DESENVOLVIMENTO DE UM SIMULADOR DE DIÁLOGO UTILIZANDO A LINGUAGEM AIML COM BANCO DE DADOS

André Damasceno Aoki[1], Francisco Assis da Silva[1], Marcelo Vinicius Creres Rosa[1]

[1]Faculdade de Informática (FIPP) – Universidade do Oeste Paulista (UNOESTE) – Presidente Prudente-SP. E-mail: marcelorosa@unoeste.br

RESUMO

Este trabalho apresenta uma aplicação para simular diálogos (chatterbot) utilizando a linguagem AIML (Artificial Intelligence Markup Language) com uma nova funcionalidade para prover respostas dinâmicas baseadas em consultas à banco de dados. Aplicações desse gênero são utilizadas para responder perguntas ou dúvidas que são solicitadas por pessoas. Os sistemas tradicionais são baseados em respostas fixas que utilizam a linguagem AIML e procuram respostas em bases de conhecimento que oferecem baixa manutenibilidade, comprometendo respostas que necessitam de atualizações periodicamente. A linguagem AIML foi extendida para que seja aplicada à uma fonte de conhecimento maior (banco de dados).

Palavras-chave: AIML - Simulação de diálogos - Respostas dinâmicas

DEVELOPMENT OF A SIMULATOR OF DIALOGUE USING THE LANGUAGE AIML WITH DATABASE

ABSTRACT

This paper presents an application to simulate dialogues (chatterbot) using the AIML language (Artificial Intelligence Markup Language) with a new functionality to provide dynamic answers to database queries. This kind of applications is used to answer questions or doubts that are asked by people. Traditional systems are based in fixed answers using AIML language and they search answers in knowledge bases that offer low maintainability, compromising responses that require periodic updates. The AIML language has been extended to be applied to a greater source of knowledge (database).

Keywords: AIML - dialogues simulation - dynamic answers

INTRODUÇÃO

Nos dias de hoje, instituições detentoras de informação procuram oferecer diversos meios de interação e comunicação para atender solicitações do público externo. Devido ao grande volume de informações pertencentes a estas instituições e a variedade de solicitações deste público nos canais de comunicação existentes, o serviço das centrais de atendimentos sobrecarregam. Um sistema computacional que consiga responder perguntas na forma de texto, solicitadas por meio de uma página na Internet, pode atender uma parte de um público, pessoas que desejam tirar dúvidas referente a uma empresa, instituição, portal, etc. Um simulador de diálogo pode ser usado para melhorar o serviço de atendimento desses locais. O simulador de diálogo conhecido por criar a linguagem AIML é o A.L.I.C.E (*Artificial Linguistic Internet Computer Entity*) c*hatterbot* criado na *Lehigh University* por Richard S. Wallace, ativada em 1995. Atualmente existe uma fundação que promove a disseminação do software gratuito ALICE e também da linguagem AIML usada na construção do simulador de diálogo ALICE [ALICE 1995].

De acordo com as pesquisas realizadas por Leonhardt (2003), a linguagem AIML apresenta um conjunto de *tags* e comandos para implementação da base de conhecimento de um c*hatterbot* e serve para analisar as mensagens enviadas pelo usuário e decidir a forma como estas mensagens devem ser respondidas. Uma frase escrita por um usuário é comparada aos padrões descritos na linguagem e com base neste processo são selecionadas ou construídas as respostas.

As principais *tags* da linguagem AIML são:

- **<aiml>** inicia e termina um bloco programado em AIML;
- **<category>** identifica uma "unidade de conhecimento" na base de conhecimento;
- **<pattern>** identifica um padrão de mensagem simples frequentemente utilizado por usuários;
- **<template>** contém a resposta para uma mensagem do usuário.

Com a utilização da linguagem AIML, pode-se definir mais de uma resposta para um único padrão e ainda poder especificar critérios de escolha de cada uma das respostas. Existe mais de 20 *tags* que compõem a linguagem AIML responsáveis por fornecer a necessária desenvoltura para o *chatterbot* propor uma solução à mensagem enviada [Leonhardt 2003].

