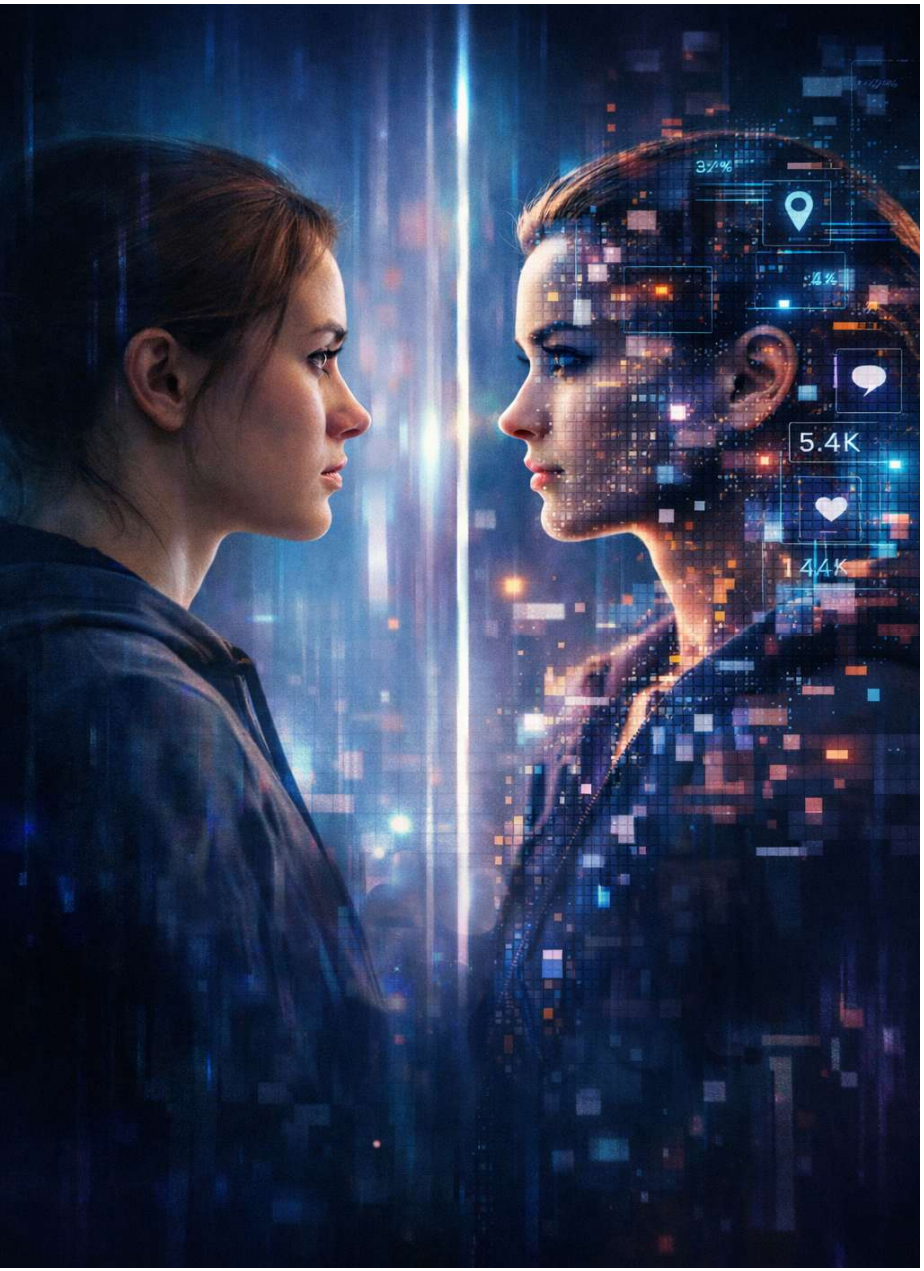
Build your Idol (with AIML)



AIML ?

Artificial Intelligence Markup Language

→ XML-basierte Sprache zur Beschreibung von Dialogen für
- Chatbots
- virtuelle Assistenten
- usw.

Ziel: Nachbildung menschlicher Gespräche durch regelbasierte Dialoge

KI-Systeme: dynamische Antworten durch statistische Modelle oder neuronale Netze

AIML: Antworten werden explizit durch Regeln festgelegt

→ Dadurch sind die Antworten transparent, nachvollziehbar und leicht kontrollierbar

AIML Grundprinzip

AIML-Dialoge bestehen aus einer Sammlung von Regeln

Jede Regel beschreibt:
→ eine mögliche Benutzereingabe und
→ die dazu passende Antwort

```
<category>
  <pattern>HALLO</pattern>
  <template>Hallo! Schön, dass du da bist.</template>
</category>
```

<pattern>
die Eingabe des Benutzers

<template>
die Antwort des Chatbots

AIML Kategorien

Die grundlegende Einheit in AIML ist die Kategorie.

