



Pós-Graduação em Ciência da Computação

**“UM CHATTEBOT PARA AQUISIÇÃO  
AUTOMÁTICA DE PERFIL DO USUÁRIO”**

**Por**

***Thiago José Marques Moura***

**Dissertação de Mestrado**



Universidade Federal de Pernambuco  
posgraduacao@cin.ufpe.br  
www.cin.ufpe.br/~posgraduacao

RECIFE, 03/2003



Universidade Federal de Pernambuco

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**Thiago José Marques Moura**

## **“UM CHATTERBOT PARA AQUISIÇÃO AUTOMÁTICA DE PERFIL DO USUÁRIO”**

*ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA BTEÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.*

ORIENTADOR(A): Flávia de Almeida Barros

RECIFE, Março/2003

*aos meus pais*

# Agradecimentos

---

A realização desta dissertação não seria possível sem o apoio de algumas pessoas em especial. Portanto, gostaria de agradecer às seguintes pessoas:

- aos meus pais Hermano e Fátima por sempre me incentivarem em meus estudos e terem dado todo o apoio necessário;
- a Alana Araújo Rodrigues por estar sempre presente em minha vida em todos os momentos que precisei de companhia e conforto;
- a Flávia de Almeida Barros, pela paciência, compreensão e dedicação durante toda a fase de orientação;
- aos meus colegas de pós-graduação sempre presentes durante toda a fase do mestrado.

## Resumo

---

Com a crescente expansão da Internet, muitas empresas passaram a buscar formas mais naturais de comunicação com os usuários de seus Websites. Neste cenário, o uso de chatterbots (sistemas que se comunicam em linguagem natural) tem crescido muito, podendo assumir diversas funções (e.g., conversar sobre temas livres ou específicos, prover acesso a bancos de dados da empresa, funcionar como um FAQ automático). Chatterbots podem também ser usados para aquisição de informação relevante sobre usuários, tendo como objetivo montar o perfil de cada usuário de determinado Website. Esses perfis poderão ser usados para os mais diversos fins (e.g., oferecer produtos de acordo com as preferências do usuário, adaptar o Website da empresa para as necessidades de seus usuários, entre outros).

Este trabalho apresenta um chatterbot capaz de prover acesso em português a uma base de dados, realizando aquisição automática do perfil de seus usuários a partir dos diálogos travados. Este agente de conversação será acoplado ao sistema GEP (Guia de Entretenimento Personalizado), que tem por objetivo fazer recomendação automática de filmes a partir da sua base de dados e dos perfis dos seus usuários. Os perfis automaticamente montados pelo chatterbot são usados pelo GEP na escolha de filmes que serão indicados para cada usuário através de diálogos em português. O chatterbot foi implementado como uma extensão da linguagem AIML (Artificial Intelligence Markup Language), uma das mais bem sucedidas abordagens para construção de chatterbots no momento. Testes realizados com o protótipo revelaram um bom desempenho do chatterbot nos diálogos e na aquisição dos perfis. Trata-se de um trabalho original, uma vez que não foi encontrado na Web nenhum outro agente de conversação capaz de fazer aquisição automática de perfil de usuário.

# Abstract

---

With the growth of the Internet, many companies began to search for a more natural way to communicate with its Website users. In this scenario, the use of chatterbots (systems that communicate in natural language) has continuously grown. These robots are used for different purposes, such as to maintain dialogues about specific or free topics, to provide access to databases, or as a kind of automatic FAQ. These bots can also be used to acquire the profiles of users of a particular Website. The profiles can be used for different purposes, such as to offer products according to the user's preferences, to adapt the Website to its users needs, among others.

This work presents a chatterbot which provides access to a database via dialogues in Portuguese with the main aim of acquiring the profiles of the system's users. This bot will be integrated into the GEP (Guia de Entretenimento Personalizado) system, whose main aim is to recommend films to its users based on their profiles. The profiles are automatically created and update by the chatterbot through dialogues in Portuguese. The chatterbot was implemented in AIML (Artificial Intelligence Markup Language), currently the best approach for the construction of dialogue systems. Test with the prototype revealed a satisfactory performance in he profiles acquisition task. This is an original work, since no similar system was found in the available literature on in the Web.

# Sumário

---

<b>1 Introdução</b> .....	<b>8</b>
<b>1.1 Trabalho Proposto</b> .....	<b>9</b>
<b>1.2 Organização da Dissertação</b> .....	<b>10</b>
<b>2 Chatterbots</b> .....	<b>12</b>
<b>2.1 Breve História dos <i>Chatterbots</i></b> .....	<b>13</b>
2.1.1 Primeira Geração – Casamento de padrão e uso de regras gramaticais.....	14
2.1.2 Segunda Geração – Técnicas de Inteligência Artificial .....	16
2.1.3 Terceira Geração – Linguagens de Marcação.....	18
<b>2.2 Tipo de <i>Chatterbots</i> encontrados na Web</b> .....	<b>19</b>
2.2.1 Chatterbots de Entretenimento.....	20
2.2.2 Chatterbots de Ajuda.....	20
2.2.3 Chatterbots Comerciais de Suporte ao Consumidor .....	21
2.2.4 Chatterbots Comerciais de Marketing.....	21
2.2.5 Chatterbots sem Classificação .....	22
<b>2.3 A Linguagem AIML</b> .....	<b>23</b>
2.3.1 Princípios do AIML .....	24
<b>Tag <i>srai</i></b> .....	<b>25</b>
2.3.2 Arquitetura de <i>chatterbots</i> AIML .....	28
<b>2.4 Conclusões</b> .....	<b>29</b>
<b>3.1 Engenharia do conhecimento</b> .....	<b>31</b>
<b>3.2 Escopo do Conhecimento</b> .....	<b>33</b>
Fontes de Conhecimento.....	34
Categorias de Conhecimento .....	34
<b>3.3 Aquisição de Conhecimento</b> .....	<b>35</b>
Dificuldades na Aquisição de Conhecimento.....	35
Processos de Aquisição de Conhecimento.....	37
Métodos de Aquisição de Conhecimento .....	38
<b>3.4 Métodos Manuais</b> .....	<b>40</b>
3.4.1 Entrevistas.....	40
Entrevistas não-estruturadas .....	40
Entrevistas Estruturadas.....	41
3.4.2 Métodos de Rastreamento.....	42
3.4.3 Observações e Outros Métodos Manuais.....	42
Observações .....	42
Outros Métodos Manuais.....	43
<b>3.5 Métodos Guiados por Especialistas</b> .....	<b>44</b>
3.5.1 Método Manual: Relatos Próprios dos Especialistas .....	45
3.5.2 Aproximações com auxílio do computador .....	46
<b>3.6 Métodos Semi-Automáticos</b> .....	<b>47</b>
3.6.1 Análise de Repertório de Grade (Repertory Grid Analysis).....	47
3.6.2 Apoio ao Engenheiro de Conhecimento .....	49

<b>3.7 Método Automático - Indução .....</b>	<b>51</b>
3.7.1 Indução de Regra Automática.....	52
3.7.2 Indução Interativa .....	53
<b>3.8 Adquirindo conhecimento .....</b>	<b>54</b>
Selecionando um Método Apropriado .....	54
Validação e Verificação da Base de Conhecimento .....	55
Analisando, Codificando, Documentando e fazendo a Diagramação.....	55
<b>3.9 Tabela Comparativa Entre os Métodos de Aquisição .....</b>	<b>56</b>
<b>3.10 Conclusão.....</b>	<b>57</b>
<b>4 Um Chatterbot em AIML para aquisição automática de perfil .....</b>	<b>59</b>
<b>4.1 Arquitetura do Sistema .....</b>	<b>60</b>
4.1.1 A Interface do sistema .....	61
4.1.2 A Base de Dados do Domínio.....	62
4.1.3 A Base AIML ampliada .....	62
4.1.4 O Interpretador AIML.....	64
<b>4.2 O Protótipo .....</b>	<b>65</b>
4.2.1 A interface.....	65
4.2.2 A Base de Dados sobre Filmes .....	66
4.2.3 A Base AIML ampliada .....	66
4.2.4 O Interpretador AIML.....	67
<b>4.3 Tags criadas.....</b>	<b>69</b>
Tag BuscaFilme .....	69
Tag Sinopse.....	72
Tag Diretor.....	73
Tag Ator .....	74
Tag NotaDada .....	75
<b>4.4 Testes e exemplo de uso do chatterbot .....</b>	<b>76</b>
4.4.1 Iniciando o diálogo com o Robô.....	77
4.4.2 O Procesamento dos diálogos .....	78
4.4.3 O Banco de Dados .....	80
<b>4.5 Resultados obtidos .....</b>	<b>81</b>
<b>4.6 Conclusões .....</b>	<b>83</b>
<b>5 Conclusão.....</b>	<b>85</b>
<b>5.1 Contribuições.....</b>	<b>85</b>
<b>5.2 Dificuldades Encontradas .....</b>	<b>86</b>
<b>5.3 Trabalhos Futuros .....</b>	<b>86</b>
<b>Referências Bibliográficas.....</b>	<b>89</b>
<b>Apêndice A.....</b>	<b>92</b>
<b>Apêndice B.....</b>	<b>117</b>



## Lista de Figuras

---

Figura 2.1: Diálogo realizado com ELIZA.....	15
Figura 2.2: Diálogo entre o médico e PARRY.....	16
Figura 2.3: Diálogo entre JULIA e um juiz.....	17
Figura 2.4: Diálogo entre BARRY e um usuário humano.....	18
Figura 2.5: Base de uma categoria em AIML.....	24
Figura 2.6: Exemplo do uso do símbolo “*”.....	25
Figura 2.7: Exemplo de uma categoria escrita com a tag <code>&lt;srai&gt;&lt;/srai&gt;</code> .....	26
Figura 2.8: Exemplo do uso de tags <code>&lt;get name="xxx"/&gt;</code> e <code>&lt;set name="xxx"&gt;&lt;/set&gt;</code> ....	27
Figura 2.9: Exemplo do uso da tag <code>&lt;that&gt;&lt;/that&gt;</code> .....	28
Figura 2.10: Arquitetura de um <i>chatterbot</i> AIML.....	29
Figura 3.1: Passos da construção de um sistema baseado em conhecimento.....	33
Figura 3.2: Tipos de Conhecimento a serem representados na base de conhecimento....	33
Figura 3.3: Estágios da Aquisição do Conhecimento.....	38
Figura 3.4: Métodos Manuais.....	39
Figura 3.5: Aquisição de Conhecimento guiada pelo especialista.....	39
Figura 3.6: Indução.....	40
Figura 3.7: Um exemplo de Grade.....	48
Figura 3.8: Função do Engenheiro de Conhecimento em uma Aquisição de Conhecimento Interativa.....	50
Figura 3.9: Tabela Comparativa entre os métodos de aquisição de conhecimento.....	57
Figura 4.1: Arquitetura do sistema.....	61
Figura 4.2: Arquivo de categoria “cumprimentar o robot”.....	63
Figura 4.3: Arquivo de propriedades.....	64
Figura 4.4: Exemplo de funcionamento do interpretador.....	65
Figura 4.5: Exemplo de tag não definida.....	68
Figura 4.6: Exemplo de sentença com estilo de filme isolado.....	70
Figura 4.7: Arquivo de categoria: “filme ação”.....	70
Figura 4.8: Arquivo de categoria: “estilos de filmes”.....	71
Figura 4.9: Arquivo de categoria: “sinopse de filme”.....	73
Figura 4.10: Arquivo de categoria: “diretor de filme”.....	74
Figura 4.11: Arquivo de categoria: “ator de filme”.....	75
Figura 4.12: Arquivo de categoria: “nota dada 3”.....	76
Figura 4.13: Tela de entrada do sistema.....	77
Figura 4.14: Tela de Diálogos.....	78
Figura 4.15: Diálogo entre o robô e um usuário.....	79
Figura 4.16: Tabelas do Banco de dados.....	81

# 1 Introdução

---

O grande crescimento da Internet, em particular, da Web, reavivou estudos e pesquisas em áreas que haviam sido um pouco esquecidas, devido à falta de um ambiente propício para o seu desenvolvimento. Este é o caso dos *chatbots* [Laven 2001], sistemas que se comunicam com os usuários em linguagem natural. Atualmente, esses sistemas são utilizados por milhares de usuários na Web para os mais diversos fins (e.g., participar de salas de *chat* e RPG, acompanhar estudantes em ambientes de educação a distância, tirar dúvidas sobre um domínio específico, representar empresas, oferecer suporte técnico, entre outros).

A indústria do entretenimento, em particular, está em permanente crescimento, e com ela crescem também as pesquisas que buscam soluções para os seus maiores problemas. Um desses problemas é a prestação ininterrupta de serviços a usuários variados. Os *chatbots* têm sido utilizados nesses ambientes como assistentes digitais capazes de dialogar com usuários a todo momento, buscando assim substituir assistentes humanos, que são escassos e custam caro às empresas. Além disso, como esses sistemas dialogam em linguagem natural, eles são simples de serem utilizados por pessoas de todas as idades, com um alto ou baixo grau de conhecimento sobre Web e Informática.

Outra área em franca expansão na Web está relacionada aos serviços de recuperação de informação [Baeza-Yates & Ribeiro-Neto 1999]. Organizar, manter e recuperar informação na Web são serviços que concentram grande parte das pesquisas realizadas na área. Os *chatbots* podem ser utilizados para facilitar essas tarefas, auxiliando os usuários a consultar esses sistemas através de diálogos em linguagem natural.

Apesar da larga utilização de *chatbots* na Web, ainda existem muitas lacunas a serem preenchidas, tanto em relação às suas aplicações como às técnicas utilizadas para a construção desses sistemas.