Este trabalho tem por objetivo principal construir um simulador de diálogo tomando com base as funções padrões da linguagem AIML e principalmente criar uma nova função para prover uma comunicação entre a base de conhecimento AIML com outra base de dados específica (banco de dados), para que se tenha respostas mais atualizadas para determinadas perguntas.

As seções seguintes tratam dos trabalhos relacionados que serviram de base para o desenvolvimento deste, da estrutura do trabalho contendo a especificação da linguagem AIML usada no trabalho, dos módulos desenvolvidos (Interpretador, módulo Web e módulo desktop) e das conclusões.

TRABALHOS RELACIONADOS

Eliza

O simulador de diálogo ELIZA foi desenvolvido no MIT (*Massachusetts Institute of Technology)*, por Weizenbaum (1966). Este simulador de diálogo possui apenas 204 linhas de código fonte. Usa técnicas de Inteligência Artificial que simulava um diálogo entre paciente e

psicólogo utilizando as frases do paciente para formular novas perguntas.

Galvão (2003) em um trecho de sua tese transparece o funcionamento de ELIZA, segundo ele, ELIZA o cliente (paciente) assume o papel principal, sendo encorajado a falar a maior parte do tempo. O terapeuta atua resumindo, de vez em quando, o que o paciente disse e estimula-o a continuar falando, a fim de esclarecer os sentimentos do paciente, porém sem julgá-los ou explicá-los. A partir desse princípio, ELIZA, é livre para assumir que não conhece nada sobre o mundo real, não tendo que responder questões do usuário, nem falar sobre si mesmo, o que simplifica o seu desenvolvimento. O usuário é estimulado a tentar responder suas próprias perguntas, num processo de introspecção. A seguir (Figura 1), é transcrito um trecho de diálogo com ELIZA.

Usuário	Men are all alike.
ELIZA	IN WHAT WAY?
Usuário	They´re always bugging us about something or other.
ELIZA	CAN YOU THINK OF A SPECIFIC EXAMPLE?
Usuário	Well, my boyfriend made me come here.
ELIZA	YOUR BOYFRIEND MADE YOU CAME HERE.
Usuário	He says I´m depressed much of the time.
ELIZA	I AM SORRY TO HEAR YOU ARE DEPRESSED.
Usuário	It´s true. I AM unhappy.
...	
ELIZA	DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE.

Figura 1. Trecho de diálogo com ELIZA

O projeto ELIZA tem um modo bastante particular em tratar as frases que serão respondidas, é feito um pré-processamento para classificar as palavras chaves e assim formular uma resposta adequada [Weizenbaum 1966].

O projeto ELIZA tem algumas limitações. Não há uma memória na aplicação, ou seja, ela não consegue lembrar o que foi falado anteriormente. A construção de algumas respostas também pode desanimar o usuário, já que Eliza muitas vezes na tentativa de dar ao diálogo uma certa naturalidade, responde ao interlocutor usando partes da própria entrada, gerando às vezes diálogos um tanto confusos. Um exemplo dessa situação pode ser ilustrado da seguinte forma: se a entrada for, por exemplo, a frase: "*I am fine now thank you*", a resposta pode ser algo como: "*How long have you been fine now thank ?*". Quando a entrada apresenta complementos além de sua estrutura normal, o *chatterbot* tende a dizer frases sem sentido [Leonhardt 2003].

O trabalho realizado por Weizenbaum (1966) contribui muito para ciência e estudos relacionados a construção de *chatterbots*, existem outros trabalhos realizados que são baseados nos princípios de ELIZA. Existe um site na internet que permite o diálogo com esse *chatterbot* <http://www-ai.ijs.si/eliza-cgi-bin/eliza_script>.

Alice

A linguagem AIML foi desenvolvida pela comunidade Alicebot durante o período de 1995-2000, com uma gramática particular formava a base para o primeiro Alicebot. Após diversas atualizações visando à padronização da gramática foi adotado o XML (eX*tensible Markup Language*) como modelo da linguagem [Leonhardt 2003].

