INTERPRETADOR AIML ALIMENTADO COM TAGS HTML5
MANUAL DE COMANDOS DO AIML

Autor: Rafael Luiz de Macedo (rafaelldemacedo@gmail.com)
(Bacharelado em Ciência da Computação, Centro Universitário Eurípides de Marília - UNIVEM)
Orientador: Fabio Dacencio Pereira (prof.fabiopereira@gmail.com)

O AIML (Artificial Intelligence Markup Language, Linguagem de Marcação da Inteligência Artificial) é um conjunto de tags XML(eXtensible Markup Language, Linguagem de Marcação Extensível) capaz de representar e relacionar expressões em linguagem natural permitindo a criação de motores capazes de manter um diálogo simples em linguagem natural.

Cada conjunto de tags pode possui uma ou varias tags chamada de categorias, pois as tags categorias possuem varias tipos de tags dentro de si que fazem a interpretação de uma mensagem escrita por um usuário e assim dar uma resposta através de funções que estão definidas nas tags dentro da tag categoria. AIML é um chatterbots, ou seja, um robô com Inteligência Artificial que interage com usuário através de conversas. Chatterbots significa: Chat = Conversa e bot = Robô.

Para desenvolver chatterbots existem varias plataformas e em varias linguagens, nesse projeto esta usando um interpretador em java e a plataforma ProgramD. O ProgramD é uma plataforma mais utilizada no desenvolvimento de chatterbots em AIML, e a mais completa em recurso além de ser a melhor implementação já testada do AIML.

O software GaitoBot é uma ferramenta que facilita o desenvolvimento de chatterbots na linguagem AIML, ele tem uma interface gráfica que com apenas um click de um botão você pode criar um novo conjunto de TAGs Category ou adicionar uma nova TAG dentro de um conjunto de TAG já criada.

Em seguida será mostrado algumas TAGs do AIML, com explicação da função que cada uma exerce no AIML.

1. **Comandos.**

Para fechar uma Tag usa-se o símbolo “/”.
Exemplo: Inicia - <category>.
Fecha - </category>.

<aiml> - Ela inicia e fechar um documento AIML </aiml>.

<category> - É uma unidade de conhecimento, ou seja, a base de conhecimento de um AIML.

<pattern> - Indica uma mensagem que é digitada pelo usuário.

<template> - É a mensagem que retorna como resposta pro usuário.

<random> - Indica uma resposta aleatoriamente através das respostas que possui.

** -** É o que marca um bloco de resposta usado dentro da tag <random>.

<that> - É o que registra a ultima sentença

<srai> - Ele direciona pra uma categoria já criada de acordo com a mensagem em que o usuário digitou assim ele usa a resposta que esta na categoria direcionada, mas para isso a mensagem que possui na categoria que ele ira redirecionar deve conter dentro da tag <srai>. Assim evitando a redundância de informações.

<star> - Se uma mensagem de entrada conter alguma palavra ou uma mensagem que esteja dentro de uma categoria que esta sendo usado o símbolo “*” ou “_” no começo ou no fim ou até mesmo no começo e no fim da mensagem, o símbolo faz com que as palavras que esteja antes ou depois da mensagem contida nesta categoria sejam irrelevantes, assim podendo executar essa categoria e fazer a função contida nela. O símbolo “*” representa a <star> que pode ser escrita na forma de tag <star> ou o símbolo “*”.

Exemplo:

<pattern> Oi * </pattern> = Oi, tudo bem?

O símbolo “*” também pode ser usado para armazenar as palavras contidas em si próprio em uma variável criada por uma tag <set>.

<think> - Essa tag é responsável por realizar as operações internas.

<set> - Armazena uma palavra ou mensagem digitada pelo usuário, para poder armazenar a palavra é necessário utilizar a TAG <star> ou “*”. Dentro da tag <set> é criada a variável. Exemplo:

```
<category>
    <pattern>Me chamo *</pattern>
    <template>
        <think><set name=”nome”>*</set></think>. (O nome será armazenado na variável “nome”).
    </template>
</category>
```

<get> - Faz a leitura de uma variável já criada, uma das TAGs que é mais utilizada em chatterbot, como no exemplo anterior, a TAG <set> cria e armazena o nome do humano em uma variável e assim podendo usar a variável toda vez que for necessário dizer o nome do humano em uma mensagem. Exemplo:

```
<category>
    <pattern>Você sabe como eu me chamo</pattern>
    <template>
        Você se chama <get name=”nome”>
    </template>
</category>
```

<thatstar> - Retorna o fragmento do texto de entrada do pattern "*" contido na <that></that>.

ADVANCED

<condition> - Existem três tipos de tags condição, a multi condition, list-condition e single name list-condition. A diferença entra as formas de condição é que a multi condition será toda avaliada, ou seja, ela não parará mesmo se houver alguma tag verdadeira, já a list-condition e single name list-condition para assim que a primeiro teste <li> for verdadeira.

<formal> - Essa tag tem a função de permitir que cada palavra seja alterada apenas a primeira letra da palavra de minúscula para maiúscula.
Exemplo:

<template>Meu nome é < formal>rafael macedo</formal></template>

Será mostrado para o usuário da seguinte forma: "Meu nome é Rafael Macedo."

<gender> - Esta tag faz com que seja alterado o verbo usado após um pronome.
Exemplo:

<gender>She told him to take a hike.</gender>

Nesse caso ele troca o pronome She = Ela por He = Ele e o verbo him por her. Assim mostrando em sua saída "He told her to take a hike".