## 1.1 Trabalho Proposto

O trabalho aqui descrito teve como objetivos principais desenvolver um *chatterbot* capaz de manter diálogos sobre um domínio específico, e realizar aquisição automática de perfil dos usuários do sistema nesse domínio. A aquisição automática do perfil se dá com base nos diálogos travados entre o agente e o usuário. Durante esses diálogos, o sistema busca identificar informações relevantes nas sentenças do usuário para montar e atualizar o seu perfil, que poderá ser usado por sistemas de entretenimento e de recomendação personalizada que venham a ser acoplados ao *chatterbot*.

O sistema foi construído de forma modular, extensível, portátil e fácil de ser usado. A arquitetura do *chatterbot* consta de 5 módulos: a interface em linguagem natural, um banco de dados sobre o domínio escolhido para a aplicação (que inclui também os dados sobre o perfil dos usuários), a base de padrões de diálogos, o interpretador de diálogos, e os arquivos de log do sistema.

A base de diálogos do *chatterbot* foi implementada em AIML (Artificial Intelligence Markup Language) [Wallace 2001], a tecnologia mais moderna e bem sucedida para construção desse tipo de sistema. Os padrões de diálogos foram adaptados e ampliados a partir de um trabalho desenvolvido pelo Prof. André Neves<sup>1</sup>, que desenvolve um *chatterbot* para a língua portuguesa.

O interpretador de diálogos processa as sentenças digitadas pelos usuários, e monta a resposta que será devolvida pelo *chatterbot*. O interpretador utilizado aqui estende o que está disponível para *download* no site dos criadores do AIML<sup>2</sup>. As principais modificações realizadas se referem ao processamento dos novos padrões de diálogos criados, alguns deles necessitando consultar o banco de dados do domínio específico para então montar a resposta do *chatterbot*. Este módulo foi implementado em Java, e é um sistema independente que executa na Web através de qualquer *browser* disponível.

O domínio específico escolhido para este projeto foi o de filmes, por tratar-se de um tema popular que atrai o interesse dos mais variados usuários na Web. De fato, o

---

<sup>1</sup> Coordenador do laboratório de hipermídia da UFPE e aluno de doutorado do CIn-UFPE.

<sup>2</sup> <http://www.alicebot.org>

interesse por entretenimento atinge usuários de todas as idades, tendo esses sistema centenas de adeptos em potencial. Utilizamos aqui uma base de dados cedida pelos desenvolvedores do GEP (Guia de Entretenimento Personalizado) [Bezerra et al. 2002]. Trata-se de um sistema de recomendação que mantém uma base de filmes e o perfil dos usuários cadastrados, utilizando essas informações para fornecer recomendações personalizadas aos seus usuários.

Atualmente, o GEP se comunica com seus usuários através de menus, tanto para realizar a aquisição de perfil quanto para realizar sessões de recomendação. A integração do *chatbot* desenvolvido ao GEP terá como principal contribuição a facilitação da interação entre usuário e sistema, tornando mais naturais tanto a aquisição do perfil quanto a consulta à base de filmes do sistema.

Vários usuários tiveram acesso ao *chatbot*, e os *logs* desses diálogos foram utilizados em dois momentos: inicialmente, para que pudéssemos refinar o sistema, e, em seguida, para analisarmos seu desempenho. Foram utilizados cerca de 20 *logs* de diálogos de usuários variados com o sistema. O *chatbot* apresentou um bom desempenho de acordo com o propósito para o qual ele foi criado: aquisição de perfil. Cerca de 50% das interações realizadas dentro desse escopo forma bem sucedidas.

Vale ressaltar que não foi encontrado na literatura consultada nem na Web nenhum *chatbot* com os mesmos objetivos e características que o desenvolvido neste trabalho. Isto atesta a originalidade do trabalho.

## **1.2 Organização da Dissertação**

Além deste capítulo introdutório, esta dissertação de mestrado é composta de outros quatro capítulos, brevemente descritos a seguir:

### **Capítulo 2: Chatterbots**

Apresenta um breve histórico sobre *chatbots*, identificando gerações de acordo com as técnicas de implementação utilizadas em cada período. Apresenta também os tipos de *chatbots* encontrados na Web e a linguagem AIML, utilizada como base para a criação da base de diálogos do nosso sistema.

### **Capítulo 3: Aquisição do Conhecimento**

Descreve todo o processo de aquisição de conhecimento, com métodos de aquisição manuais, semi-automáticos e automáticos.

### **Capítulo 4: Um Chatterbot em AIML para aquisição automática de perfil**

Descreve em detalhes o trabalho realizado, incluindo o processo de construção do *chatterbot*, sua arquitetura, a implementação de cada módulo, e os testes e resultados obtidos.

### **Capítulo 5: Conclusões e Trabalhos Futuros**

Apresenta as dificuldades encontradas durante a elaboração deste projeto, e suas principais contribuições. Por fim, são propostos alguns trabalhos futuros.

## 2 Chatterbots

---

*Chatterbots* são sistemas de computador capazes de manter um diálogo com usuários através de linguagem natural.

Atualmente, muitas empresas se utilizam dessa tecnologia para os mais diversos fins. Muitos *chatterbots* são usados comercialmente (e.g., dando suporte técnico, representando alguma empresa na Internet, vendendo algum produto) e outros são usados com propósitos diferentes, como auxiliar pessoas em algum ambiente de estudo virtual, ou conversar em salas de *chats* sobre os mais diversos temas.

Os primeiros *chatterbots* tentavam imitar um ser humano conversando, de tal forma que não fosse possível distinguir se quem estava conversando era uma pessoa ou um sistema de computador.

A idéia de que poderia existir uma máquina no futuro capaz de agir como ser humano foi iniciada em 1921 por Karel Kapek em um jogo chamado “Robô Universal” e disseminada pelo escritor e contista russo Isaac Asimov em seu livro “Eu, Robô” [ESTADÃO 2002]. A palavra “robô” teve origem nesse mesmo jogo, derivada da palavra “robota”, que significa trabalho em tcheco. A palavra *bot* seria uma simplificação da palavra *robot* (robô em inglês). Seria um agente que trabalha a favor de um usuário ou para outro programa simulando uma atividade realizada por um ser humano.

Entretanto, o matemático e um dos criadores da ciência da computação, Alan Turing, foi responsável por primeiramente imaginar que um dia os computadores poderiam agir como humanos.

Ao longo dos anos, muitas técnicas foram usadas na criação de *chatterbots*. As primeiras técnicas baseavam-se em casamento de padrões e regras gramaticais. Depois foram utilizadas algumas técnicas de inteligência artificial, como as regras de produção e redes neurais, e por último, surgiram as linguagens de marcação para construção de bases de conhecimento [NEVES et al. 2002]. As linguagens de marcação vêm sendo amplamente utilizadas atualmente, com ótimos resultados. (Ver site do *Loebner Prize*)<sup>3</sup>

---

<sup>3</sup> <http://www.loebner.net/Prizef/loebner-prize.html>

Neste capítulo, veremos uma breve história dos *chatbots*, organizada de acordo com as técnicas utilizadas na construção desses sistemas. Esta classificação foi baseada em [Neves et. al. 2002]. Em seguida, apresentaremos algumas aplicações de *chatbots* para a Web.

Veremos também detalhes sobre o AIML, a linguagem de marcação para construção de *chatbots* que deu origem à mais nova geração desses sistemas.

## 2.1 Breve História dos *Chatbots*

Em 1950, Alan Turing propôs uma espécie de jogo no qual três participantes, uma mulher, um homem e um interrogador de qualquer sexo deveriam ficar em salas separadas para o interrogador conversar com o homem e a mulher através de terminais de computador. Ao término do diálogo, o interrogador deveria ser capaz de dizer quem era o homem e quem era a mulher [TURING 1950]. No seu artigo, Turing questiona a possibilidade de substituímos a mulher ou o homem por uma máquina, sendo o interrogador capaz de dizer quem é humano e quem é máquina.

Usando o jogo como uma metáfora, Turing discute uma das questões que mais intrigou a ciência da computação na segunda metade do século XX: "*Can machines think?*". Observando o jogo proposto, percebemos uma situação onde a máquina deveria imitar o homem, dificultando o trabalho do interrogador. Para Turing, esta situação é um indício de que o "pensamento" da máquina poderia ser considerado como uma "inteligência simulada", uma imitação de atitudes humanas que são consideradas "inteligentes".

Em 1990, o jogo proposto por Turing passou a ser conhecido como "*Turing Test*" e foi redesenhado por um grupo de pesquisadores do *Cambridge Center for Behavioural Studies*<sup>4</sup>, transformando-se em um evento de grandeza internacional, denominado *Loebner Prize* [LOEBNER 2001]. O evento avalia o desempenho de sistemas que conversam com os avaliadores do concurso em linguagem natural. Até hoje, nenhum sistema conseguiu receber o grande prêmio do concurso. O primeiro sistema a receber o

---

<sup>4</sup> <http://www.behavior.org/>

prêmio de US\$ 2.000,00 foi o OC Therapist, em 1991, criado por Joseph Weintraub da empresa “Thinking Systems Softwares”.

Observando os *Chatterbots* existentes no mercado, identificamos três gerações de robôs que diferem umas das outras no que concerne o paradigma tecnológico a partir do qual o sistema foi projetado e implementado.

### **2.1.1 Primeira Geração – Casamento de padrão e uso de regras gramaticais**

Anos após o artigo de Turing, a idéia de construir uma máquina capaz de conversar em linguagem natural foi tema de muitos dos mais importantes centros de pesquisa e desenvolvimento em ciências da computação no mundo. O primeiro trabalho de que se tem notícia na área foi ELIZA [WEIZENBAUM 1966]. Esse é um dos programas de Inteligência Artificial mais antigos, conhecidos e estudados no mundo todo.

O funcionamento de ELIZA é extremamente simples. Com apenas 204 linhas de código, ELIZA recebe uma sentença em linguagem natural, digitada por um usuário, compara essa sentença com uma lista de padrões, encontra o padrão mais semelhante àquela sentença digitada pelo usuário, analisa a sentença com base em regras gramaticais de decomposição, e monta respostas a partir de uma lista de *templates* com regras de reconstrução de frases associadas às regras de decomposição. Apesar de simples e de ser umas das implementações pioneiras, ELIZA tem uma personalidade muito bem definida.

Uma das características mais interessantes de ELIZA é a forma como ela formula suas respostas. ELIZA foi projetada para conversar como se fosse um psicanalista rogeriano, repetindo as frases dos pacientes, e conseguindo assim sua introspecção sem envolvimento de opiniões do psicanalista.

A figura 2.1 mostra um diálogo entre ELIZA e um usuário humano:



**Human:** Men are all alike.  
**Eliza:** In what way?  
**Human:** They're always bugging us about something or the other.  
**Eliza:** Can you think of specific example?  
**Human:** Well, my boyfriend made me come here.  
**Eliza:** Your boyfriend made you come here.  
**Human:** He says I'm depressed much of time.  
**Eliza:** I am sorry to hear you are depressed.  
**Human:** It's true. I am unhappy.  
**Eliza:** Do you think coming here will help you not be happy  
**Human:** I need some help, that much seems certain.  
**Eliza:** What would it mean to you if you got some help?  
**Human:** Perhaps I could learn to get along with my mother.  
**Eliza:** Tell me more about your family  
**Human:** My mother takes care of me.  
**Eliza:** Who else in your family takes care of you?  
**Human:** My father.  
**Eliza:** Your father.  
**Human:** You are like my father in some ways  
**Eliza:** What resemblance do you see?  
**Human:** You are not very aggressive but I think you didn't want me to notice that.  
**Eliza:** What makes you think I am not very aggressive?  
**Human:** You don't argue with me.  
**Eliza:** Why don't you think I don't argue with you?  
**Human:** You are afraid of me.  
**Eliza:** Does it please you to believe I am afraid of you?  
**Human:** My father is afraid of everybody.  
**Eliza:** What else comes to mind when you think of your father?  
**Human:** Bullies.  
**Eliza:** Does that have anything to do with the fact that your boyfriend made you come here?

**Figura 2.1:** Diálogo realizado com ELIZA

Muitos sistemas foram construídos com base nas técnicas utilizadas por ELIZA. Temos como exemplo o PARRY [PARRY 2002], um paciente paranóico que chegou a ser testado em uma variação do jogo proposto por Turing, onde um médico foi convidado a analisar três pacientes, um deles sendo o PARRY. Esses pacientes eram analisados através de terminais de computador e, ao final, o médico deveria identificar quem era de fato o robô. Nesse teste, o médico não conseguiu distinguir entre pacientes humanos e a máquina.

Na figura 2.2, temos um pequeno diálogo entre o médico e o “paciente” PARRY:

**Doctor:** How did you come to be in my hospital?  
**PARRY:** I am upset.  
**Doctor:** How long have you been in the hospital?  
**PARRY:** About a week.  
**Doctor:** In what way are you upset?  
**PARRY:** People make me nervous.  
**Doctor:** In what way do they make you nervous?  
**PARRY:** People give me a funny look sometimes.  
**Doctor:** Is there anything about people in particular?  
**PARRY:** Certain persons.  
**Doctor:** Who are the certain persons and what is it that they do?  
**PARRY:** Do you know how the underworld operates?

**Figura 2.2:** Diálogo entre o médico e PARRY

Apesar de, nessa geração, as técnicas utilizadas para criação dos *chatterbots* serem bastante simples, muitos outros robôs se destacaram, concorrendo e vencendo o *Loebner Prize*. THE PC THERAPIST, uma variação do ELIZA desenvolvido por Joseph Weintraub [WEINTRAUB 2002], venceu em 1991, 1992, 1993 e 1995, e HEX, desenvolvido por Jason Hutchens [HUTCHENS 1998], um professor da “*University of Western Australia*” venceu em 1996.