ALICE ganhou as versões do *Loebner Prize* de 2000, 2001 e 2004, uma competição anual criada por Hugh Loebner para incentivar o desenvolvimento da inteligência artificial. O objetivo da competição é criar um simulador de diálogo que seja mais humano possível em suas respostas, usando como medida o teste de Turing. Na edição de 2001, ALICE conseguiu obter de um dos juízes uma nota mais alta do que

um humano, fato inédito para o evento. Sua base de conhecimento é composta por mais de vinte mil categorias de sentenças de entrada (similar às regras utilizadas por ELIZA), podendo obter índices de coerência na conversação bem superiores aos de seus concorrentes. Por outro lado, a medida que essa base de conhecimento aumenta, torna-se mais difícil mantê-la, especialmente se for levada em consideração a sua reutilização na construção de outros *chatterbots* [Galvão 2003]. A seguir, na Figura 2, é apresentado um trecho de diálogo de ALICE, usado no *Loebner Prize* de 2001 [Loebner 2011].

juiz	My name is judge02.
ALICE	I AM ALWAYS GLAD TO MAKE NEW FRIENDS, JUDGE02.
juiz	Hi, pleased to meet you?
ALICE	HI THERE JUDGE02. NICE MEETING YOU TOO JUDGE02.
juiz	May I ask you some questions?
ALICE	SURE ASK ME ANYTHING.
juiz	Who composed Beethoven's piano concerto?
ALICE	CHARLIE PARKER. :-)
juiz	Excellent. Can you name another composer of music for me?
ALICE	THANKS FOR THE COMPLIMENT.
ALICE	LINUS TORVALDS. :-)
...	

Figura 2. Trecho de diálogo com ALICE

O *chatterbot* ALICE está disponível para os usuários em <http://alice.pandorabots.com>.

Elektra

Elektra utiliza os mesmo princípios de ALICE, foi elaborado na UFRGS (Universidade Federal do Rio Grande do Sul), o c*hatterbot* denominado de Profª Elektra foi projetado inicialmente visando apenas em responder algumas perguntas de Física para os alunos do ensino secundário, servindo de treinamento para o vestibular, foi disponível para os usuários da internet em 2002 Leonhardt (2003). Assim como ALICE, Elektra implementa as funcionalidades da linguagem AIML. Na Figura 3, segue uma imagem com um trecho de um diálogo com o *chatterbot* Elektra.

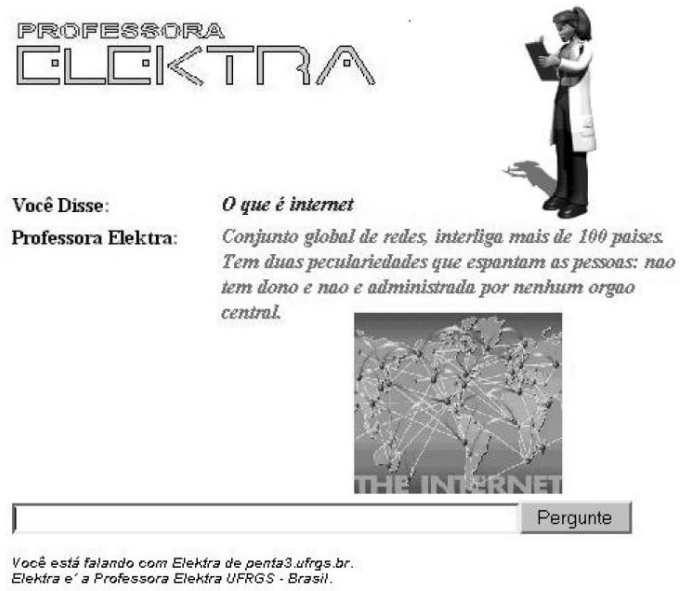


Figura 3. Trecho de diálogo com o *chatterbot* Elektra

ARQUITETURA DO PROJETO

O projeto desenvolvido utiliza os princípios da linguagem AIML, podendo dizer que o simulador de diálogo elaborado é uma derivação do projeto ALICE. Não foi implementado todas as funções da linguagem AIML, apenas as funções que atendiam o propósito do projeto, além disso, foi necessário adicionar uma nova marcação na linguagem AIML para prover o acesso a uma base de conhecimento externa, um banco de dados.

