人工智能：用AIML写一个机器人

博客分类： java

XML 正则表达式 Groovy Eclipse 嵌入式

最近搞了一把人工智能，感觉AIML(Artificial Intelligence Mark-up Language)确实是个好东西，特笔记之。

AIML OVERVIEW:

http://www.pandorabots.com/pandora/pics/wallaceaimltutorial.html

AIML的一个java引擎：

http://www.geocities.com/phelio/chatterbean/?200931#BOTS

1: AIML OVERVIEW

首先看看AIML到底长啥样：

Xml代码

```xml
<aiml>
<category><pattern>WHO WANTS TO KNOW</pattern><template>ALICE wants to know.</template></category>
<category><pattern>WHY ARE YOU CALLED</pattern><template>  <srai>WHAT DOES ALICE STAND FOR</srai>  </template></category>
<category><pattern>WHY ARE YOU NAMED *</pattern><template>  <srai>WHAT DOES ALICE STAND FOR</srai>  </template></category>
<category><pattern>WHY DO YOU DREAM *</pattern><template>I dream about adding new code to ALICE.</template></category>
<category><pattern>WHY SILVER</pattern><template>ALICE is competing for the Loebner Silver Medal.</template></category>
<category><pattern>WHY WERE YOU NAMED ALICE</pattern><template>
<srai>WHAT DOES ALICE STAND FOR</srai></template></category>
<category><pattern>WHY WERE YOU NAMED</pattern><template><srai>WHAT DOES ALICE STAND FOR</srai>
</template></category>
<category><pattern>WHY</pattern><that>I AM SMARTER *</that><template>ALICE won an award for being the "most human" robot.</template></category>
<category><pattern>WOULD ALICE *</pattern><template><srai>WOULD YOU <star/> </srai> </template>
</category>
</aiml>
```

简单说明一下AIML的常用标签:

1：pattern tag:支持模式匹配(正则表达式，模糊匹配),及基于template的返回

2：random tag:支持随机回答（一对多）

4：think,system tag: 支持简单逻辑记忆及自定义函数(本来打算扩展一下AIML, 搞一个支持Groovy语言的标签， 结果看到了它的<system>标签， 遂作罢)

5：javascript tag: 支持嵌入js脚本(适用于web chat开发，比如根据情绪改变表情等)。

6：srai tag: 支持多对一回答.

详细内容请参加AIML的官方文档：

http://alicebot.org/TR/2005/WD-aiml/WD-aiml-1.0.1-008.html

2：按照AIML， 鄙人整理了一个简单的DEMO，扩展了AIML的DATE标签，支持了java的时间掩码，并有自学习功能.

机器人Alice表现如下，可见其多么聪明:

Html代码 载 ☆

```
Alice>Hi ya! Welcome!
you say>what's your name
Alice>sorry, what?
you say>this is not a good answer  //开始学习功能
Alice>Sorry. What would be a good answer?
you say>my name is Alice
Alice>Alright! You can ask me again to see if I got it.
you say>what's your name
Alice>my name is Alice
you say>what is your name
Alice>my name is Alice
you say>my name is Lichunlei
Alice>hello, Lichunlei.
you say>do you remember me?
Alice>Your name is Lichunlei, seeker.  //Alice的记忆功能
you say>what's time now?
Alice>It is 10:59 A.M.
you say>what date is today?
Alice>Monday.
```

如果感觉机器人Alice的答案不满意， 只需输入包含not和good answer的句子，在你的指导下，Alice就可以开始学习新知识。

让它如此智慧的原因就是AIML文件， 此为机器人的大脑.

下为Alice的AIML文件:

Xml代码 载 ☆

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<aiml>
<!-- Copyright (c) 2007 ALICE A.I. Foundation, Inc. -->
<!-- Last modified Seo 21, 2009, by Lichunlei -->

<category><pattern>WHAT IS TIME *</pattern><template>It is <date format="h:mm a"/>.</template></category>
<category><pattern>WHAT DAY IS TODAY</pattern><template><date format="E"/>.</template></category>
<category><pattern>WHAT IS TODAY *</pattern><template><date format="EEE"/>.</template></category>
<category><pattern>MY NAME IS *</pattern><template><think><set name="name"><star/></set></think>hello, <get name="name"/>.</template></category>
<category><pattern>DO YOU REMEMBER ME</pattern><template>Your name is <get name="name"/>, seeker.</template></category>
<category><pattern>I CAN NOT *</pattern><template>Why can't you do <set name="it"><person/></set>?</template></category>
```

```
<category><pattern>MY INPUT</pattern> <template> 1:<input index="1"/> 2:<input index="2"/> 3:<input index="3"/></template></category>
<category><pattern>*</pattern><template>sorry, what?</template>
</category>