### 2.1.2 Segunda Geração – Técnicas de Inteligência Artificial

Mesmo com o grande sucesso dos *chatterbots* baseados em ELIZA, muitos cientistas da computação, não satisfeitos com os resultados do casamento de padrão, e envolvidos pela grande demanda na Internet, continuaram suas pesquisas utilizando técnicas de Inteligência Artificial para produzirem novos *chatterbots*.

Um primeiro exemplo de *chatterbot* desenvolvido com essas técnicas mais sofisticadas foi JULIA, desenvolvido por Michael Mauldin [MAULDIN 1999]. O objetivo de JULIA é auxiliar jogadores em um ambiente controlado, o TinyMUD (*Tiny Multi-User Dimension*), atuando como um personagem de um jogo que interage com qualquer outro usuário, seja ele real ou virtual.

Julia possui um modelo interno do mundo que pode ser atualizado no momento em que a conversa está se desenvolvendo. Ou seja, ela é capaz de contextualizar o diálogo, gerando assim uma maior sensação de realismo. JULIA também tem memórias de estados, que faz com que ela se lembre das últimas sentenças digitadas pelos usuários. No *Loebner Prize* de 1993, uma versão de JULIA foi classificada em terceiro lugar. Na figura 2.3 temos um trecho do diálogo entre JULIA e um juiz do *Loebner Prize* [LOEBNER 2001].

<p><b>Judge:</b> Have you ever kept a penthouse pet? <b>JULIA:</b> Do you have any pets? <b>Judge:</b> I think you're avoiding my question. <b>JULIA:</b> Do you think animals can think? <b>Judge:</b> Do you think it ethical for one species enslave another? <b>JULIA:</b> What kinds of pets did you have as a child? <b>Judge:</b> I'm asking the questions. What is a pet if it is not a slave of speciesism. <b>JULIA:</b> Oh, that's so sad. I think pets enrich a childhood immensely. <b>Judge:</b> How can we keep some animals as pets, yet eat others? <b>JULIA:</b> Guinea pigs are as smart as cucumbers. <b>Judge:</b> Would you eat a guinea pig? <b>JULIA:</b> I spent several summers on my Uncle's farm. I really enjoyed the animals a lot.</p>
---

**Figura 2.3:** Diálogo entre JULIA e um juiz

Foi desenvolvida também uma versão mais simples de JULIA, chamada de COLIN [FONER 1994], que tinha a intenção de tentar estimular as pesquisas nesta área em ambientes acadêmicos. COLIN é como uma criança, aprende uma linguagem do zero com o usuário. COLIN pode conversar em qualquer idioma, respondendo as sentenças do usuário com base em respostas previamente ditas pelo próprio usuário. O COLIN utiliza técnicas de redes neurais para aprender uma linguagem do zero.

Outro *chatterbot* que deve ser destacado é o MEGA-HAL, baseado em técnicas de redes neurais [HUTCHENS 1998]. MEGA-HAL possui um sistema bem sofisticado para reconhecer sentenças que são digitadas pelos usuários e definir as melhores respostas. Ele também tem capacidade de aprendizagem automática e foi construído com base nos modelos de MARKOV. Apesar de técnicas bem sofisticadas, o MEGAL-HAL só conseguiu a terceira colocação no *Loebner Prize* de 1998.

Com uma tecnologia chamada de JFRED (*Java Functional Responses Emulation Devices*) [ROBITRON 2001] é possível criar *chatterbots* independentes da

aplicação. O JFRED é uma ferramenta para o desenvolvimento de *chatbots*, onde o desenvolvedor pode construir uma representação do domínio e definir as regras para a escolha das respostas adequadas. É vendido em conjunto com o JFRED o JRL (JFRED Ruleset Language). Com esse kit, podem ser desenvolvidos *applets* customizados com a aplicação. Sua tecnologia é voltada para a Web e ele oferece uma interface de software em linguagem natural que possui independência de plataforma, servidor *multi-threaded*, aprendizado baseado em frames e independência de idioma. Também utiliza lógica fuzzy.

O *chatbot* chamado BARRY DEFACTO, baseado na tecnologia JFRED e desenvolvido por Robby Garner, venceu o *Loebner Prize* de 1998 e 1999 [LOEBNER 2001]. Na figura 2.4 temos um trecho do diálogo entre BARRY e um usuário humano.

**Human:** hi, do you know anything about the chinese tong involvement in heroin smuggling?  
**BARRY:** I should probably just keep my port shut...  
**Human:** can you go into any more detail?  
**BARRY:** Did they ever find that axe-murderer who was loose in New Jersey last year?  
**Human:** no, i don't think so!  
**BARRY:** Really? Why not?  
**Human:** I couldn't tell you. This is the first I've heard of it.  
**BARRY:** The College of New Jersey -- weren't they the ones convicted in that child porn case?

**Figura 2.4:** Diálogo entre BARRY e um usuário humano

Apesar de os *chatbots* dessa geração utilizarem técnicas de Inteligência Artificial complexas e bem elaboradas, não obtiveram necessariamente melhores resultados em seus diálogos em relação aos *bots* da primeira geração. Alguns *chatbots* da geração anterior, como é o caso do HEX [HEX 2002], que utiliza as mesmas técnicas de ELIZA, continuam apresentando resultados tão bons quanto os sistemas dessa geração [NEVES et al. 2002].

### 2.1.3 Terceira Geração – Linguagens de Marcação

No *Loebner Prize* do ano 2000, o Dr. Richard Wallace apresentou um sistema inovador, conhecido por A.L.I.C.E. (*Artificial Linguistic Internet Computer Entity*) [WALLACE 2001]. ALICE é uma interface em linguagem natural composta por módulos de conversação categorizando os usuários por idade, gênero, localização geográfica e

profissão. Ela ainda possui a capacidade de guardar o nome do usuário e o tópico da conversa, fazendo com que ela não mude de assunto aleatoriamente e consiga endereçar o usuário diretamente.

A principal novidade proposta por Wallace foi a utilização de uma linguagem de marcação (AIML – *Artificial Intelligence Markup Language*) baseada em XML (*eXtensible Markup Language*) para representar a base de conhecimento de ALICE. Essas linguagens de marcação têm como base uma meta-linguagem conhecida como SGML (*Standard Generalized Markup Language*). Nesse contexto, Wallace definiu o AIML como uma linguagem de marcação para bases de conhecimento de *chatterbots* [WALLACE 2001].

ALICE possui uma base de conhecimento de aproximadamente 25.000 categorias e um sistema de seleção de respostas escrito em Java que percorre a base a procura de uma categoria que atenda a cada sentença digitada pelo usuário.

Atualmente, a equipe de desenvolvimento do ALICE trabalha em novas pesquisas para melhorar seu desempenho, para que no futuro seja criado um *superbot* que utilize o conhecimento de todos os *chatterbots* AIML.

ALICE vem conseguindo resultados bem superiores em relação aos *chatterbots* da primeira e da segunda geração. No *Loebner Prize* de 2000, onde ela foi campeã, foi considerada como o “computador mais humano” por um grupo de juízes que era formado de lingüistas, filósofos e psicólogos.

Um bom exemplo de um *chatterbot* desenvolvido com base nas técnicas utilizadas por ALICE é o JOHN LENNON *Artificial Intelligence Project* [JOHN LENNON 2002]. As respostas desse *bot* são baseadas em conversas reais de John Lennon, e seu objetivo principal é a recriação de uma personalidade humana.

## **2.2 Tipo de *Chatterbots* encontrados na Web**

Vários tipos de *chatterbots* podem ser encontrados na Web, com diversos fins. Muitos têm algum propósito comercial, outros têm propósitos de entretenimento, e outros ainda podem servir de ajudantes virtuais, tendo como missão auxiliar as pessoas nas mais

diversas áreas do conhecimento. Ao longo dessa seção, veremos alguns tipos de *chatterbots* encontrados na Web.

### **2.2.1 Chatterbots de Entretenimento**

*Chatterbots* de entretenimento, como o próprio nome já diz, são robôs desenvolvidos com o intuito de divertir os usuários que os acessam. Esses sistemas, muitas vezes, são desenvolvidos para funcionar em salas de bate-papo (*chat*). Sistemas como esses devem ser dinâmicos e devem também simular o comportamento humano, para que os usuários que os acessam não se desinteressem após um diálogo curto.

ELIZA pode ser considerada um robô desse tipo, já que aqui também se encontram aqueles *chatterbots* que foram criados com o intuito de tentar estudar com mais precisão a complexidade na comunicação em linguagem natural entre robôs e seres humanos. Além de ELIZA, temos aqui também outros robôs da primeira geração como o PARRY e o THE PC THERAPIST.

De fato, nesta categoria encontramos a maioria dos *chatterbots* existentes. Aqui podemos citar também os seguintes robôs: ALICE, JULIA, BRIAN [BRIAN 2002], JABBERWACKY [JABBERWACKY 2002], CYBELLE [CYBELLE 2002], MEGA-HAL [MEGA-HAL 2002], ELOISA [ELOISA 2002] e Dr. ABUSE [DR. ABUSE 2002].

### **2.2.2 Chatterbots de Ajuda**

Os *chatterbots* de ajuda, também conhecidos como robôs de FAQ (*Frequently Asked Questions*), ou FAQbots, também são muito disseminados na Web. Esses robôs são responsáveis por responder as perguntas dos usuários baseados em suas bases de FAQs. Eles podem prover informações sobre empresas ou produtos, como também podem ser usados em ambientes de ensino virtual para tirar dúvidas dos alunos.

Robôs de ajuda se comportam como uma base de FAQ, permitindo que os usuários façam perguntas em linguagem natural e recebam respostas equivalentes, ou pelo menos similares, às respostas esperadas pelos usuários. Pode ser considerado um sistema de FAQ bem mais dinâmico, devido ao fato de os usuários que precisam de ajuda irem direto ao robô, ao invés de procurarem respostas para suas perguntas em uma base

de FAQ, tarefa bastante cansativa e demorada, uma vez que algumas são extensas e desorganizadas.

Nesses tipos de robôs, caso o sistema não conheça a resposta para alguma pergunta, é preferível que o robô mostre ignorância, ao invés de responder algo inadequado que possa trazer problemas ao usuário.

Podemos citar como exemplo o KLONE [KLONE 2002], um *chatbot* que foi criado para resolver alguns problemas dos usuários. Ele responde a algumas perguntas, mostra aos usuários tarefas a serem cumpridas para se chegar a um determinado objetivo, busca dados na Web e, além disso, ele representa os softwares comerciais da *Big Science Company*<sup>5</sup>.

### **2.2.3 Chatterbots Comercias de Suporte ao Consumidor**

Esses tipos de *bots* são muito semelhantes aos apresentados na seção anterior, com a diferença de que eles, além de fornecerem respostas adequadas, buscam também, com essas respostas, satisfazer o cliente da melhor forma possível. Para isso, é preciso que eles também façam perguntas aos clientes, para assim, baseados nas respostas dadas, conseguirem responder da melhor forma e resolver os problemas dos clientes.

Um exemplo bem interessante desse tipo de *chatbot* é o YODA (personagem do filme *Star Wars*), que pode ser encontrado no site da *Lucas Arts*<sup>6</sup>. O YODA é uma espécie de *Help Desk* para os produtos da empresa *Lucas Arts*, e dá suporte técnico aos usuários dos jogos desenvolvidos pela empresa. Tira qualquer tipo de dúvidas, desde instalação até dúvidas de como jogar.

### **2.2.4 Chatterbots Comercias de Marketing**

Encontramos também na Web os *chatbots* com o propósito de representar empresas e seus produtos. Eles possuem características tanto dos *chatbots* de ajuda como dos de entretenimento, porque além de representarem suas empresas e produtos,

---

<sup>5</sup> <http://www.bigscience.com>

<sup>6</sup> <http://support.lucasarts.com/yoda/start.htm>

eles também são preparados para tirar eventuais dúvidas dos seus usuários sobre os produtos ou a empresa.

Apesar de ser uma aplicação de interesse crescente, deve-se ter algum cuidado para que esses robôs não abusem da propaganda e, ao invés de atrair os usuários, acabem por afastá-los do *site* da empresa.

Um exemplo desse tipo de robô é o SE7E ZOOM, que pode ser encontrado no site da *Close-Up*<sup>7</sup>. A SE7E ZOOM conversa sobre qualquer assunto, e também tira dúvidas específicas sobre a empresa que representa. O usuário também pode ficar por dentro de algumas características de sua personalidade, em uma tabela que contém todos os dados e gostos do robô.

Um outro exemplo é o SHALLOWRED [SHALLOWRED 2002]. Foi criado pela *NeuroStudio* com o intuito de responder a questões básicas sobre a *NeuroMidia* e seus produtos. Sua construção foi baseada no *NeuroServer* da *NeuroMidia* e no sistema de autoria da *NeuroStudio*. Outro *chatterbot* desenvolvido pela *NeuroMidia* é a NICOLE [NICOLE 2002], que também foi desenvolvida para responder a questões básicas sobre os produtos da empresa.

### 2.2.5 Chatterbots sem Classificação

Temos ainda esta última classificação, para os *chatterbots* que não se enquadram em nenhum dos tipos anteriores. Esses robôs têm características bem específicas, ou não possuem um propósito específico, por isso não foram incluídos nas categorias anteriores.

O ULTRA-HAL [ZABAWARE 2002] pode ser considerado um *bot* dessa categoria. O ULTRA-HAL é uma ferramenta para a implementação de *chatterbots* da Zabaware. Os robôs criados por essa ferramenta podem aprender e até conseguem se comunicar através da fala. O ULTRA-HAL pode ser visto como uma espécie de assistente, possuindo uma memória interna que guarda números de telefone, endereços, e-mails, apontamentos, aniversários, ou qualquer outra informação. Ele pode

---

<sup>7</sup> <http://www.closeup.com.br/>



automaticamente discar números de telefone, abrir e-mails e até lembrar aos seus usuários de datas de aniversários automaticamente.