<gender>Ela disse-lhe para fazer uma caminhada</gender>

"Ele disse a ela para fazer uma caminhada".

<input índex= "n"/> - Essa tag permite que seja dada uma resposta de entrada ao usuário de acordo como o valor que o "n" assumi, uma forma de explicar mais detalhado seria a seguinte maneira: Caso o valor seja 1, será dado a ultima resposta ao usuário, caso seja o valor 2 a segunda ultima resposta ao usuário, o valor 3 a terceira ultima resposta e assim por diante.
Exemplo:

Usuário: Meu nome é Rafael.

Robô: Rafael, você é casado?

Usuário: Não.

Robô: Que tipo de carro você dirige?

Usuário: Eu dirijo um Fusion.

<input índex=”1”/> Eu dirijo um Fusion.

<input índex=”2”/> Não.

<input índex=”3”/> Meu nome é Rafael.

É opcional utilizar o índex=”n”, pois o <input/> é equivalente ao <input índex=”n”>.

<person> - Essa tag interpreta o pronome pessoa (I, You, He, She, It, We, You, They) e altera do primeiro pronome pessoa para terceira pessoa. Mas há um principal problema que a tag <person> em Inglês é saber quando usar “I” (Eu) e quando usa “Me” (Eu).
Exemplo:

Usuário: I admire robots like you.

Robot: Joe said he admire robots like me.

Tradução:

Usuário: Eu admiro robôs como você.

Robô: Joe disse que admira robôs como eu.

A especificação do XML requer que cada início da tag <person> seja seguido por uma marca final correspondente como </person>, ele oferece suporte a uma notação abreviada em tags atômicas assim como <star/>, a tag <person> é outro exemplo de tags abreviada. Uma comparação que pode ser feita é através da tag utilizada em HTML <IMG>.
<person/> = <person><star/></person>.

<person2> - Essa tag tem a mesma função da tag anterior mostrado nesse manual <person>, mas com uma diferença que ela altera o primeiro pronome pessoa para a segunda pessoa.

<sentence> - Essa tag tem a função parecida com a tag <formal>, pois ela faz com que apenas a primeira letra da primeira palavra da mensagem seja alterada para maiúscula.
Exemplo:

<sentence>testando a tag sentence</sentence>

Será mostrada em sua saída a mensagem nessa forma: Testando a tag sentence.

<learn filename= ”X”> - Essa tag tem a função de carregar recursivamente um arquivo AIML. A semântica de um carregar é as mesmas de uma mesclagem, as categorias que são carregadas primeiras têm prioridade, excluindo as categorias que são padrões duplicatas.

<lowercase> - Essa tag já mostra qual é a sua função só pelo nome dela lowercase = letras minúsculas, pois ela transforma todas as letras maiúsculas da palavra para minúsculo.
Exemplo:

<template>Eu tenho 21<lowercase>ANOS</lowercase></template>

Assim será mostrado a mensagem dessa forma: “Eu tenho 21 anos”.

<uppercase> - Essa tag tem a função parecida com a tag anterior <lowercase>, a diferença que ela transformar todas as letras da palavra para minúsculo.
Exemplo:

<template>Estudo na <uppercase>univem</uppercase></template>

Em sua saída será mostrado a seguinte mensagem: “Estudo na UNIVEM”.

<personf> - Essa tag tem a mesma função da tag <person>, porém ela insere um escape “%20” em lugares que são espaços retornados pela <personf/> e devido a essas seqüencias de escapes acaba permitindo os métodos HTTP GET transmitam consultas de varias palavras. As seqüencias de caracteres de pesquisa que é utilizado nos sites Ask.com e Webster Dicionary utilizam <personf/>.
Exemplo:

<category>
<pattern>O que é um *</pattern>
<template>
O que faz
<a HREF="http://www.dictionary.com/cgi-bin/dict.pl?term=<personf/>">
<set_it> <person/> </set_it>
 significa?
ou com o pesquisador “Ask Jeeves”:
<a HREF="http://www.ask.com/AskJeeves.asp?ask=WHAT%20IS%20A
%20<personf/>">
O que é um <person/>?

</template>
</category>

TAGS EXPERIMENTAIS

Existem algumas tags experimentais, que são tags que não é padrão do AIML. Em seguida mostrarei algumas tags experimentais.

<justbeforethat> - Essa tag dar acesso a secundo ultima resposta, pois essa tag substitui a tag <that índex=”nx,ny”>.

<justthat> - Essa tag tem a função parecida com a tag anterior <justbeforethat>, porém a diferença que ela dar acesso a segunda ultima resposta que cliente/usuário fez.

HTML

Algumas TAGs em HTML podem ser utilizadas no AIML, pois o próprio AIML reconhece a função que é executada por cada uma dessas TAGs.
Abaixo mostra algumas tags HTML:

<a> - Mostra um link que direciona o humano a um site que já esteja citado dentro dela. Exemplo:

```
<category>
        <pattern>Site UNIVEM</pattern>
        <template>
                <a href=http://www.fundanet.br/ target=”web”>web<a>
        </template>
</category>
```

**
 -** Apenas da espaçamento igual a função da tecla enter do teclado.

<applet>;
;;<img>;<p>;<table>;<ul>;

Referencias

AIML Reference Manual disponível em: < http://www.alicebot.org/documentation/aiml-reference.html#justbeforethat> acessado em 05 de outubro de 2011.