  <!-- Greeting categories. -->
  <category>
    <pattern>WELCOME</pattern>
    <template>
      <think>
        <system> <!-- Defines a method to create new categories from user input at run-time. -->
          import bitoflife.chatterbean.AliceBot;
          import bitoflife.chatterbean.Context;
          import bitoflife.chatterbean.Graphmaster;
          import bitoflife.chatterbean.aiml.Category;
          import bitoflife.chatterbean.text.Transformations;

          void learn(String pattern, String template)
          {
            /* The "match" variable represents the current matching context. */
            AliceBot bot = match.getCallback();
            Context context = bot.getContext();
            Transformations transformations = context.getTransformations();

            pattern = transformations.normalization(pattern);
            Category category = new Category(pattern, new String[] {template});
            Graphmaster brain = bot.getGraphmaster();
            brain.append(category);
          }
        </system>
      </think>
      Hi ya! Welcome!
    </template>
  </category>

  <!-- A category set to learn simple user-fed categories. -->
  <category>
    <pattern>* NOT *  GOOD ANSWER</pattern>
    <template>
      Sorry. What would be a good answer?
    </template>
  </category>
  <category>
    <pattern>_</pattern>
    <that>WHAT WOULD BE A GOOD ANSWER</that>
    <template>
      <system>learn("<input index="3"/>", "<input index="1"/>")</system>
      Alright! You can ask me again to see if I got it.
    </template>
  </category>
</aiml>
```

之所以Alice可以学习， 重要的一点是<input/>标签，此标签记住了之前对方的聊天记录， 通过index可以得到(倒序

索引)

程序相对简单，两个class：

Alice工厂： AliceBotMother

Java代码

```
1.  package co.aiml;
2.
3.  import java.io.FileInputStream;
4.  import java.io.ByteArrayOutputStream;
5.
6.  import bitoflife.chatterbean.AliceBot;
7.  import bitoflife.chatterbean.Context;
8.  import bitoflife.chatterbean.parser.AliceBotParser;
9.  import bitoflife.chatterbean.util.Searcher;
10.
11. public class AliceBotMother
12. {
13.
14.   private ByteArrayOutputStream gossip;
15.
16.
17.   public void setUp()
18.   {
19.     gossip = new ByteArrayOutputStream();
20.   }
21.
22.   public String gossip()
23.   {
24.     return gossip.toString();
25.   }
26.
27.   public AliceBot newInstance() throws Exception
28.   {
29.     Searcher searcher = new Searcher();
30.     AliceBotParser parser = new AliceBotParser();
31.     AliceBot bot = parser.parse(new FileInputStream("Bots/context.xml"),
32.                                 new FileInputStream("Bots/splitters.xml"),
33.                                 new FileInputStream("Bots/substitutions.xml"),
34.                                 searcher.search("Bots/mydomain", ".*\\.aiml"));
35.
36.     Context context = bot.getContext();
37.     context.outputStream(gossip);
38.     return bot;
39.   }
40. }
```

命令行聊天程序：

Java代码

```
1.  package co.aiml;
2.
```

```
3.  import java.io.BufferedReader;
4.  import java.io.IOException;
5.  import java.io.InputStreamReader;
6.
7.  import bitoflife.chatterbean.AliceBot;
8.
9.  public class Chat
10. {
11. public static final String END = "bye";
12.
13. public static String input()
14. {
15. BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
16. System.out.println("you say>");
17. String input = "";
18. try
19. {
20. input = in.readLine();
21. } catch (IOException e) {
22. // TODO Auto-generated catch block
23. e.printStackTrace();
24. }
25. return input;
26. }
27.
28. public static void main(String[] args) throws Exception
29. {
30.      AliceBotMother mother = new AliceBotMother();
31.      mother.setUp();
32.      AliceBot bot = mother.newInstance();
33.      System.err.println("Alice>" + bot.respond("welcome"));
34. while(true)
35. {
36. String input = Chat.input();
37. if(Chat.END.equalsIgnoreCase(input))
38. break;
39.
40. System.err.println("Alice>" + bot.respond(input));
41. }
42.
43. }
44. }
```

需要说明的是:

Java代码 载 ☆

```
1.  AliceBot bot = parser.parse(new FileInputStream("Bots/context.xml"),
2.                              new FileInputStream("Bots/splitters.xml"),
3.                              new FileInputStream("Bots/substitutions.xml"),
4.                              searcher.search("Bots/mydomain", ".*\\.aiml"));
```

context.xml：设置application的属性， 及时间格式等可变属性

Xml代码 载 ☆

```xml
<context>
<!-- The id is a unique string that identifies this context. -->
<bot name="id" value="test_cases" />

<!-- Bot predicates are set at load time, and cannot be changed at runtime. -->
<bot name="output" value="Logs/gossip.txt" />
<bot name="randomSeed" value="1" />
<bot name="series" value="Alpha" />
<bot name="version" value="0.7.5 Alpha" />
<bot name="location" value="Atlanta" />
<bot name="name" value="Alice" />

