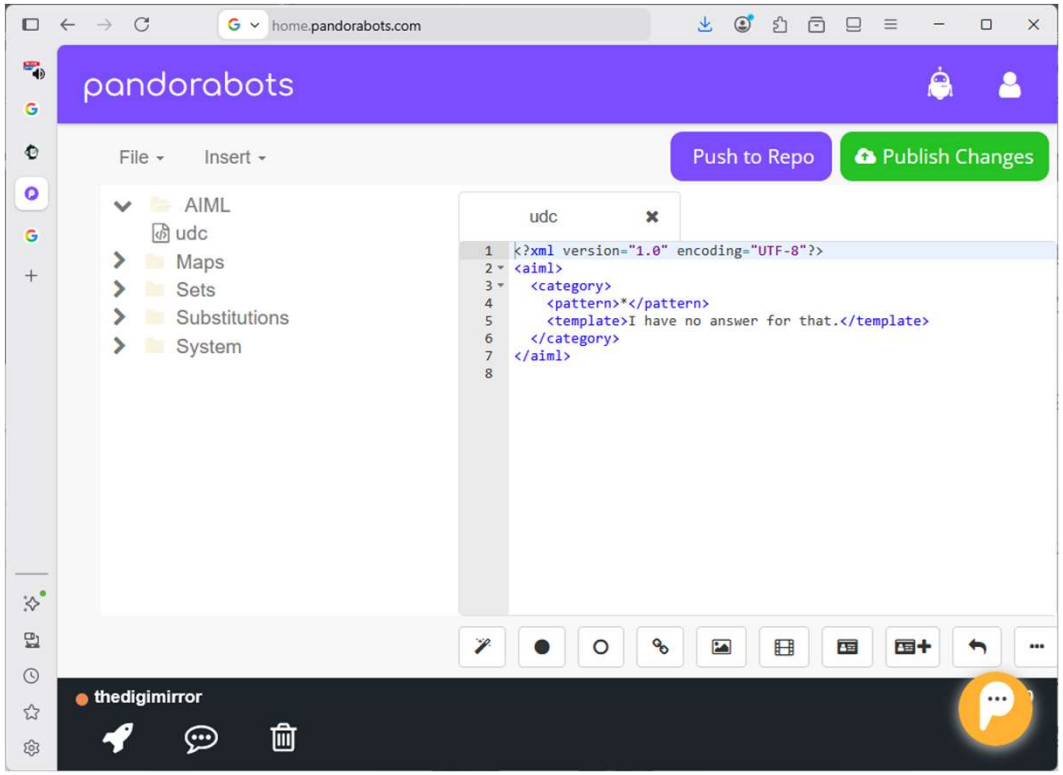
Build your …
Chatbot

Build your Chatbot … with PandoraBots

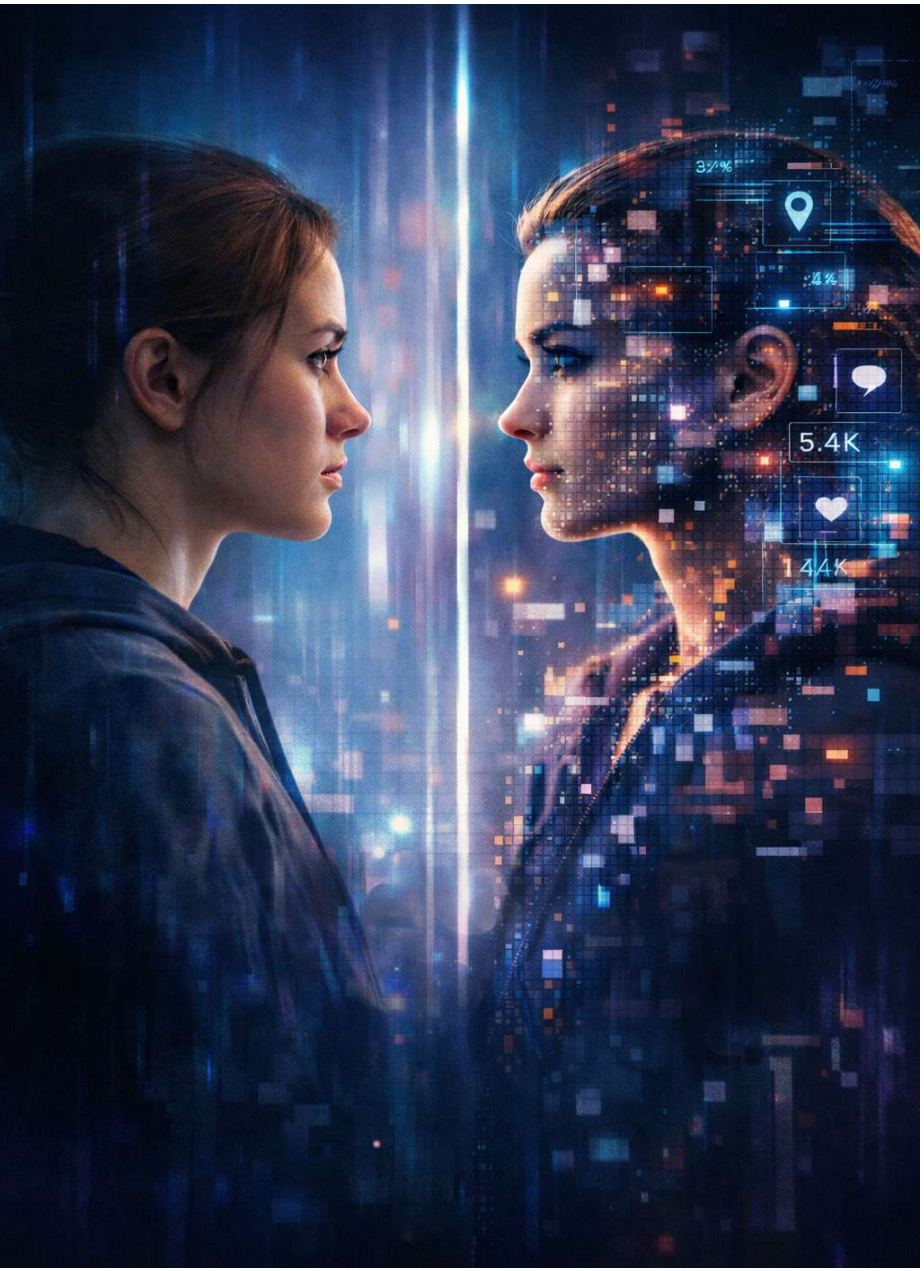


Pandorabots

How can we help you build a chatbot?

https://home.pandorabots.com/

Build your Idol (with AIML)



AIML ?

Artificial Intelligence Markup Language

→ XML-based language for describing dialogues for
- chatbots
- virtual assistants