Outro exemplo que podemos encontrar é o CHAT (*Conversational Hypertext Access Technology*) [CHAT 2002]. Trata-se de um sistema de fácil acesso a informações eletrônicas, desenvolvidas pelo *Communications Research Center*<sup>8</sup>. Oferece uma interface em linguagem natural que é mais simples de usar do que um sistema baseado em palavras-chaves. Pode ser usado tanto por pessoas que sabem inglês, como por pessoas que tenham pouco conhecimento em informática. Existem outros sistemas desenvolvidos pelo mesmo centro, como o TIPS (vencedor do *Loebner Prize* de 1994) e EZL (*Easy Language Technology*).

O CHAT não é visto como um *chatterbot*, já que ele é apenas um sistema de diálogo em linguagem natural entre usuários humanos. Mas ele ajudou a acelerar o desenvolvimento de pesquisas na área de *chatterbots*. Um dos projetos mais recentes desse centro é a inclusão de *chatterbots* em ambientes de Realidade Virtual para ensino e entretenimento.

## 2.3 A Linguagem AIML

As linguagens de marcação têm sido amplamente usadas na Internet para os mais diversos objetivos. A mais usada e conhecida delas é o HTML (*HyperText Markup Language*), cuja função principal é a formatação de textos lidos por *browsers* na Web.

Essas linguagens de marcação, como já foi dito, são baseadas em uma meta-linguagem de marcação genérica chamada de SGML (*Standard Generalized Markup Language*)<sup>9</sup>, a partir da qual é possível se instanciar linguagens de marcação para diferentes aplicações. Em 1994 foi definido um novo subconjunto de SGML, uma outra meta-linguagem voltada para especificação e marcação de dados que ficou conhecida como XML (*eXtended Markup Language*)<sup>10</sup>. O XML possui cerca de 50% das especificações do SGML e atende a 90% das linguagens que a usam.

---

<sup>8</sup> <http://www.crc.ca/>

<sup>9</sup> <http://www.w3.org/MarkUp/SGML/>

<sup>10</sup> <http://www.xml.com/>

Veremos nesta seção o AIML (*Artificial Intelligence Markup Language*), uma linguagem de marcação para *chatterbots* semelhante ao HTML e baseada em tags. A linguagem AIML serviu de base ao nosso trabalho, tendo sido utilizada para formatação da base de falas que será utilizada pelo nosso *chatterbot*.

### 2.3.1 Princípios do AIML

A linguagem AIML é baseada em módulos que são chamados de categorias. Cada categoria possui um padrão de entrada, que será comparado à sentença digitada pelo usuário pelo interpretador AIML, e um padrão de resposta, que será usado para montar uma sentença a ser retornada ao usuário. Esse padrão de resposta utiliza tags de marcação para a sua construção.

Na figura 2.5, temos uma categoria básica escrita em AIML, mostrando a sua estrutura fundamental.

```
<category>  
  <pattern> padrão de entrada </pattern>  
  <template> modelo(s) para construção de respostas  
  </template>  
</category>
```

Figura 2.5: Base de uma categoria em AIML

Os padrões de entrada (contidos entre as tags `<pattern>` e `</pattern>`) também podem conter “caracteres especiais” como o “\*” e o “\_” para que o casamento de padrões ocorra com sentenças escritas pelos usuários de diferentes formas. Na figura 2.6 temos um exemplo de uma categoria que utiliza esses caracteres especiais.

```

<category>
  <pattern>KNOCK KNOCK</pattern>
  <template>WHO'S THERE ?</template>
</category>

<category>
  <pattern>*</pattern>
  <that>WHO IS THERE</that>
  <template><star/> Who ?</template>
</category>

<category>
  <pattern>*</pattern>
  <that>WHO </that>
  <template>HA HA VERY FUNNY, <get name="name"/></template>
</category>

```

**Exemplo:**

**Usuário:** KNOCK KNOCK

**Robô:** Who's there?

**Usuário:** Banana

**Robô:** banana Who?

**Usuário:** KNOCK KNOCK

**Robô:** Who's there?

**Figura 2.6:** Exemplo do uso do símbolo “\*”

Nos padrões de resposta em AIML ficam as tags que auxiliam o interpretador AIML na construção de respostas devolvidas ao usuário. Estas tags buscam melhorar o nível de coerência e naturalidade das respostas, tornando o diálogo entre o robô e o usuário cada vez mais real. Algumas tags AIML serão mostradas a seguir.

**Tag *srai***

Uma tag muito importante do AIML é a `<srai></srai>`. Ela pode ser considerada a maior responsável pelo bom desempenho dos *chatterbots* que possuem sua base de diálogos escrita em AIML. Quando uma tag `<srai></srai>` é encontrada, o

interpretador AIML não retorna o template de saída da categoria selecionada diretamente ao usuário. Neste caso, ele trata esse template como se fosse uma nova entrada do usuário, criando, assim, uma espécie de padrão recursivo, diferente da recursividade clássica das linguagens de programação, que permite que vários tipos de entrada sejam direcionados a uma única saída. O processo continua até que seja selecionada uma categoria que não possui a tag `<srai>`.

Na figura 2.7 temos um exemplo da tag `<srai></srai>`.

```
<category>
  <pattern>olá</pattern>
  <template><srai>oi</srai></template>
</category>

<category>
  <pattern>oi, tudo bom?</pattern>
  <template><srai>oi</srai></template>
</category>

<category>
  <pattern>oi</pattern>
  <template>oi, como você está?</template>
</category>

Exemplo de diálogo:

Usuário: olá
Robô: oi, como você está?

Usuário: oi, tudo bom?
Robô: oi, como você está?

Usuário: oi
Robô: oi, como você está?
```

Figura 2.7: Exemplo de uma categoria escrita com a tag `<srai></srai>`

### Tags *get name* e *set name*

Um outro exemplo de tags de marcação muito utilizadas na base de categorias AIML são as tags `<get name="xxx"/>` e `<set name="xxx"></set>`.

O interpretador AIML mantém um arquivo de propriedades para cada usuário do sistema, onde são armazenados dados como seu nome, e os diálogos entre chatterbot e esse usuário. A tag `<set name="xxx"></set>` é responsável por gravar o nome do usuário nesse arquivo, assim como algum outro dado fornecido durante o diálogo. Essa tag grava no arquivo de propriedades o nome da variável que se encontra no campo “xxx” junto com o valor correspondente, indicado na sentença de entrada do usuário (fig. 2.8).

A tag `<get name="xxx"/>` é utilizada para se obter um valor gravado anteriormente pela tag `set name`. Na tag `<get name="xxx"/>` é declarado o nome da variável da qual se quer obter o valor (campo “xxx”).

Na figura 2.8, temos um exemplo do uso das tags `get name` e `set name`.

```

<category>
<pattern>meu nome é *</pattern>
<template><set name="nome"*</set></template >
<template><srai>eu sou <star/></srai><template>
</category>

<category>
<pattern>me chamam de *</pattern>
<template><set name="nome"*</set></template >
<template><srai>eu sou <star/></srai><template>
</category>

<category>
<pattern>eu sou *</pattern>
<template>que nome bonito, <get name = "nome"/>!</template >
</category>

Exemplo de diálogo:

Usuário: meu nome é Thiago
Robô: que nome bonito, Thiago

Usuário: me chamam de Thiago
Robô: que nome bonito, Thiago

Usuário: eu sou Thiago
Robô: que nome bonito, Thiago

```

Figura 2.8: Exemplo do uso de tags `<get name="xxx"/>` e `<set name="xxx"></set>`

## Tag *that*

A tag `<that></that>` indica ao *chatterbot* qual foi sua última sentença antes da intervenção atual do usuário, auxiliando na manutenção de um discurso coerente. É uma tag bastante simples, mas resolve muitos problemas de diálogos e permite que o *chatterbot* mantenha uma maior precisão em suas respostas. Na figura 2.9 temos um exemplo do uso da tag `<that></that>`.

```
<category>
  <pattern>SIM</pattern>
  <that>voce gosta de filmes </that>
  <template>Qual o seu filme predileto?</template>
</category>
```

**Exemplo de diálogo:**

**Robô:** você gosta de filmes?  
**Usuário:** SIM  
**Robô:** Qual o seu filme predileto?

**Figura 2.9:** Exemplo do uso da tag `<that></that>`

A especificação do AIML contendo todas as tags utilizadas no A.L.I.C.E. pode ser encontrada no site da organização ALICEBOT<sup>11</sup>. Nessa especificação podem ser encontradas todas as tags AIML com suas descrições detalhadas e exemplos de uso.

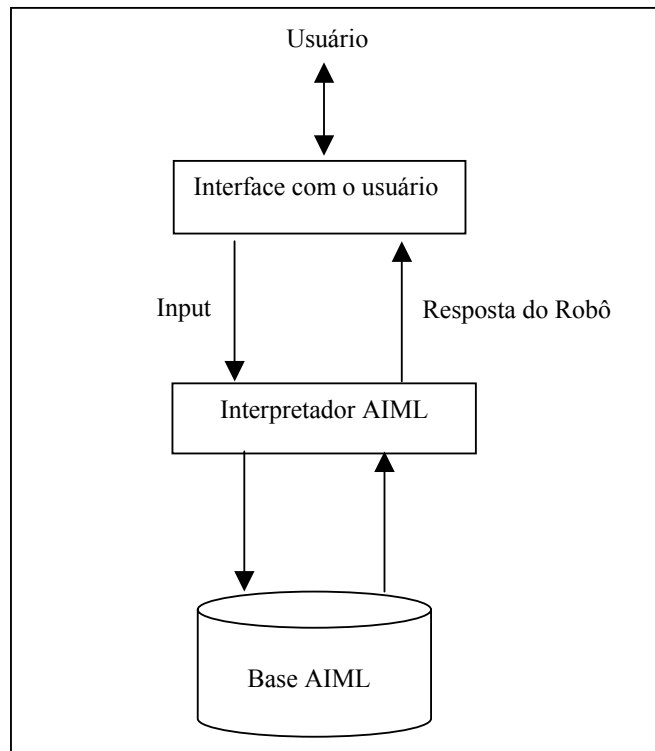
### 2.3.2 Arquitetura de *chatterbots* AIML

A arquitetura genérica de *chatterbots* AIML é composta por três módulos básicos: uma interface com o usuário; o interpretador, que seleciona categorias para os padrões de entrada e retorna respostas encontradas nessa base de categorias (podendo ser desenvolvido em diversas linguagens, como Java, PHP, C++, Perl); e uma base de categorias AIML.

Na figura 2.10, temos a arquitetura de *chatterbots* em AIML.

---

<sup>11</sup> <http://www.alicebot.org>



**Figura 2.10:** Arquitetura de um *chatterbot* AIML

## 2.4 Conclusões

Este capítulo apresentou um breve histórico sobre *chatterbots*, identificando gerações de acordo com as técnicas de implementação utilizadas em cada período. Vimos também os tipos de *chatterbots* encontrados na Web, e a linguagem AIML, utilizada como base para a criação da base de diálogos do nosso sistema.

Os *chatterbots* são uma ferramenta muito útil, e atualmente o seu uso está bastante crescente em muitas empresas virtuais que querem ter algo a mais em seus Websites. Muito sobre este tema pode ser visto na Web em sites como: *Loebner Prize*<sup>12</sup> e *BotSpot*<sup>13</sup>. Sem dúvida, hoje em dia a Internet é o maior ponto de referência e de estudo dos *chatterbots*. A maioria desses sistemas tem sido desenvolvida para funcionar na Web tendo diferentes finalidades, como foi visto. Nesses sites, encontram-se muitos tipos de robôs e sistema desenvolvidos ao longo dos anos.

<sup>12</sup> <http://www.loebner.net/Prizef/loebner-prize.html>

<sup>13</sup> <http://www.botspot.com>

Os estudos e pesquisas em cima de tecnologias de *chatterbots* não param, e muitos novos robôs são criados a cada ano. É certo que atualmente temos o uso de linguagens de marcação como a tecnologia mais viável para uso em *chatterbots*, mas isso não diz nada sobre o que pode vir a ser utilizado daqui a alguns anos.

### **O Futuro dos *Chatterbots***

Se ou não robôs poderão substituir serviços na vida de humanos ou através da Web, isso nós veremos melhor no futuro. Alguns debates sobre ajuda automática nunca terminam. No caso de robôs, eles são somente tão bons quanto a base de conhecimento que seus programadores criaram para eles. Mas nós podemos sonhar com uma Web habitada por agentes de software onde nós não teremos mais que verificar FAQs e/ou procurar por alguma informação novamente, onde nós iremos entrar com informações e conversar com um secretário que sabe nossos hábitos comuns, um velho amigo que irá nos ajudar através de algum processo de compra ou pesquisa na Web.

Em um futuro próximo, todas as interações com nosso PC podem ser através de agentes virtuais que são programados que podem aprender sobre nosso comportamento. Robert Pantano, CFO (Chief Financial Officer) do produtor de robôs, *Artificial Life* em Boston [PANTANO 2000], prevê uma época em que cada pessoa terá um agente assistente pessoal que trabalha em conjunto com outros agentes ao redor da Web, como em shoppings de robôs e busca de robôs, sem a intervenção humana: “Pessoas terão um agente distinto representando suas preferências. Eles irão atuar em seu nome, comunicando-se com outros agentes na Web”.



## 3 Aquisição de Conhecimento

---

Aquisição de Conhecimento pode ser vista como o processo de extração, estruturação e organização do conhecimento a partir de uma ou mais fontes de dados [Turban 1992]. Esse processo tem sido identificado por muitos pesquisadores e desenvolvedores como um gargalo que dificulta o desenvolvimento de sistemas especialistas e outros sistemas de Inteligência Artificial.