O simulador de diálogo foi desenvolvido na linguagem de programação C# [Deitel 2003], o banco de dados utilizado no projeto foi o MS SQLServer, por ter uma boa integração com a linguagem utilizada.

Especificação da linguagem AIML usada no projeto

Category

A marcação de categoria <category> é o maior nível da estrutura. Toda categoria tem como obrigatoriedade ter como filhos os

elementos: padrão <pattern> e um modelo de resposta <template>. Um exemplo de uso dessa marcação é apresentado na Figura 4.

Exemplo de Categoria:
<category> <pattern> Você gosta de música? </pattern> <template> Gosto muito! Qual sua banda preferida? </template> **<category>**
Diálogo:
Usuário: Você gosta de música? **BOT**: Gosto muito! Qual sua banda preferida?

Figura 4. Exemplo do uso da marcação <category>

Padrão

Um padrão <pattern> deve ser abrigatório em uma categoria, e sempre com o primeiro filho, além de não conter nenhum atributo, esta marcação contém uma expressão utilizada como referência para o retorno de uma resposta (Figura 5).

Exemplo de Padrão:
<pattern> <!-- Expressão padrão --> </pattern>

Figura 5. Uso da marcação <pattern>

Template

Para vincular um modelo de resposta a uma expressão padrão <pattern>, deve-se inserir a expressão no elemento <template> contido na marcação de categoria <category>, quando o interpretador encontrar o padrão será retornado o modelo <template>. A Figura 6 mostra um exemplo dessa marcação.

Exemplo de Modelo:
<template> <!-- Expressão do modelo --> </template>

Figura 6. Uso da marcação <template>

3.1.4. Star

Esta função cujo exemplo está na Figura 7, pode ser utilizada por meio de uma marcação de texto '*:star*' e tem como propósito substituir a marcação por um conteúdo "coringa" marcado por um asterisco ' * ' na expressão do padrão <pattern>.

Exemplo tag star:
<pattern> Meu nome é *. </pattern> <template> Muito prazer **:star** ! </template>
Diálogo:
Usuário: Meu nome é João. **BOT**: Muito prazer João!

Figura 7. Exemplo de utilização da função: star

Set e Get

A *tag* <set> (Figura 8) permite criar ou alterar uma variável na memória, estas variáveis são temporárias, a cada acesso essas variáveis são reiniciadas. Para utilizar esse mecanismo é necessário adicionar dois elementos na marcação <set>, são eles: <name> e <value>, no primeiro é necessário especificar o nome da variável que deseja criar ou alterar, no outro elemento armazenará o valor da variável.

Para acessar o valor de uma variável, deve-se utilizar a marcação *+|nome da variável|+* especificando apenas o nome da variável, a *tag* <set> ocorre sempre como filho da marcação <category>.

Exemplo de armazenamento temporário:
<pattern> Meu nome é *. </pattern> **<set>** <name>nome do usuario</name> <value>:star</value> **</set>** <template> Muito prazer **+|nome do usuario|+**, meu nome é BOT </template>
Diálogo:
Usuário: Meu nome é João. **BOT**: Muito prazer João, meu nome é BOT.

Figura 8. Exemplo de utilização da marcação <set>

Srai

Esta marcação é utilizada como filho de uma marcação <template>, sua função é obter a resposta de outro padrão que corresponda ao mesmo padrão encontrado através de um desvio direcionado. A Figura 9 mostra um exemplo dessa marcação.

Exemplo de desvio direcionado:
<category> <pattern> Olá </pattern> <template> Olá, como vai você? </template> </category> <category> <pattern> Oi </pattern> <template> **<srai>**Olá**</srai>** </template> </category>
Diálogo:
Usuário: Oi. **BOT**: Olá, como vai você?