<!-- Default values for predicates, can be changed later at runtime. -->
<set name="dateFormat" value="yyyy-MM-dd HH:mm:ss" />
<set name="name" value="dear friend" />
<set name="me" value="Alice" />
<set name="engine" value="ChatterBean" />
<set name="topic" value="*" />
</context>
```

如上属性，都可以用AIML的<bot>标签及<get>标签访问得到。

splitters.xml：定义什么是句子，即句子的结束符。

Xml代码 载 ☆

```xml
<splitters>
  <splitter value="..." type="sentence"/>
  <splitter value="." type="sentence"/>
  <splitter value="!" type="sentence"/>
  <splitter value="?" type="sentence"/>
  <splitter value=";" type="sentence"/>
  <splitter value="," type="word"/>
  <splitter value=":" type="word"/>
</splitters>
```

substitutions.xml：定义容错规则及特殊字符映射等。

Xml代码 载 ☆

```xml
<substitutions>
  <!-- Input substitutions correct spelling mistakes and convert "sentence"-ending characters i
nto characters that will not be identified as sentence enders. -->
  <input>
    <correction><!--sentence correction-->
      <substitute find="=reply" replace=""/>
      <substitute find="name=reset" replace=""/>
      <substitute find=":-)" replace=" smile "/>
```

```
      <substitute find=":)" replace=" smile "/>
      <substitute find=",)" replace=" smile "/>
      <substitute find=";)" replace=" smile "/>
      <substitute find=";-)" replace=" smile "/>
      <substitute find="&amp;quot;" replace=""/>
      <substitute find="/" replace=" "/>
      <substitute find="&amp;gt;" replace=" gt "/>
      <substitute find="&amp;lt;" replace=" lt "/>
      <substitute find="(" replace=" "/>
      <substitute find=")" replace=" "/>
      <substitute find=" u " replace=" you "/>
      <substitute find=" ur " replace=" your "/>
      <substitute find=" you'd " replace=" you would "/>
      <substitute find=" you're " replace=" you are "/>
      <substitute find=" you re " replace=" you are "/>
      <substitute find=" you've " replace=" you have "/>
      <substitute find=" you ve " replace=" you have "/>
     <substitute find=" what's " replace=" what is "/>
      ...
    </correction>
    <protection><!-- sentence protection -->
      <substitute find=",what " replace=".  what "/>
      <substitute find=", do " replace=".  do "/>
      <substitute find=",do " replace=".  do "/>
      ...
    </protection>
  </input>
  <gender>
    <substitute find=" on her " replace="on him"/>
    <substitute find=" in her " replace="in him"/>
    <substitute find=" his " replace="her"/>
    <substitute find=" her " replace="his"/>
    <substitute find=" him " replace="her"/>
    ...
  </gender>
  <person>
    <substitute find=" I was " replace="he or she was"/>
    <substitute find=" mine " replace="his or hers"/>
  </person>
  <person2>
    ...
    <substitute find=" your " replace="my"/>
  </person2>
</substitutions>
```

比如在上面的聊天DEMO中， 我输入what's your name， 和输入what is your name， 都能得到正确的回答，这是因为：

在substitutions.xml文件中有如下设置；

Xml代码

```
<substitute find=" what's " replace=" what is "/>
```

3：扩展AIML标签(基于AIML的java引擎：chatterbean)：

package bitoflife.chatterbean.aiml是chatterbean对于AIML标签的实现包。目前为止，实现了大多数常用AIML标签.

而对date标签只有一个最简单的实现， 也不支持java时间掩码.

鄙人理想中的date标签应该是：

Xml代码

```xml
<category><pattern>WHAT IS TIME *</pattern><template>It is <date format="h:mm a"/>.</template>
</category>
```

标签类只需扩展TemplateElement即可。

所以， 修改之:

Java代码

```java
public class Date extends TemplateElement
{

  private final SimpleDateFormat format = new SimpleDateFormat();

  /**date tag format value, add by lcl**/
  private String formatStr = "";

  public Date()
  {
  }

  public Date(Attributes attributes)
  {
  //得到时间掩码
  formatStr = attributes.getValue(0);
  }

  public String process(Match match)
  {
    try
    {
    format.applyPattern(formatStr);
    return format.format(new java.util.Date());
    }
    catch (Exception e)
    {
    return defaultDate(match);
    }
  }

  private String defaultDate(Match match)
  {
    try
    {
      format.applyPattern((String) match.getCallback().getContext().property("predicate.dateFor
mat"));
      return format.format(new java.util.Date());
```

```
38.     }
39.     catch (NullPointerException e)
40.     {
41.       return "";
42.     }
43.   }
44.
45. }
```

4：要想让Alice足够聪明， 必须要有足够多的AIML, 如下地址是其所有的资料库：
http://www.alicebot.org/downloads/sets.html
加入到程序中， Alice几乎无所不知了。

5：如果需要做一个某领域的机器人专家， 基于AIML来实现，是一个不错的选择。

6:附件是Alice源码，及其上面的DEMO，eclipse工程.

Alice.rar (504.4 KB)
下载次数: 789