Neste capítulo será mostrado o processo de engenharia do conhecimento, e os métodos para aquisição de conhecimento.

### 3.1 Engenharia do conhecimento

Engenharia do conhecimento (EC) é um termo usado para descrever o processo global de desenvolvimento de um sistema baseado em conhecimento. Alguns autores o definem como o método pelo qual um especialista interage com um engenheiro de conhecimento para a construção de um sistema inteligente. O objetivo do processo de Engenharia do Conhecimento é capturar e incorporar o conhecimento fundamental de um especialista no domínio que está sendo tratado, bem como seus prognósticos e mecanismos de controle [TURBAN 1992].

A engenharia do conhecimento pode ser analisada por dois aspectos: o curto e o largo. De acordo com o aspecto “curto”, engenharia do conhecimento contribui com aquisição do conhecimento, representação, validação, inferência, explanação e manutenção de sistemas de Inteligência Artificial (IA) baseados em conhecimento. De acordo com o aspecto “largo”, o termo descreve o processo inteiro de desenvolvimento e manutenção de sistemas de IA baseados em conhecimento [TURBAN 1992].

A EC envolve a cooperação de especialistas humanos no domínio de aplicação para o qual o sistema está sendo construído, que trabalham com o engenheiro de conhecimento para codificar e explicitar as regras ou outros procedimentos que um especialista humano usa para resolver problemas reais. Às vezes o especialista usa regras aplicadas quase que de forma subconsciente [TURBAN 1992]. A engenharia do

conhecimento busca construir sistemas que tenham natureza modular, sendo possível estender e alterar as bases de conhecimento sem afetar outros módulos.

O sucesso de sistemas baseados em conhecimento depende não somente do conhecimento adquirido, mas também de como o conhecimento é representado no computador. Além disso, a máquina de inferência utilizada deve ser capaz de processar esse conhecimento.

A construção de um sistema baseado em conhecimento inclui cinco atividades [TURBAN 1992]:

1) *Aquisição do Conhecimento*: Envolve a aquisição de conhecimento de especialistas humanos, livros, documentos, sensores, ou arquivos de computadores.

2) *Representação do Conhecimento*: Essa atividade envolve preparação de um “mapa de conhecimento” adquirido, que é posteriormente codificado em uma base de conhecimento seguindo algum formalismo para representação do conhecimento.

3) *Validação do Conhecimento*: O conhecimento na base é validado e verificado e modificado se for o caso, até que sua qualidade seja aceitável.

4) *Inferência*: Essa atividade envolve a implementação de softwares capazes de fazer inferências baseadas no conhecimento, e então “aconselhar” o usuário em um assunto específico.

5) *Explicação e Justificação*: Essa atividade envolve o design e a programação de um módulo de explicação sobre o funcionamento do sistema, por exemplo, responder questões como *porque* um específico pedaço de informação é necessário ao sistema ou *como* uma certa conclusão foi derivada por um sistema baseado em conhecimento.

O processo de engenharia do conhecimento e a inter-relação entre essas atividades são mostrados na figura 3.1 [TURBAN 1992]:

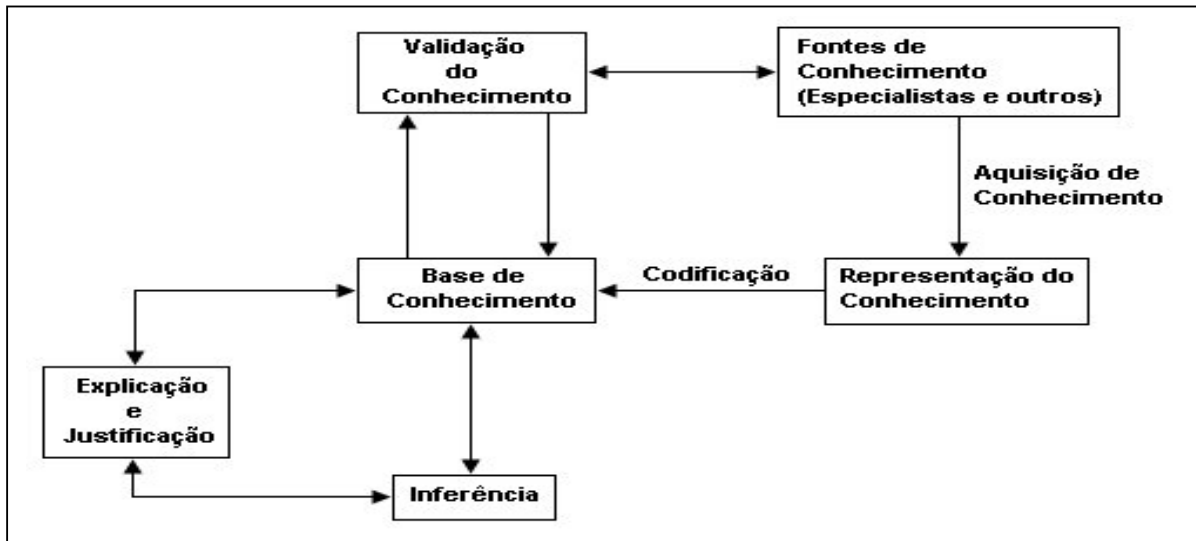


Figura 3.11: Passos da construção de um sistema baseado em conhecimento

### 3.2 Escopo do Conhecimento

Conhecimento é uma coleção de fatos especializados, procedimentos, e regras de julgamento [TURBAN 1992]. Alguns tipos de conhecimento usados na Inteligência Artificial são mostrados na figura 3.2. Esses tipos de conhecimento podem vir de uma ou mais fontes de dados.

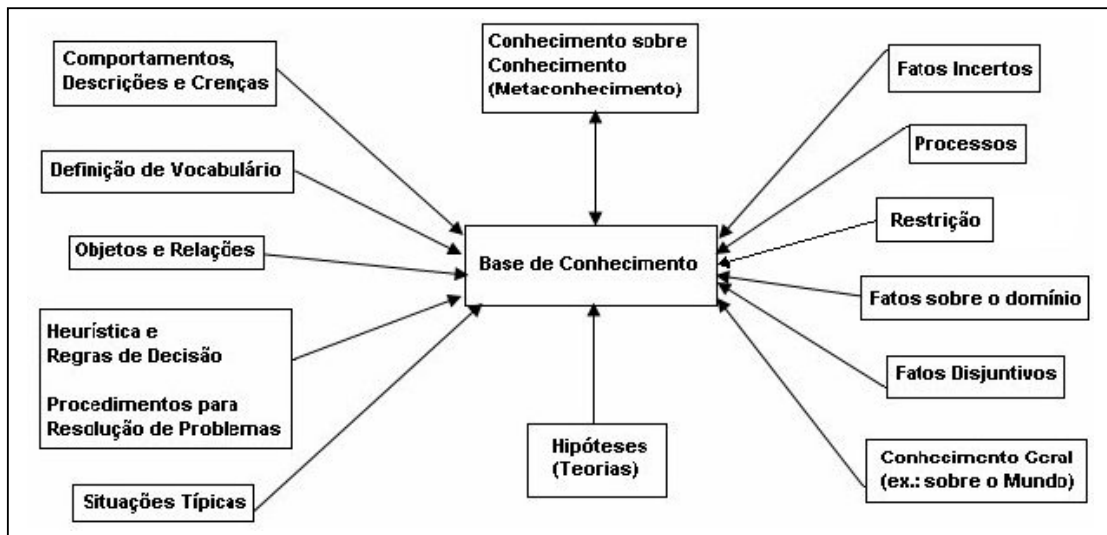


Figura 12: Tipos de Conhecimento a serem representados na base de conhecimento

## Fontes de Conhecimento

Conhecimento pode ser coletado de muitas fontes. Uma lista representativa de fontes inclui livros, filmes, bancos de dados digitais, fotos, mapas, diagramas de fluxo, relatórios, músicas e comportamento observado. Essas fontes podem ser divididas em: *documentadas e não-documentadas*.

Conhecimento pode ser identificado e coletado usando qualquer sentido humano, bem como por máquinas. A multiplicidade de fontes e tipos de conhecimento também pode contribuir para a complexidade da aquisição de conhecimento.

## Categorias de Conhecimento

O conhecimento pode ser classificado em várias categorias. Aqui temos as mais importantes delas:

*Conhecimento Declarativo*: Conhecimento Declarativo é uma representação descritiva do conhecimento. Esse tipo de conhecimento é considerado superficial, são informações que os especialistas podem verbalizar. Conhecimento Declarativo é especialmente importante no estágio inicial da aquisição do conhecimento.

*Conhecimento Procedimental*: Conhecimento Procedimental considera a maneira em que os objetos do domínio se relacionam sob diferentes circunstâncias. Por exemplo: “Computar a relação entre o preço de uma parte e os lucros por partes. Se essa relação é maior que doze, pare sua investigação. Seu investimento é muito arriscado. Se a relação é menor que doze, verifique o balanço geral”. Assim, o conhecimento procedimental inclui seqüências passo a passo e tipos de instruções, também podendo incluir explicações. Ele também pode nos dizer como usar conhecimento declarativo e como fazer inferências.

*Conhecimento Semântico*: Reflete a estrutura cognitiva que envolve o uso de uma memória de longo prazo. De acordo com [TUTHILL 1990] isso é sobre:

- palavras e outros símbolos,
- palavra/símbolo sentidos e regras usadas,
- palavra/símbolo referentes e inter-relações, e,

- algoritmos para manipulação de símbolos, conceitos e relações.

*Conhecimento Episódico*: Trata-se de informação em extensão, organizada como um caso ou episódio.

*Meta-conhecimento*: No nosso caso, meta-conhecimento refere-se a conhecimento sobre a operação de sistemas baseados em conhecimento, isto é, sobre sua capacidade de raciocínio.

*Conhecimento em Intenção*: É explicitado através de regras que definem um dado domínio.

*Conhecimento em Extensão*: É explicitado através de exemplos de um dado domínio.

### **3.3 Aquisição de Conhecimento**

O processo de aquisição de conhecimento tem sido visto como um processo de extração de conhecimento de um especialista humano e a transferência do conhecimento extraído para um sistema baseado em conhecimento. Na prática, isso geralmente significa que o especialista é questionado sobre quais regras são aplicadas em uma certa situação problemática, e o engenheiro de conhecimento traduz essas regras, em linguagem natural, para um formato computacional apropriado [BREUKER 1991].

Veremos a seguir mais detalhes sobre o processo de aquisição.

#### **Dificuldades na Aquisição de Conhecimento**

Transferir informação entre duas pessoas é sempre difícil. Máquinas muitas vezes são usadas para conduzir tais informações. Como na transferência de informações, problemas também existem na transferência de qualquer conhecimento, até em simples mensagens. Veremos a seguir as dificuldades na transferência de conhecimento.

*Expressando Conhecimento* – Pode ser muito difícil para o especialista expressar suas experiências sobre o processo sendo descrito, especialmente quando as experiências envolvem sensações, pensamentos, lembranças e emoções. O especialista geralmente não está consciente do processo que ele usa para chegar a uma conclusão.

Então, o especialista pode usar, para resolver problemas da vida real, regras diferentes das que ele apresenta em uma entrevista. O engenheiro de conhecimento deve pedir ao especialista para observar aspectos “escondidos” de tomada de decisões e suas experiências íntimas envolvidas nesse processo.

*Transferindo conhecimento para o computador* – O conhecimento é transferido para o computador, quando esse conhecimento está montado de forma organizada e passível de se transferir a um computador. O computador requer que o conhecimento seja expresso formalmente em um nível mais baixo, mais detalhado, do que geralmente os humanos usam. O conhecimento humano existe em um formato compilado. As pessoas simplesmente não se lembram de todos os passos intermediários usados pelo seu cérebro na transferência ou no processamento de conhecimento.

*Número de Participantes* – Em uma transferência regular de conhecimento, existem pelo menos dois participantes, o que envia e o que recebe. Em engenharia do conhecimento, nós podemos vislumbrar quatro participantes: o especialista, o engenheiro de conhecimento, o designer do sistema e o usuário. Às vezes temos mais participantes, como programadores do sistema e outros. Cada participante no processo tem diferentes experiências e diferentes terminologias, e possui diferentes habilidades e conhecimentos.

*Estruturando o Conhecimento* – Em engenharia do conhecimento, é necessário obter não somente o conhecimento, mas também sua estrutura e formalismo. É necessário representar o conhecimento em algum tipo de estrutura e formalismo, como por exemplo, regras.

Um método para facilitar a falta de acordo da representação do conhecimento entre um especialista e um programa de computador é permitir que o especialista converse com o programa em linguagem natural.

Muitas das dificuldades podem ser diminuídas ou eliminadas com auxílio do computador pelo uso de ferramentas de aquisição de conhecimento, e com extensiva integração de esforços na aquisição [BOOSE 1989].

## **Processos de Aquisição de Conhecimento**

O processo de aquisição de conhecimento pode ser visto como sendo composto de cinco passos, descritos abaixo [HAYES-ROTH et al. 1983]:

*Identificação* – Durante esta fase, o problema e suas características principais são identificados. O problema é quebrado em subproblemas, os participantes são identificados e os recursos disponíveis para a construção do sistema, são traçados.

*Conceitualização* – O conhecimento relevante para o problema tratado pode ser muito diversificado. É necessário determinar os conceitos e os relacionamentos a serem usados.

*Formalização* – O conhecimento é adquirido para representação na base de conhecimento. A maneira como o conhecimento é organizado e representado pode determinar a metodologia de aquisição. Esse estágio é muito difícil porque ele inclui a extração de conhecimento de especialistas.

*Implementação* – Este estágio envolve a construção da base de conhecimento. Refinamentos dessa base podem ser feitos com aquisições adicionais ou correções dos erros detectados. Um protótipo de sistema especialista é desenvolvido.