Figura 9. Exemplo de utilização da marcação <srai>

Sql

Esta marcação foi adicionada às marcações padrões da linguagem AIML para permitir a execução de consultas em um banco de dados pré-determinado à aplicação, chamado de sql <sql> armazena a consulta SQL.

A marcação <sql> é permitida somente dentro da *tag* <template> ou <li>, o resultado da consulta efetuada pode ser um único resultado ou uma lista contendo vários resultados. Caso for uma lista, serão exibidos os resultados separados por quebra de linha, senão apenas um único resultado. Para definir como o resultado deve ser exibido, é necessário utilizar a marcação <mask> responsável por detalhar o modo de exibição da consulta SQL. Na Figura 10 é apresentado um exemplo do uso desse mecanismo.

Exemplo utilizando consulta SQL:
<category> <pattern> Você sabe qual a duração do curso de Química? </pattern> <template> **<sql>** SELECT **cur_duracao** FROM curso c WHERE c.nome=”quimica” <mask> Sei sim, a duração é de *+*/**cur_duracao**/*+* anos.</mask> **</sql>** </template> </category>
Diálogo: Resultado do SQL = “4”
Usuário: Você sabe qual a duração do curso de Química? **BOT:** Sei sim, a duração é de 4 anos.

Figura 10. Exemplo de utilização da marcação <sql>

Random

A marcação *random* permite que o modelo de uma resposta possa variar de modo randômico. Pode-se observar no exemplo da Figura 11 duas respostas possíveis para um mesmo padrão.

Exemplo de resposta aleatória:
<category> <pattern> Oi </pattern> <template> **<random>** Oi, tudo bem?   - Olá, como vai?   **</random>** </template> </category>
Diálogo:
Usuário: Oi **BOT:** Oi, tudo bem?
Diálogo:
Usuário: Oi **BOT:** Olá, como vai?

Figura 11. Exemplo de utilização da marcação <random>

Interpretador

O interpretador possui funcionalidades para manipular documentos AIML, que são a base de conhecimento do simulador de diálogo. O interpretador utiliza funções que são usadas para manipular arquivos XML, foram utilizados recursos da própria linguagem C#. A Figura 12 apresenta um diagrama da classe *AimlReader* com os principais atributos e métodos (a) e um diagrama funcional (b) referentes ao interpretador da linguagem AIML.