- etc.

Goal: Replicating human conversations through rule-based dialogues

AI systems: Dynamic responses via statistical models or neural networks

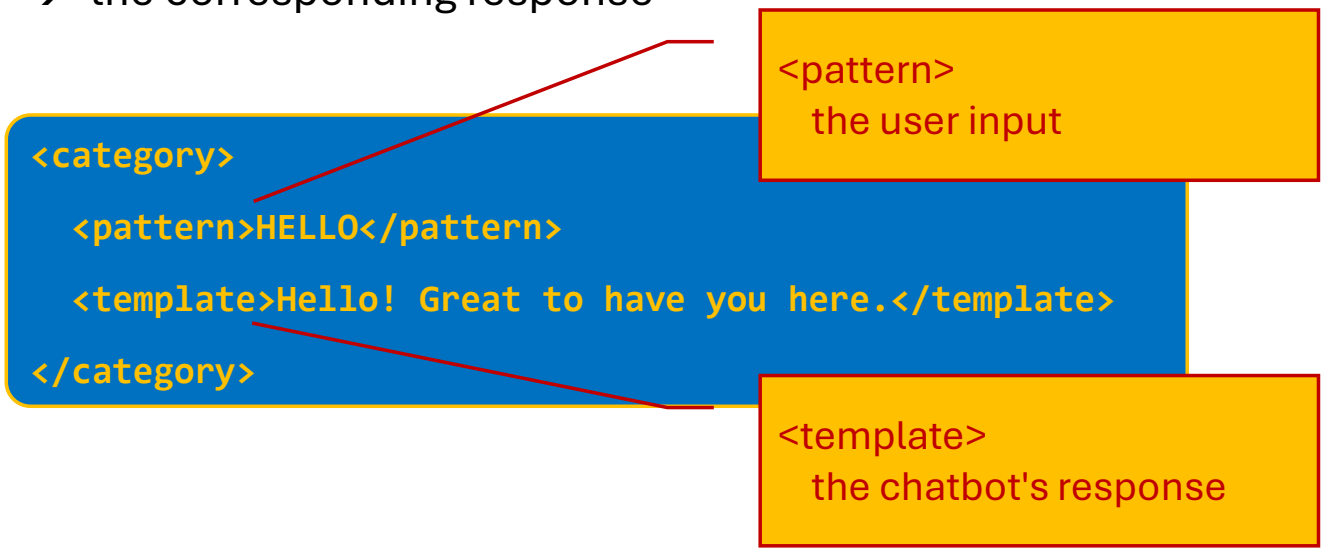
AIML: Responses are explicitly defined by rules

→ This makes the responses transparent, traceable, and easy to control

AIML Basic principle

AIML dialogues consist of a collection of rules.