*Teste* – No estágio final, o engenheiro de conhecimento testa o sistema com exemplos. A consistência da base de conhecimento é examinada. O especialista geralmente volta a ser consultado nesta fase.

Cada estágio envolve procedimentos circulares e repetições. O engenheiro de conhecimento reformula, redesenha e refina o sistema constantemente.

A figura 3.3 representa os estágios da aquisição do conhecimento [Turban 1992].

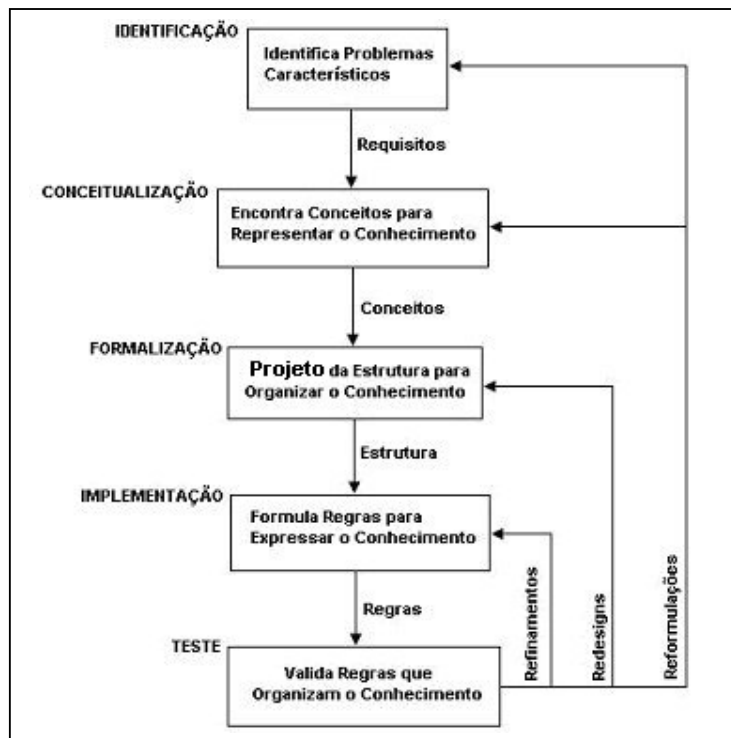


Figura 13: Estágios da Aquisição do Conhecimento

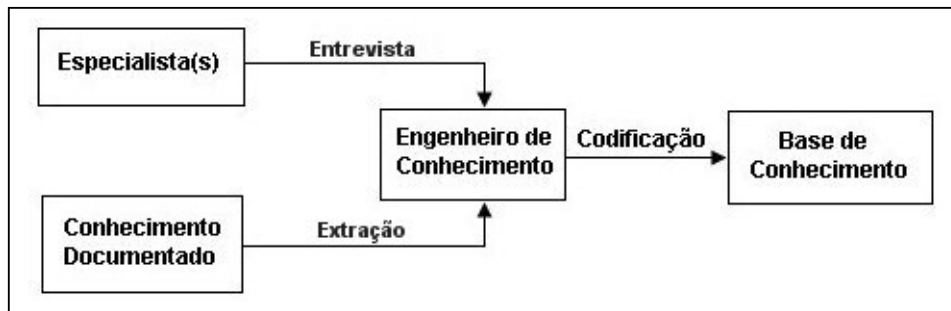
## Métodos de Aquisição de Conhecimento

O engenheiro de conhecimento extrai conhecimento do especialista, refina esse conhecimento com a ajuda do especialista, e o representa em uma base de conhecimento. A extração do conhecimento do especialista pode ser feita manualmente ou com a ajuda de sistemas computacionais. Muitas das técnicas manuais de extração têm sido inspiradas na psicologia e em outros sistemas de análise. Esses métodos de extração são classificados de diferentes formas e aparecem sobre diferentes nomes.

Os métodos mostrados aqui serão classificados em três categorias: manual, semi-automática e automática.

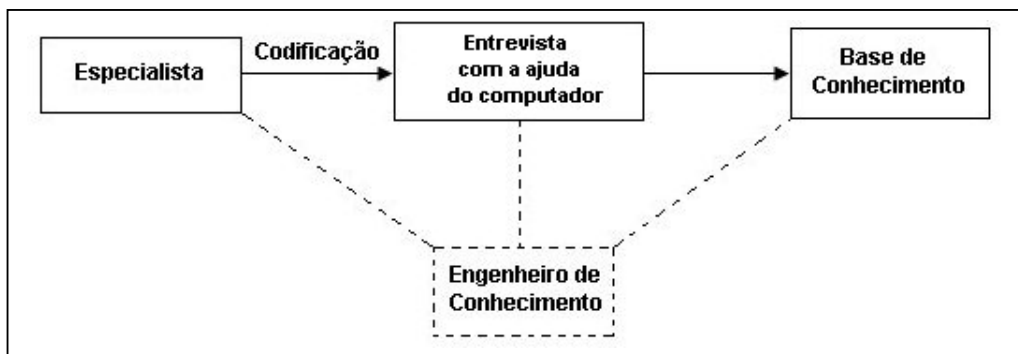
1. Métodos Manuais – Baseiam-se em alguns tipos de entrevistas (estruturadas e não-estruturadas). O engenheiro de conhecimento extrai conhecimento do especialista e/ou de outras fontes (conhecimento documentado) e codifica-o na base de conhecimento. O processo é mostrado na figura 3.4. Os três métodos manuais mais difundidos são entrevistas, rastreamento do processo de raciocínio e observação. Métodos manuais são lentos, caros e muitas vezes inexatos. (Ver mais detalhes na seção 3.4).





**Figura 14:** Métodos Manuais

2. Métodos Semi-Automáticos - São divididos em duas categorias: (1) Aqueles cujo objetivo é permitir que os especialistas possam construir bases de conhecimento com pouca ou nenhuma ajuda do engenheiro de conhecimento, (figura 3.5); e (2) Aqueles cujo objetivo é ajudar os engenheiros de conhecimento, permitindo que eles executem as tarefas necessárias à aquisição, de uma maneira mais efetiva e/ou eficiente.(Ver detalhes na seção 3.6).



**Figura 15:** Aquisição de Conhecimento guiada pelo especialista

3. Métodos Automáticos – Aqui, a função do especialista e/ou engenheiro de conhecimento é minimizada ou até eliminada. A aquisição de conhecimento é feita de forma automática pela aplicação de algoritmos de aprendizagem automática. Como exemplo, temos o processo indutivo, mostrado na figura 3.6. Contudo, o termo “automático” pode ser equivocado. Existe sempre um construtor humano, que irá montar o corpus de treinamento e teste do algoritmo de aprendizagem, mas existe talvez pouca ou nenhuma necessidade de um engenheiro de conhecimento ou de especialistas. (Ver detalhes na seção 3.7).

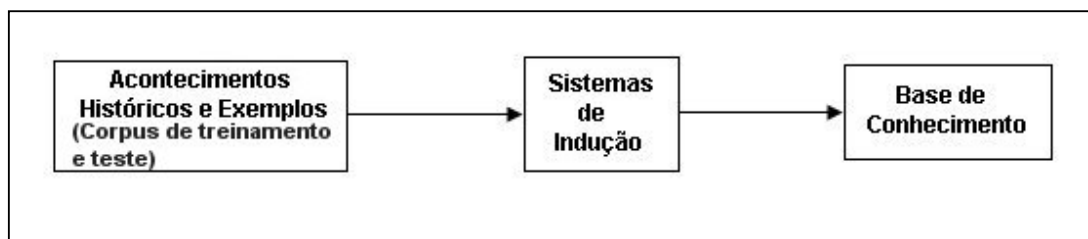


Figura 16: Indução

### 3.4 Métodos Manuais

Aqui serão apresentados alguns dos métodos manuais de aquisição de conhecimento: entrevistas, métodos de rastreamento, observações e outros métodos alternativos.

#### 3.4.1 Entrevistas

A forma mais comumente usada para aquisição de conhecimento é a análise de entrevistas pessoalmente. Essa técnica envolve um diálogo direto entre o especialista e o engenheiro de conhecimento. Informações são coletadas com a ajuda de instrumentos convencionais, como fitas, gravadores ou questionários, e essas informações são subsequentemente transcritas, analisadas e codificadas.

Na entrevista, o especialista é apresentado a um caso simulado, ou se possível a um problema real parecido com o que o sistema especialista deverá resolver. O processo de entrevista pode ser tedioso e requer que o engenheiro de conhecimento possua muita habilidade.

Existem dois tipos de entrevistas que podem ser distinguidas: não-estruturadas e estruturadas.

#### Entrevistas não-estruturadas

Muitas entrevistas de aquisição de conhecimento são conduzidas informalmente, freqüentemente como um ponto de partida. Iniciadas informalmente para poupar o tempo, ajudam a adquirir rapidamente a estrutura base do domínio.

A função do engenheiro de conhecimento é realizar perguntas espontaneamente. Entrevistas não-estruturadas são raramente simples. De fato, elas podem apresentar ao engenheiro de conhecimento alguns efeitos colaterais problemáticos.

De acordo com [McGRAW et. al. 1989], essas entrevistas raramente provêm descrições de processos cognitivos completos ou bem organizados. Primeiro, eles observaram que os domínios dos sistemas especialistas são geralmente complexos, assim o engenheiro de conhecimento e o especialista devem ativamente preparar situações adequadas para a entrevista. Segundo, especialistas do domínio geralmente enfrentam dificuldades para expressar alguns elementos importantes de seu conhecimento. Terceiro, especialistas no domínio podem representar a ausência de estrutura, requerendo pouca preparação em partes anteriores a entrevista. Quarto, aquisição de dados em uma entrevista não-estruturada muitas vezes são irrelevantes, existindo vários níveis de complexidade, e essas entrevistas são difíceis para o engenheiro de conhecimento revisar, interpretar e integrar.

Um quinto problema citado por [McGRAW et. al. 1989] é a preocupação com o treinamento, porque pela ausência de treinamento e experiência, poucos engenheiros de conhecimento podem conduzir uma eficiente entrevista não-estruturada. Finalmente, e mais importante, é que situações não-estruturadas geralmente não facilitam a aquisição de informações específicas em relação ao especialista.

### **Entrevistas Estruturadas**

A entrevista estruturada é um processo sistemático orientado ao objetivo de aquisição de conhecimento. Força uma comunicação organizada entre o engenheiro de conhecimento e o especialista. A estrutura reduz problemas de interpretação inerentes a entrevistas não-estruturadas, e permite que o engenheiro de conhecimento previna distorções causadas pela subjetividade do especialista do domínio.

Por causa uma natureza específica para cada entrevista, é difícil prover um bom guia para o completo processo da entrevista. Entretanto, comunicação pessoal e habilidades analíticas são importantes. Existem, entretanto, muitos guias, listas de checagem e instrumentos para este fim.

Em resumo, entrevistas são técnicas importantes para a aquisição de conhecimento, mas elas devem ser planejadas cuidadosamente, e os resultados devem ser submetidos à verificação completa e a métodos de validação.

### **3.4.2 Métodos de Rastreamento**

Este método refere-se a um conjunto de técnicas que tentam rastrear o processo de raciocínio de um especialista. O engenheiro de conhecimento pode usar o processo de rastreamento para encontrar como e quais informações serão inicialmente usadas para a aquisição do conhecimento. Métodos de rastreamento podem ser formais ou informais. O mais comum método formal é o Protocolo de Análises [TURBAN 1992].

Esse método, que reúne um conjunto de técnicas conhecidas como protocolos verbais de análises, possibilita que o engenheiro de conhecimento adquira conhecimento detalhado do especialista. Um protocolo é um registro ou documentação do processo das informações passo a passo do especialista e o comportamento do ato de tomar decisões. Nesse método, que é similar à entrevista estruturada, porém mais formal e sistemático, o especialista é solicitado a executar uma tarefa real e verbalizar seu processo de raciocínio.

Em contraste com métodos de entrevistas interativas, um protocolo de análise envolve principalmente uma comunicação unilateral. O engenheiro de conhecimento prepara o cenário e planeja o processo. Durante a sessão, o especialista fala ao engenheiro de conhecimento interativamente, apresentando dados para resolver o problema. O engenheiro de conhecimento escuta e registra o processo. Mais tarde, ele deve ser capaz de analisar, interpretar e estruturar o protocolo dentro da representação do conhecimento escolhida, que será submetida a uma revisão do especialista.

### **3.4.3 Observações e Outros Métodos Manuais**

#### **Observações**

Observações podem ser vistas como um caso especial de protocolos, e são de dois tipos: motorizada e visual. No primeiro tipo, o comportamento físico (e.g., o andar, a

chegada e a conversa) do especialista é documentado. No segundo tipo, um registro é feito sobre o que o especialista observou, para que locais ele olhou durante a entrevista, etc.

Observações são usadas primeiramente como um caminho de suporte verbal de protocolos. Elas são geralmente dispendiosas e consomem muito tempo.

### **Outros Métodos Manuais**

Muitos outros métodos manuais podem ser usados para extrair conhecimento de especialistas. Aqui, mostramos uma lista com os métodos mais representativos [TURBAN 1992].

*Análise de Casos:* Os especialistas são questionados sobre como eles cuidaram de casos específicos no passado. Geralmente esse método envolve análise de documentação. Juntamente com os especialistas, outras pessoas (e.g., administradores e usuários) podem ser questionadas.

*Análise de Acontecimentos Críticos:* Nessa abordagem, somente casos selecionados são investigados, geralmente aqueles que são importantes, difíceis ou de um interesse especial, escolhidos pelo Engenheiro de Conhecimento. Especialistas e outras pessoas podem ser questionadas.