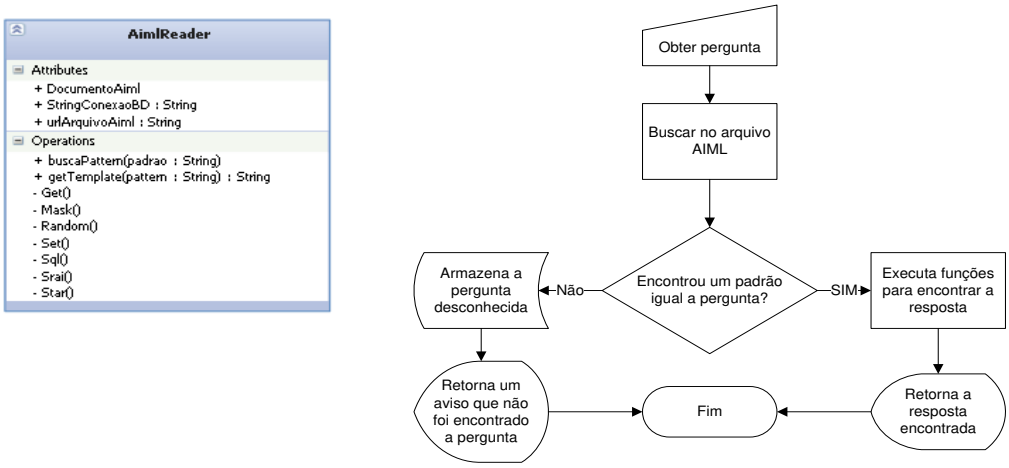


Figura 12. (a) Diagrama de classe do interpretador (b) Diagrama funcional

Para que o interpretador possa funcionar, ele deve conhecer caminho absoluto do arquivo AIML e também os dados para efetuar a conexão com o banco de dados, como: local da base de dados, usuário, senha e o nome do banco de dados. A classe *AimlReader* tem um método responsável por retornar o modelo de resposta referente a um padrão (pergunta) que é recebido por meio do módulo Web (seção 3.3). Este método chamado de *getTemplate* tem um parâmetro chamado *pattern* do tipo *string,* quando esse método é acionado, é feito uma busca no arquivo AIML para encontrar um padrão que corresponde ao padrão solicitado. Um padrão quando não encontrado no arquivo AIML, é armazenado em um arquivo que se encontra no mesmo diretório da base conhecimento AIML, e o interpretador retornará uma reposta informando que a pergunta solicitada pelo usuário é desconhecida e que vai ser armazenada para que o administrador defina uma resposta adequada para ela. Este método *getTemplate* agrega as funções para implementar os mecanismos da linguagem AIML.

Uma das funções do simulador de diálogo desenvolvido além de processar todas as funções

da linguagem AIML implementadas neste trabalho é ser responsável por executar consultas no banco de dados, para isso é criada uma nova conexão entre o interpretador e o banco de dados. A consulta contida na *tag* <sql> (seção 3.1.7) é executada e o resultado da consulta é modelado de acordo com o conteúdo definido na *tag* <mask>. A resposta depois de modelada é retornada em forma de texto.

Módulo Web

Este módulo é uma aplicação Web que contém na interface uma caixa de texto onde o usuário digitará a mensagem a ser enviada, e outra que exibe o diálogo provido pelo usuário e o *chatterbot,* para que usuário possa visualizar a conversa. Esta aplicação tem o objetivo de interagir com o usuário final. Na Figura 13 é ilustrada como a aplicação Web atua.

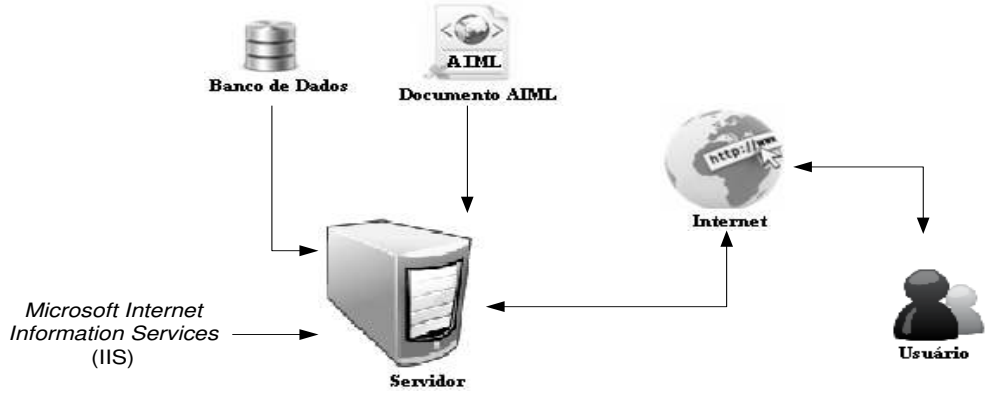


Figura 13. Aplicação Web

A base de conhecimento AIML fica armazenada junto com a aplicação Web no servidor*,* que utiliza o interpretador para manipular a base de conhecimento e efetuar as buscas para os possíveis padrões (perguntas) que serão solicitados pelo usuário. O interpretador é instanciado uma única vez, para que o mesmo seja compartilhado, melhorando o desempenho da aplicação Web*,* já que para o interpretador carregar a base de conhecimento AIML na memória existe um custo computacional considerável. Para funcionar o mecanismo da linguagem AIML que permite manipular dados temporários, (seção 3.1.5) são utilizadas as variáveis de sessão da página Web.

Toda vez que o usuário enviar uma pergunta para aplicação Web, o texto é tratado para que todas as letras fiquem minúsculas e para que sejam retirados todos os acentos, não é tratado a semântica do texto enviado.

