AIML - Artificial Intelligence Markup Language

Iryna Repinetska, Chris Roeseler

Institut für Informatik Humboldt-Universität zu Berlin

November 4, 2016

- Die kognitiven Schnittstellen als die neue Form des Zusammenspiels zwischen Menschen und Maschienen.
- Ein Beispiel von solchen kognitiven Oberflächen sind Chatbots oder Programme, die ein Dialog mit Menschen simulieren sollen.
- Artificial Intelligence Markup Language (AIML) wurde in den Jahren von 1995 bis 2000 by Richard Wallace entwickelt und ist heuzutage am meisten benutzte Sprache für die Entwicklung von Chatbots.

- Artificial Intelligence Markup Language basiert auf den Konzepten der Mustererkennung.
- AIML ist eine XML-basierte sowie tag-basierte Markup Sprache.
- Die Allgemeine Form eines AIML Befehls hat die folgede Struktur:

$$< Befehl > ParameterListe < /Befehl >$$

.

- Die grundlegende Einheiten des Dialoges werden Kategorien (categories) gennant und bilden die Wissensbasis des Chatbots.
- Das AIML-Vokabular besteht aus Wörtern, Leerzeichen und den Sonderzeichen $*$ und _ .

<aiml>

- Markiert Anfang und Ende des AIML Dokuments
- Optional Angabe von Version und Kodierung
- Vergleichbar mit <html></html>

```
<aiml version="1.0.1" encoding="UTF-8"?>
<category>
    <pattern> HELLO </pattern>
    <template>
        Hello User
    </template>
</category>
</aiml>
```

<category>

- Genannt knowledge unit
- Enthält:
  - Möglichen Anfrage String(<pattern>)
  - Antwort/en des Bots(<template>)
  - Optional: Kontext(<topic>)

```
<aiml version="1.0.1" encoding="UTF-8"?>
<category>
   <pattern> HELLO </pattern>
   <template>
      Hello User
   </template>
</category>
</aiml>
```

<pattern>

- Matched den user input
- Kann wildcard character enthalten
- case insensitiv

```
<aiml version="1.0.1" encoding="UTF-8"?>
<category>
    <pattern> HELLO </pattern>
    <template>
        Hello User
    </template>
</category>
</aiml>
```

<template>

- Antwort des Bots
- Informationen speichern für spätere Gespräche
- Programme aufrufen
- Weiterführende Fragen stellen

```
<aiml version="1.0.1" encoding="UTF-8"?>
<category>
    <pattern> HELLO </pattern>
    <template>
        Hello User
    </template>
</category>
</aiml>
```

<star>

- Korrespondiert mit wildcard character
- Indizierung aus pattern möglich

```
<category>
  <pattern> I AM * FROM * </pattern>

  <template>
    <star index=''2''/> is a nice region... <star index=''1''/> is a shit name though.
  </template>

</category>
```

<srai>

- Matched ein <pattern>und stellt die Anfrage geändert neu
- Divide and Conquer - Beispiel Bot
- Symbolic Reduction - Pattern vereinfachen
- Synonyms resolution
- Keywords detection

```
<category>
  <pattern> HELLO </pattern>
  <template>
    Hello User
  </template>
</category>
```

```
  <category>
    <pattern> HELLO *</pattern>

    <template>
      <srai>HELLO</srai>
    </template>
  </category>
```

<random>

- verschiedene Antwortmöglichkeiten die zufällig ausgegeben werden

```
<category>
  <pattern>HELLO</pattern>
  <template>
    <random>
      <li> Hello! </li>
      <li> Hi! Nice to meet you! </li>
    </random>
  </template>
<category>
```

<that>

- Kontext Antworten
- <that>: zuletzt ausgegebenes template
- Zusätzliche notwendige Bedingung beim matching

```
<category>
   <pattern>WHAT ABOUT MOVIES</pattern>
   <template>Do you like comedy movies</template>
</category>

<category>
   <pattern>YES</pattern>
   <that>Do you like comedy movies</that>
   <template>Nice, I like comedy movies too.</template>
</category>

<category>
   <pattern>NO</pattern>
   <that>Do you like comedy movies</that>
   <template>Ok! But I like comedy movies.</template>
</category>
```

<set><get><think>

- Speichern von Variablen z.B. Name

```
<category>
  <pattern>I am *</pattern>
  <template>
    Hello <set name = "username"> <star/> </set>!
  </template>
</category>

<category>
  <pattern>Good Night</pattern>
  <template>
    Bye<get name = "username"/> Thanks for the conversation!
  </template>
</category>
```

- think speichert im Hintergrund

```
  <template>
    Hello <think><set name = "username"> <star/>! </set></think>
  </template>
```

<topic>

- Definiert catergory's die nur bei gesetztem topic matchen

```
<category>
   <pattern>LET DISCUSS MOVIES</pattern>
   <template>Yes <set name = "topic">movies</set></template>
</category>

<topic name = "movies">
   <category>
      <pattern> * </pattern>
      <template>Watching good movie refreshes our minds.</template>
   </category>

   <category>
      <pattern> I LIKE WATCHING COMEDY! </pattern>
      <template>I like comedy movies too.</template>
   </category>

   <category>
      <pattern> STOP TALKING ABOUT MOVIES! </pattern>
      <template>OK.<think><set name = ''topic''></set></think></template>
   </category>
</topic>
```

<condition>

- Ähnlich zu break
- Bedingte Antworten im template

```
<category>
  <pattern> HOW ARE YOU? </pattern>
  <template>
   <random>
     <li><think><set name = ''state''> happy</set></think></li>
     <li><think><set name = ''state''> sad </set></think></li>
   </random>
  <condition name="state" value="happy">
   I am happy.
  </condition>
  <condition name="state" value="sad">
   I am sad.
  </condition>
 </template>
</category>
```

AIML ist heuzutage die am weit verbreitetse Programmiersprache für die Entwicklung von Chatbots. Diese Verbreitung beruht auf folgenden Gründen:

- AIML ist durch eine einfache Anwendung gekennzeichnet. Da sie auf XML (eXtensible Markup Language) basiert und die Implentation von Dialogen unter Verwendung von Tags einfacher ist;
- es gibt verschiedene AIML-Editoren sowie Entwicklungsplattformen, die den Chatbot-Entwicklern bei der Codeerstellung sowie Web-Bereitstellung des Chatbots helfen;
- Der wesentliche Teil aller Chatbot-Projekte, die mit AIML implementiert wurden, sind Open Source Software. Das gibt die Möglichkeit Source Code sowie die entsprechende Dokumentation für die neuen Projekte zu benutzen.