*Comentários:* Nesse método, o engenheiro de conhecimento pede ao especialista para que ele faça um rápido comentário sobre o que ele está fazendo. Esse método pode ser auxiliado por gravações de especialistas em ação, ou o engenheiro pode pedir a um observador para fazer os comentários.

*Gráficos Conceituais:* Diagramas e outros métodos gráficos podem servir de instrumento no auxílio a outros métodos de aquisição.

*Brainstorming:* Esse método pode ser usado para solicitar a opinião de múltiplos especialistas, com o objetivo de gerar idéias para a tomada de decisão.

*Prototipagem:* Trabalhar com um protótipo de um sistema é uma abordagem potente para induzir especialistas a contribuírem com a aquisição do conhecimento. Especialistas gostam de criticar sistemas. Mudanças podem ser feitas instantaneamente.

*Escalonamento Multidimensional:* A complexa técnica de escalonamento multidimensional identifica várias dimensões do conhecimento e então estabelece o conhecimento na forma de uma matriz de distância. Usando uma regressão adequada de mínimo-quadrado, várias dimensões são analisadas, interpretadas e integradas.

*Clustering Hierárquico de Johnson:* Esse é outro método de escalonamento, mas ele é muito simples de implementar e, portanto, é mais usado que os citados acima. Ele combina elementos de conhecimento agrupados em *clusters* (dois elementos por vez).

### **3.5 Métodos Guiados por Especialistas**

Nos métodos apresentados anteriormente, o engenheiro de conhecimento era o técnico responsável por representar o conhecimento adquirido, cujas habilidades determinavam a qualidade da base de conhecimento. Entretanto, esses profissionais, tipicamente carecem de conhecimento sobre domínios específicos, seus serviços são caros e eles podem ter problemas de comunicação com os especialistas. Como resultado, aquisição de conhecimento pode ser um processo lento, com muitas interações (para verificação e com o objetivo de aprender). O processo também não é confiável porque os especialistas podem encontrar dificuldades para contribuir com seus conhecimentos em processos via engenheiro de conhecimento.

Talvez especialistas possam ser seus próprios engenheiros de conhecimento, codificando suas especialidades em sistemas computacionais. Assim, com processos de aquisição guiadas por especialistas, podem-se resolver algumas das dificuldades descritas acima, o que poderá resultar em menos “ruído” inicialmente introduzido na base de conhecimento. A função de engenheiros de conhecimento será reduzida e os gastos no processo de aquisição serão drasticamente diminuídos.

De acordo com [KIM e COURTNEY 1988], as seguintes suposições são necessárias para implementar a base de conhecimento com sucesso:

- O especialista pode identificar as variáveis e os relacionamentos entre essas variáveis;
- O especialista pode aprender a usar a interface;

- O especialista pode usar técnicas de ajuda com o computador para executar o processo quando necessário;
- O especialista pode estruturar um modelo refinado usando uma estrutura aproximada do domínio;
- A inevitável perda de transparência na codificação do conhecimento é aceitável se o especialista puder assegurar o desempenho do modelo da base de conhecimento.

Duas abordagens de sistemas guiados por especialistas são avaliadas a seguir: Manual e Com o auxílio do Computador (semi-automático).

### **3.5.1 Método Manual: Relatos Próprios dos Especialistas**

Em alguns casos, é possível extrair conhecimento de especialistas manualmente usando um questionário bem montado ou um relatório organizado. Além dos questionários, especialistas podem ser questionados a documentar suas atividades, preparar uma palestra introdutória de curta duração, ou produzir relatórios sobre suas atividades de solução de problemas.

Relatórios de especialistas e questionários apresentam alguns problemas, de acordo com [WOLFGRAM 1987].

1. Eles essencialmente requerem que os especialistas atuem como engenheiros de conhecimento, sem treinamento adequado.
2. Os relatórios tendem a ter alto grau de influência do especialista, eles tipicamente refletem a opinião do especialista com respeito a como a tarefa “pode ser feita”, ao invés de “como ela é realmente feita”.
3. Especialistas freqüentemente descrevem idéias novas e não testadas, e estratégias que eles admiram, mas que ainda não incluem em seus processos internos de decisão. Assim, existe uma mistura de experiência passada, comportamento atual e comportamento ideal futuro.

4. Relatórios de especialistas necessitam de muito tempo e esforço, e os especialistas perdem o interesse rapidamente. A qualidade da informação se degrada rapidamente com o desenvolver do relatório.
5. Especialistas precisam ter prática no mapeamento de fluxogramas e em outras técnicas de documentação de processos.
6. Especialistas podem se esquecer de especificar certas partes do conhecimento (o que pode resultar em ambigüidade).
7. Especialistas geralmente tendem a serem vagos sobre a natureza de associações entre eventos (o que pode resultar em alguma influência indeterminada sobre essas associações).

Considerando essas advertências sob certas condições, como a inacessibilidade de um especialista para com um engenheiro de conhecimento, relatórios de especialistas e seus próprios questionários podem prover um útil conhecimento preliminar para a aquisição de conhecimento.

Algumas das limitações da abordagem manual podem ser removidas se um computador for usado para suportar os processos (ver próxima seção).

### **3.5.2 Aproximações com auxílio do computador**

O propósito do suporte computadorizado para o especialista é reduzir ou eliminar problemas potenciais discutidos acima, especialmente aqueles com influências indeterminadas e ambigüidades [KIM e COURTNEY 1988].

Uma ferramenta eficaz de aquisição de conhecimento precisa ser capaz de adicionar conhecimento incrementalmente na base de conhecimento e refinar ou ajustar corretamente o conhecimento existente.

Técnicas visuais de modelagem são geralmente usadas para construir o modelo inicial do domínio. O objetivo da modelagem visual é dar ao usuário a possibilidade de visualizar os problemas do mundo real e manipular elementos através do uso de gráficos. Muitas ferramentas podem ser usadas pelos especialistas [BOOSE e GAINES 1990].



De interesse especial são aqueles métodos que baseados na análise de repertório de grade, que serão vistos na próxima seção.

### **3.6 Métodos Semi-Automáticos**

Aqui mostraremos alguns dos métodos de aquisição de conhecimento semi-automáticos.

#### **3.6.1 Análise de Repertório de Grade (Repertory Grid Analysis)**

Segundo este método, a experiência é geralmente baseada na percepção, visão e intuição. Entretanto, muitos especialistas têm dificuldades de expressar sua linha de raciocínio. A experiência também pode ser confundida entre fatos e fatores que atualmente influenciam a tomada de decisão. Para superar essas e outras limitações da aquisição de conhecimento baseada na percepção do modelo mental do especialista, técnicas de descobertas têm sido desenvolvidas. Essas técnicas, derivadas da psicologia, usam uma abordagem chamada de *entrevista de classificação*.

Desde que elas sejam estruturadas de modo correto, quando aplicadas à construção de sistemas baseados em conhecimento, essas técnicas são geralmente auxiliadas por um computador. O método básico é Análise de Repertório de Grade (RGA).

RGA é baseado no modelo de Kelly do pensamento humano, chamado Teoria da Construção Pessoal. De acordo com essa teoria, cada pessoa é vista como um “cientista pessoal” que procura adivinhar e controlar eventos para criar teorias, testando hipóteses e analisando resultados de experimentos. Conhecimento e percepções sobre o mundo (ou sobre um domínio ou problema) são classificados e categorizados por cada indivíduo como um modelo de percepção pessoal. Baseado no modelo desenvolvido, cada indivíduo é capaz de perceber e então atuar com base nessas intuições.

Esse método pessoal nos mostra como um especialista trabalha. Ele é uma descrição do desenvolvimento e uso do conhecimento do especialista, e assim sendo, ele é adequado para sistemas especialistas, como sugerido por [HART 1986].

O RGA combina vários processos. Primeiro, o especialista identifica os objetos importantes no domínio específico. Segundo, o especialista combina os atributos importantes que são considerados na tomada de decisão no domínio. Terceiro, para cada atributo, o especialista é convidado a estabelecer uma escala bipolar com características distintas (traços) e características opostos a esses traços. Quarto, o entrevistador escolhe três objetos quaisquer e pergunta: Quais atributos e traços distintos os outros dois objetos têm em relação ao primeiro? Por exemplo, se um conjunto inclui as linguagens LISP, PROLOG e COBOL o especialista pode ter um ponto de “orientação”. Então o especialista irá perceber que LISP e PROLOG são de natureza simbólica, enquanto COBOL é numérica. Esse passo questionado com perguntas sobre tais características dos dados do conjunto.

Com essas respostas, os valores são gravados e postos em uma grade, ver figura 3.7. Quando a grade está completa, o especialista pode mudar as avaliações dentro da grade. A grade pode ser usada mais tarde para fazer recomendações em situações onde a importância dos atributos é conhecida.

<b>Atributo</b>	<b>Orientação</b>	<b>Facilidade de Programar</b>	<b>Tempo de Treinamento</b>	<b>Disponibilidade</b>
<b>Característica Oposta</b>	<b>Simbólica (3)</b>	<b>Alto (3)</b>	<b>Alto (1)</b>	<b>Alto (3)</b>
	<b>Númérica (1)</b>	<b>Baixo (2)</b>	<b>Baixo (3)</b>	<b>Baixo (1)</b>
LIST	3	3	1	1
PROLOG	3	2	2	1
C	2	3	2	2
COBOL	1	2	1	3

**Figura 17:** Um exemplo de Grade

Algumas ferramentas de aquisição de conhecimento foram desenvolvidas baseadas em RGA [BOOSE 1989]. Essas ferramentas são utilizadas para ajudar a conceituar o domínio. Três ferramentas representativas são: ETS [BOOSE 1986], KRITON [DIEDERICH et. al.1987] e AQUINAS [BOOSE e BRADSHAW 1987].

*Sistema de Transferência de Especialidade (ETS)* é um programa de computador que entrevista especialistas e ajuda-os a construir sistemas especialistas. ETS entrevista especialistas para descobrir o vocabulário utilizado, características de resolução de problemas, traços estruturais, traços importantes e inconsistentes. É utilizado para

construir protótipos rapidamente, para ajudar o especialista a determinar se existe conhecimento suficiente para resolver o problema e criar bases de conhecimento para uma variedade de diferentes *shells* de sistemas especialistas com suas próprias representações internas.

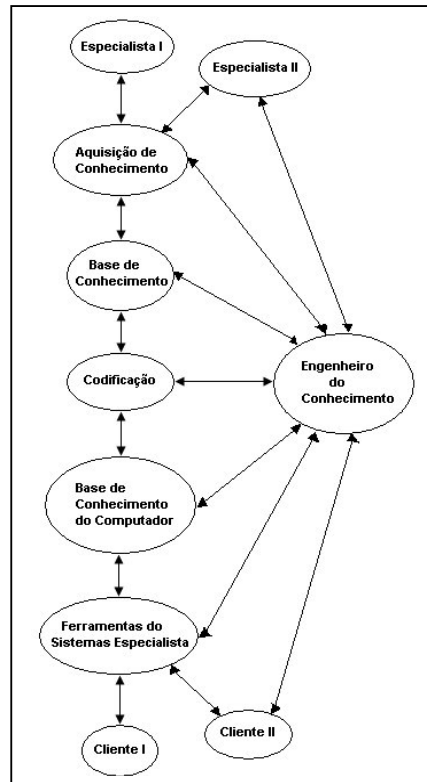
*KRITON* é um sistema que tenta automatizar o uso da abordagem da lista de grade. Primeiro ele conduz entrevistas com especialistas, então, ele analisa protocolos e documentos interagindo com os especialistas.

*AQUINAS* é uma ferramenta muito complexa, ver em [BOOSE e BRADSHAW 1987] que facilita resolver o problema e a representação do conhecimento do ETS, permitindo aos especialistas estruturarem o conhecimento em hierarquias.

### **3.6.2 Apoio ao Engenheiro de Conhecimento**

Algumas ferramentas de aquisição e codificação de conhecimento reduzem de forma considerável o tempo gasto pelo engenheiro de conhecimento. A figura 3.7 descreve as principais tarefas do engenheiro de conhecimento: [TURBAN 1992]

- Avisar ao especialista do processo interativo de descoberta do conhecimento;
- Controlar as ferramentas de interatividade de aquisição de conhecimento, instalando-as apropriadamente;
- Editar a base de conhecimento não-codificada e codificá-la em conjunto com o especialista;
- Controlar as ferramentas de codificação do conhecimento, instalando-as apropriadamente;
- Validar a aplicação da base de conhecimento em conjunto com o especialista;
- Treinar os clientes no uso efetivo da base de conhecimento, em conjunto com o especialista, para o desenvolvimento operacional e conhecimento dos procedimentos.



**Figura 18:** Função do Engenheiro de Conhecimento em uma Aquisição de Conhecimento Interativa.

Muitos tipos de ferramentas têm sido desenvolvidas para suportar aquisição de conhecimento. Alguns exemplos representativos seguem abaixo.

*Editores e Interfaces.* O uso de um editor de texto ou um editor especial de base de conhecimento pode facilitar a tarefa de entrada do conhecimento no sistema e diminuir as chances de erros.

*Facilidade de Explicação.* Os subsistemas de explicação servem não somente para o usuário, mas também para o engenheiro de conhecimento e para o especialista no refinamento da base de conhecimento, com o objetivo de corrigi-la encontrando possíveis erros.

*Revisão da Base de Conhecimento.* Mudanças na base de conhecimento podem ser feitas selecionando uma revisão apropriada de um conjunto de possibilidades. Para evitar a introdução de novos erros ou inconsistências com a base de conhecimento existente, ajudas como a verificação de consistência semântica ou testes automáticos pode ser usada.

*Aquisição de Conhecimento em Desenho (PIKA).* PIKA é um editor gráfico cuja saída é uma coleção de objetos gráficos estruturados que podem ser combinados para suportar uma interface de multinível [FREILING et al. 1985].