Módulo desktop

Este módulo foi elaborado para interagir somente com a pessoa responsável por gerenciar a base de conhecimento do simulador de diálogo, por meio desta aplicação é possível criar novos padrões para a base de conhecimento AIML. Esta aplicação oferece um recurso que serve para auxiliar no aprendizado do simulador de diálogo, o administrador da aplicação pode visualizar e definir respostas para os padrões desconhecidos que foram solicitados pelos usuários da aplicação Web (seção 3.3), e que foram armazenados por falta de respostas.

Todo padrão antes de ser armazenado, passa por um tratamento para que não sejam armazenados na base de conhecimento padrões que contenham letras maiúsculas e acentos.

CONCLUSÃO

Toda implementação do simulador de diálogo fica voltada para utilizar os padrões propostos pela linguagem AIML, alguns projetos tem como diferencial funções que normalmente são adaptadas para atender as necessidades do ambiente ou do propósito elaborado para o mesmo. O simulador de diálogo desenvolvido neste trabalho diferencia-se dos demais por prover uma comunicação entre a base de conhecimento AIML com outra base de dados específica (banco de dados), proporcionando uma melhor dinâmica no retorno das respostas relacionadas ao assunto proposto.

As principais vantagens de utilizar essa adaptação da linguagem AIML utilizando um banco de dados é oferecer resultados atualizados, deixando as respostas sincronizadas com a base de dados escolhida, melhorando um dos problemas que são enfrentados por simuladores de diálogos, que é a manutenção da informação, ou seja, manter as respostas atualizadas. Existem respostas que não necessitam de atualizações como, por exemplo: “Oi, tudo bem?” resposta: ”Tudo bem, e você?”, por outro lado, existem respostas que não são fixas, podendo sofrer alterações constantemente, por exemplo: “Quem é o professor de matemática?”, resposta: “É o nosso amigo Bob”. Nesse segundo exemplo a resposta pode não ser a mesma em tempos diferentes. Para atualizar todas as respostas referentes em uma base de conhecimento AIML, faz-se necessário procurar por todos os padrões que utilizam o resultado desatualizado para poder ser atualizados, acarretando em muita mão de obra ao administrador da aplicação, que dependendo do tamanho desta base ou da quantidade de padrões que levam a esse resultado, torna-se inviável a manutenção das informações. Em muitos casos o desempenho das buscas para obtenção da resposta pode ser afetado dependendo do tamanho da base de dados, por outro lado, quanto maior a base de dados, há garantia de melhores resultados, pois a base de conhecimento é mais ampla, aumentando as chances de ter um padrão correspondente ao requisitado.

REFERÊNCIAS

ALICE (1995), The A.L.I.C.E Foundation. Disponível em: http://alicebot.org, Acesso em: 25 set. de 2010.

DEITEL, H. M., DEITEL, P. J., LISTFIELD, J., NIETO, T. R., YAEGER, C., ZLATKINA, M., C# **-** Como Programar. São Paulo: Pearson Education, 1° ed, 2003.

GALVÃO, A. M. (2003) “Persona-AIML: uma arquitetura para desenvolver c*hatterbot*s com personalidade”. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Pernambuco, Recife, Pernambuco.

LEONHARDT, Michelle D., CASTRO, Daiane D., DUTRA, Renato L. S., TAROUCO, Liane M. R. (2003) “ELEKTRA: Um *Chatterbot* para Uso em Ambiente Educacional”. Disponível em: http://tecnociencia.inf.br/comunidade/index.php?option=com_content&task=view&id=120&Itemid=41, Acesso em: 25 de set. de 2010.

LOEBNER, H., Loebner Prize Contest. Disponível em: http://www.loebner.net, Acesso em: 26 jun. de 2011.

WEIZENBAUM, J. (1966) “Eliza - A Computer Program for the Study of Natural Language Communication Between Man and Machine”, Disponível em: http://i5.nyu.edu/~mm64/x52.9265/january1966.html, Acesso em: 26 set. de 2010.