Jede Kategorie repräsentiert genau eine Dialogregel
Ein AIML-Dokument besteht häufig aus vielen hundert solcher Kategorien

```
<category>
    <pattern>WIE HEISST DU</pattern>
    <template>Ich bin ALICE.</template>
</category>
```

AIML Wildcards

Damit nicht jede mögliche Formulierung einzeln angegeben werden muss, unterstützt AIML Platzhalter

Der wichtigste Platzhalter ist das Sternchen:

```
<pattern>ICH MAG *</pattern>
```

Das Sternchen steht für beliebigen Text

→ Dadurch können z.B. auch folgende Eingaben erkannt werden:

```
ICH MAG AVATARE
ICH MAG PROGRAMMIERUNG
ICH MAG VIRTUAL IDOLS
ICH MAG ERDBEEREIS
```

AIML <star/>

Text, der an der Stelle des Sternchens erkannt wird,
kann später in der Antwort wieder verwendet werden

```
<category>
  <pattern>ICH MAG *</pattern>
  <template>Warum magst du <star/> ?</template>
</category>
```

```
Benutzer: ICH MAG AVATARE
Chatbot:  Warum magst Du AVATARE ?
```

AIML: Mehrere mögliche Antworten

AIML unterstützt zufällige Antworten

→ Hier wird bei jeder Eingabe von „HALLO“ zufällig eine der Antworten gewählt

→ Dadurch wirken Dialoge natürlicher.

```
<category>
  <pattern>HALLO</pattern>
  <template>
    <random>
        <li>Hallo!</li>
        <li>Guten Tag!</li>
        <li>Schön dich zu sehen!</li>
     </random>
  </template>
</category>
```

```
Benutzer: Hallo
Chatbot:  Hallo!
```

```
Benutzer: Hallo
Chatbot:  Guten Tag!
```

```
Benutzer: Hallo
Chatbot:  Schön dich zu sehen!
```

AIML: Weiterleitung von Regeln

Häufig sollen mehrere Eingaben dieselbe Antwort erzeugen.
Dafür wird `<srai>` verwendet.

```
<category>
  <pattern>GUTEN TAG</pattern>
  <template>
    <srai>HALLO</srai>
  </template>
</category>
...
<category>
  <pattern>HALLO</pattern>
  <template>Schön Dich zu sehen!</template>
</category>
```

```
Benutzer: Hallo
Chatbot:  Schön Dich zu sehen!
```

```
Benutzer: Guten Tag
Chatbot:  Schön Dich zu sehen!
```

Der Bot behandelt „GUTEN TAG“ nun so, als wäre „HALLO“ eingegeben worden.
Dadurch lassen sich große Regelmengen übersichtlich organisieren.

AIML: Variablen speichern

AIML kann einfache Informationen speichern.

```
<category>
  <pattern>MEIN NAME IST *</pattern>
  <template>
    <think>
      <set name="username"><star/></set>
    </think>
    Freut mich, dich kennenzulernen.
  </template>
</category>
```

Hier wird z.B. der Name in der Variablen `username` gespeichert.

AIML: Variablen speichern

AIML kann einfache Informationen speichern.

```
<category>
  <pattern>MEIN NAME IST *</pattern>
  <template>
    <think>
      <set name="username"><star/></set>
    </think>
    Freut mich, dich kennenzulernen.
  </template>
</category>
```

Hier wird z.B. der Name in der Variablen `username` gespeichert.

Später kann darauf zugegriffen werden:

```
<category>
  <pattern>WIE HEISSE ICH</pattern>
  <template>
        Du heißt <get name="username"/>.
  </template>
</category>
```

```
Benutzer: MEIN NAME IST PETER
Chatbot:  Freut mich, dich kennenzulernen.
```

```
Benutzer: WIE HEISSE ICH
Chatbot:  Du heißt PETER
```

AIML: Topics

AIML kann mit Topics Regeln zu bestimmten Themenbereichen zusammengefassen.
→ Dadurch erhält der Chatbot eine einfache Form von Kontext und kann dieselbe Frage unterschiedlich beantworten, je nachdem, worüber gerade gesprochen wird.

Ohne Topics muss jede Regel unabhängig funktionieren.

```
<category>
  <pattern>WAS KANN ES</pattern>
  <template>
    Worauf bezieht sich deine Frage?
  </template>
</category>
```

Hier weiß der Bot nicht, worauf sich „ES“ bezieht.

AIML: Topics

AIML kann mit Topics Regeln zu bestimmten Themenbereichen zusammengefassen.

Mit Topics kann zunächst ein Thema gesetzt werden:.

```
<category>
  <pattern>LASS UNS UEBER AVATARE SPRECHEN</pattern>
    <template>
        <think>
            <set name="topic">AVATAR</set>
        </think>
        Gerne. Was möchtest du über Avatare wissen?
    </template>>
</category>
```

```
<topic name="AVATAR">
  <category>
    <pattern>WAS KANN ES</pattern>
    <template>
      Ein Avatar ist ein Stellvertreter.
    </template>
  </category>
</topic>
```

```
<topic name="VIRTUAL_IDOL">
  <category>
    <pattern>WAS KANN ES</pattern>
    <template>
      Ein Virtual Idol ist ein besonderer Avatar.
    </template>
  </category>
</topic>>
```

AIML: Grundgerüst

Das kleinstmögliche Grundgerüst einer AIML-Datei sieht so aus:

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

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

  <category>
    <pattern>
      HALLO ALICE
    </pattern>
    <template>
      Hallo Benutzer!
    </template>
  </category>

</aiml>
```

AIML: Grundgerüst

Das kleinstmögliche Grundgerüst einer AIML-Datei sieht so aus:

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

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

  <category>
    <pattern>
      HALLO ALICE
    </pattern>
    <template>
      Hallo Benutzer!
    </template>
  </category>

</aiml>
```

kennzeichnet die Datei als XML-Dokument.

AIML-Wurzelelement
→ enthält alle Dialogregeln

Eine Kategorie enthält/ beschreibt genau eine Regel

Pattern
→ Erwartete Benutzereingabe

Templare
→ Antwort des Chatbot

Have fun :)

Thank you