Each rule describes:
→ a possible user input and
→ the corresponding response



AIML Categories

The fundamental unit in AIML is the

→ Each category represents exactly one dialogue rule
→ An AIML document often consists of many hundreds of such categories

```
<category>
    <pattern>WHAT'S YOUR NAME</pattern>
    <template>My name is ALICE.</template>
</category>
```

AIML Wildcards

To avoid having to specify every possible phrasing individually, AIML supports placeholders

The most important placeholder is the asterisk:

```
<pattern>I LIKE *</pattern>
```

The asterisk represents any text.

→ This allows inputs such as the following to be recognized:

```
I LIKE AVATARS
I LIKE PROGRAMMING
I LIKE VIRTUAL IDOLS
I LIKE STRAWBERRY ICE
```

AIML <star/>

Text recognized at the position of the asterisk
can be reused later in the response.

```
<category>
  <pattern>I LIKE *</pattern>
  <template>Why do you like <star/> ?</template>
</category>
```

```
User:      I LIKE AVATARS
Chatbot:   Why do you like AVATARS ?
```

AIML: Multiple possible answers

AIML supports random responses

→ Whenever "HELLO" is entered, one of the responses is selected at random.
→ This makes dialogues feel more natural..

```
<category>
  <pattern>HELLO</pattern>
  <template>
    <random>
        <li>Hello!</li>
        <li>Good day!</li>
        <li>Nice to see you!</li>
     </random>
  </template>
</category>
```

```
User:    Hello
Chatbot: Hello!
```

```
User:    Hallo
Chatbot: Good day!
```

```
User:    Hallo
Chatbot: Nice to see you!
```

AIML: Forwarding of rules

Often, multiple inputs are intended to produce the same response.
The `<srai>` tag is used for this purpose.

```
<category>
  <pattern>GOOD DAY</pattern>
  <template>
    <srai>HELLO</srai>
  </template>
</category>
...
<category>
  <pattern>HELLO</pattern>
  <template>Nice to see you!</template>
</category>
```

```
User:    Hello
Chatbot: Nice to see you!
```

```
User:    Good Day
Chatbot: Nice to see you!
```

The bot now treats "GOOD DAY" as if “HELLO" had been entered.
This allows large sets of rules to be organized clearly.

AIML: Save variables

AIML can store simple information.

```
<category>
  <pattern>MY NAME IS *</pattern>
  <template>
    <think>
      <set name="username"><star/></set>
    </think>
    Nice to meet you.
  </template>
</category>
```

Here, for example, the name is stored in the variable “username”

AIML: Save variables

AIML can store simple information.

```
<category>
  <pattern>MY NAME IS *</pattern>
  <template>
    <think>
      <set name="username"><star/></set>
    </think>
    Nice to meet you.
  </template>
</category>
```

Here, for example, the name is stored in the variable “username”

It can be accessed later:

```
<category>
  <pattern>WHAT IS MY NAME</pattern>
  <template>
        Your name is <get name="username"/>.
  </template>
</category>
```

```
User:      MY NAME IS PETER
Chatbot:   Nice to meet you.
```

```
Benutzer: WHAT IS MY NAME
Chatbot:   Your name is PETER.
```

AIML: Topics

AIML allows rules concerning specific subject areas to be grouped together using topics.

→ This gives the chatbot a simple form of context,
enabling it to answer the same question differently depending
on the current topic of conversation.

```
<category>
  <pattern>WHAT CAN IT DO</pattern>
  <template>
    What does your question refer to?
  </template>
</category>
```

Without topics, each rule must function independently.
Here, the bot does not know what "IT" refers to.

AIML: Topics

AIML allows rules concerning specific subject areas to be grouped together using topics.

With Topics, a subject can be set initially:

```
<category>
  <pattern>LETS TALK ABOUT AVATARS</pattern>
    <template>
        <think>
            <set name="topic">AVATAR</set>
        </think>
        Sure. What would you like to know about avatars?
    </template>>
</category>
```

```
<topic name="AVATAR">
  <category>
    <pattern>WHAT CAN IT DO </pattern>
    <template>
      An avatar is a representative.
    </template>
  </category>
</topic>
```

```
<topic name="VIRTUAL_IDOL">
  <category>
    <pattern>WHAT CAN IT DO</pattern>
    <template>
      A Virtual Idol is a specialised avatar.
    </template>
  </category>
</topic>>
```

AIML: Basic structure

The smallest possible basic structure of an AIML file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>

<aiml version="1.0.1" encoding="UTF-8">

  <category>
    <pattern>
      HELLO ALICE
    </pattern>
    <template>
      Hello User!
    </template>
  </category>

</aiml>
```

AIML: Basic structure

```
<?xml version="1.0" encoding="UTF-8"?>

<aiml version="1.0.1" encoding="UTF-8">

  <category>
    <pattern>
      HELLO ALICE
    </pattern>
    <template>
      Hello User!
    </template>
  </category>

</aiml>
```

identifies the file as an XML document.

AIML root element
→ contains all dialogue rules

A category contains/ describes exactly one rule

Pattern
→ Expected user input

Template
→ Chatbot response

Have fun :)

Thank you