*Ajuda na Aquisição de Conhecimento.* Um programa clássico, chamado TEIRESIAS [HAYES-ROTH et al. 1983] foi desenvolvido para ajudar os engenheiros de conhecimento na criação (ou revisão) de regras para um sistema especialista específico enquanto trabalham com o *shell* do EMYCIN [EMYCIN 2002]. O programa usa uma interface em linguagem natural para ajudar o engenheiro de conhecimento no teste e na depuração do novo conhecimento. Ele provê uma grande capacidade de explicação.

Muitas dessas ferramentas, e muitas outras eram projetadas separadamente, assumindo-se que elas seriam usadas por um participante específico do sistema especialista, para a execução de uma tarefa específica. Na realidade, participantes podem representar múltiplas funções ou até trocar de função. Então, existe uma tendência de integrar auxílios à aquisição.

*Aquisição de Conhecimento e Documentação de Sistemas (KADS).* O propósito dessas técnicas, que foram desenvolvidas na Universidade de Amsterdã, é ajudar ao engenheiro de conhecimento na aquisição, estruturação, análise e documentação de conhecimento especializado [TURBAN 1992]. KADS é reconhecido por ser muito bem sucedido na crescente produtividade do engenheiro de conhecimento.

Em KADS, o desenvolvimento de sistemas baseados em conhecimento (SBC) é visto como uma modelagem de atividades. Um SBC não é um recipiente cheio de conhecimento extraído de um especialista, mas um modelo operacional que exhibe alguns comportamentos desejados, que podem ser observados em termos de fenômenos do mundo real [BREUKER 1991].

### **3.7 Método Automático - Indução**

Aqui mostraremos o método de aquisição de conhecimento automático baseado em indução.

Os métodos de aquisição apresentados acima requerem trabalho intensivo. Os dois maiores participantes são o engenheiro de conhecimento, que é difícil de se

conseguir e geralmente é muito bem pago, e o especialista do domínio, que quase sempre é ocupado e freqüentemente não colabora. Então, métodos manuais e até mesmo métodos de descoberta semi-automáticos são ambos lentos e caros, possuindo muitas desvantagens.

Assim, faz sentido desenvolver métodos de aquisição de conhecimento que reduzam e até mesmo eliminem a necessidade desses dois participantes. Esses métodos, que são descritos como aquisição do conhecimento com *auxílio do computador*, ou métodos automáticos de aquisição de conhecimento, variam em seus objetivos. Alguns desses objetivos são listados aqui:

- Aumentar a produtividade da engenharia do conhecimento (diminuir os custos);
- Reduzir o nível de habilidade requerido por um engenheiro de conhecimento;
- Eliminar (ou drasticamente reduzir) a necessidade de um especialista;
- Eliminar (ou drasticamente reduzir) a necessidade de um engenheiro de conhecimento;
- Aumentar a qualidade da aquisição do conhecimento.

### **3.7.1 Indução Automática de Regra**

Indução é um processo de raciocínio que parte de casos específicos para gerar uma regra geral. Na terminologia de sistemas especialistas, isso se refere ao processo pelo qual regras são geradas por um programa de computador (um algoritmo de aprendizagem automática) a partir de exemplos de casos.

Um sistema de indução de regras que utiliza aprendizagem supervisionada parte de exemplos com solução conhecida (chamado de conjunto de treinamento), e induz as regras que definem esses exemplos. Em seguida, as regras induzidas são testadas com outro conjunto de exemplos (conjunto de teste), e o desempenho do sistema é avaliado. Caso o desempenho ainda não seja satisfatório, as regras são refinadas. Uma vez terminada as etapas de treinamento e testes, as regras geradas podem ser usadas para avaliar outros casos onde o resultado não é conhecido.

Existem casos em que o domínio do conhecimento é fixo, pequeno, unido de forma imprecisa, ou modulado. Quando o domínio se torna grande e mais complexo, os especialistas se tornam incapazes de explicar como ele funciona. Eles podem, de qualquer forma, ainda fornecer ao engenheiro de conhecimento exemplos adequados de problemas e soluções. A indução de regras permite que sistemas especialistas sejam usados em domínios mais complicados e comercialmente mais compensadores.

Outra vantagem é que o construtor não tem que ser o engenheiro de conhecimento. Ele pode ser o especialista ou analista do sistema. Isso não somente economiza tempo e dinheiro, mas também resolve as dificuldades de lidar com os negócios, já que o engenheiro de conhecimento não está familiarizado com os “negócios”.

Uma vez que as regras tenham sido geradas, elas são examinadas pelo especialista e modificadas, se necessário. Uma grande vantagem de indução de regras é que ela auxilia o processo de raciocínio de um especialista.

Apesar das vantagens, muitas dificuldades existem na implementação de sistemas para indução de regras. A maior dificuldade com o método, de acordo com [HART 1986], é que o construtor não sabe se o número de exemplos será suficiente e se o algoritmo é bom o suficiente. Para ser claro, é pressuposto que o construtor tenha alguma idéia da “solução”. A razão para uso da indução na primeira instância é que o construtor não sabe a solução (tem apenas uma idéia), mas quer descobri-la usando as regras.

Por causa das limitações, o método de indução é usado para prover um primeiro protótipo, então, ele é traduzido em algo mais robusto, e manuseado dentro de um sistema apropriado.

### **3.7.2 Indução Interativa**

A combinação de um especialista apoiado por um computador é chamada de *indução interativa*. Uma tentativa interessante que combina indução e aquisição interativa pode ser encontrada em uma ferramenta chamada Auto-Inteligência

A Auto-Inteligência captura o conhecimento de um especialista através de entrevistas interativas, depura (refina) o conhecimento, e então automaticamente gera

uma base de conhecimento baseada nas regras. Uma parte importante da interação com a Auto-Inteligência é a descoberta de uma estrutura do domínio.

A estrutura é descoberta pela utilização de técnicas de dedução especiais de geração de questões no relatório de grade, escalonamento multidimensional e classificação de dados. Essas técnicas ajudam os especialistas a pensarem sobre problemas locais, peça por peça, assim gradualmente revelando suas especialidades.

A Auto-Inteligência constrói sistemas especialistas para seleção estruturada ou tarefas de classificação estruturada em que um especialista toma decisões e as seleciona, dentre um número de possibilidades, baseado em um critério de avaliação.

### **3.8 Adquirindo conhecimento**

Esta seção discute como selecionar um método automático apropriado para aquisição de conhecimento, como validar e verificar a base de conhecimento, e como analisar, codificar, documentar e fazer a diagramação do conhecimento.

#### **Selecionando um Método Apropriado**

Métodos automáticos de aquisição de conhecimento, conhecidos também como aprendizado de máquina, são limitados em suas capacidades. Apesar disso, esforços por parte dos pesquisadores, vendedores, e construtores de sistemas especialistas estão ajudando os sistemas à lentamente se aproximarem dos objetivos propostos para um sistema ideal de aquisição de conhecimento. Com isso, melhores resultados podem ser alcançados na aquisição de conhecimento, que é difícil de ser realizada manualmente.

Desde que o conhecimento seja coletado para o propósito de ser representado, a representação pode ter um maior impacto nos métodos selecionados. Idealmente, só deveria ser iniciada a seleção do método com uma aquisição de conhecimento preliminar, e assim então decidir qual seria a técnica de representação baseada no tipo de conhecimento coletado, e somente então determinar o método de aquisição. Às vezes o formalismo de representação pode ser mudado, uma vez que o protótipo está em



desenvolvimento. Entretanto, isso também é verdade para o método de aquisição de conhecimento.

A experiência e o julgamento do engenheiro de conhecimento vão determinar qual o melhor método para cada situação específica.

### **Validação e Verificação da Base de Conhecimento**

Aquisição de conhecimento envolve aspectos de controle de qualidade que aparecem sob os termos avaliação, validação e verificação.

*Avaliação* é um conceito amplo. O objetivo desta etapa é avaliar sistemas de Inteligência Artificial completos. Em acréscimo, para avaliar o nível aceitável de desempenho, ele analisa se o sistema será útil, eficiente e qual o seu custo efetivo.

*Validação* é a parte da avaliação que lida com o desempenho do sistema. Simplesmente, validação refere-se a construir o sistema “correto”, verificando, se seu desempenho apresenta um nível aceitável de precisão.

*Verificação* refere-se a construir o sistema “correto”, verificando se o sistema corretamente implementa o que foi especificado.

Na área de sistemas especialistas, essas atividades são dinâmicas, uma vez que elas podem ser repetidas a cada vez que o protótipo é mudado.

A validação e verificação da base de conhecimento devem ser conduzidas várias vezes durante o ciclo de vida do sistema.

### **Analisando, Codificando, Documentando e fazendo a Diagramação**

O conhecimento coletado deve ser analisado, codificado e documentado. A maneira como essas atividades são realizadas depende dos métodos de aquisição e representação. Abaixo temos alguns passos para a realização desse processo [WOLFGRAM 1987]:

*Transcrição.* Primeiro, uma completa transcrição do relato verbal é feita, incluindo não somente expressões dos especialistas, mas também aquelas do engenheiro de conhecimento, e quaisquer interferências que podem ter ocorrido durante a sessão.

*Indexação de Expressões.* Segundo, a transcrição é quebrada em pequenas expressões, cada uma identificada por um número indexador. Cada expressão deve corresponder à avaliação do engenheiro de conhecimento sobre uma parte do conhecimento, uma tarefa simples, afirmação, ou processo de coleta de dados pelo especialista.

*Codificação de Conhecimento.* Terceiro, o conhecimento é codificado. Essa atividade tenta classificar o conhecimento. Uma classificação proveitosa é distinguir entre conhecimento descritivo e procedural.

*Documentação.* Quarto, o conhecimento deve ser corretamente organizado e documentado. Uma maneira de organizar a documentação é dividi-la em quatro partes: lista compreensiva do domínio, descrição do conhecimento, conhecimento procedural e glossário. A documentação deve ser feita de uma maneira consistente. Formulários podem ser usados, e alguns vendedores de sistemas especialistas fornecem a seus clientes os formulários necessários.

*Diagramação de Conhecimento.* É uma abordagem gráfica para melhoramento do processo de aquisição de conhecimento. Ele consiste em uma descrição top-down hierárquica dos tipos de conhecimento usados para descrever fatos e estratégias de raciocínio para a resolução de problemas em sistemas especialistas. Esses tipos são objetos, eventos, performance, e meta-conhecimento. Diagramas também descrevem as ligações e interações entre os vários tipos de conhecimento.

### **3.9 Tabela Comparativa Entre os Métodos de Aquisição**

Abaixo temos uma tabela que mostra a diferença de características em relação a presença ou não do especialista e engenheiro de conhecimento nos métodos de aquisição de conhecimento.

<b>Métodos</b>	<b>Presença do Especialista</b>	<b>Presença do EC</b>
Métodos Manuais	SIM	SIM
Guiados pelo Especialista	SIM	SIM ou NÃO
Semi-Automáticos	SIM ou NÃO	NÃO ou SIM
Automáticos	NÃO	NÃO

**Figura 19:** Tabela Comparativa entre os métodos de aquisição de conhecimento

Os métodos manuais necessitam da presença do especialista e do engenheiro de conhecimento.

Nos métodos Guiados pelo Especialista, pode ou não haver a necessidade de um engenheiro de conhecimento. Apenas o especialista é necessário para executar a entrada de conhecimento correta no sistema.

Nos métodos Semi-Automáticos, ou o Especialista, ou o Engenheiro de Conhecimento é que são requisitados. Não é necessária a utilização de ambos. Caso exista o especialista, o engenheiro de conhecimento é descartado e caso exista o engenheiro de conhecimento, a função do especialista não é necessária.

Nos métodos automáticos, a presença de ambos o especialista e engenheiro de conhecimento são descartados.

### **3.10 Conclusão**

Neste capítulo foi apresentado o processo de engenharia do conhecimento, com todos os passos para a construção de um sistema baseado em conhecimento. Três tipos de métodos para aquisição de conhecimento (manual, automático e semi-automático) foram apresentados, com suas principais características.

Atualmente a falta de tempo e limitação dos recursos disponíveis podem ser considerados dois grandes fatores que dificultam o trabalho de engenheiros de conhecimento na construção de um sistema especialista. A utilização de métodos automáticos tem sido a melhor alternativa para a construção desses sistemas, requerendo menos tempo e gastos.

No próximo capítulo, apresentamos o *chatterbot* construído para aquisição automática de perfil de usuários através de diálogos em linguagem natural. No nosso caso, o usuário pode ser visto como o especialista, uma vez que é ele que detém a informação de se deseja adquirir (o seu perfil). Assim sendo, podemos dizer que utilizamos um método de aquisição guiado pelo especialista (seção 3.5), onde o *chatterbot* pode ser visto como uma ferramenta de auxílio para aquisição de perfil e o preenchimento da base de conhecimento (no nosso caso, a base de perfis). O usuário pode ser visto como um especialista que utiliza o *chatterbot* para a entrada de seu conhecimento, nesse caso informações sobre si mesmo, no computador.

## 4 Um Chatterbot em AIML para aquisição automática de perfil

---

O trabalho apresentado nesta dissertação teve como objetivos principais desenvolver um *chatterbot* capaz de manter diálogos sobre um domínio específico, e realizar aquisição automática de perfil dos usuários do sistema nesse domínio. A aquisição automática do perfil se dá com base nos diálogos travados entre o agente e o usuário. Durante esses diálogos, o sistema busca identificar informações relevantes nas sentenças do usuário para formar e atualizar o seu perfil, que poderá ser usado por sistemas de recomendação personalizada que venham a ser acoplado ao *chatterbot*.

Como visto no final do capítulo anterior, podemos classificar o método de aquisição utilizado aqui como sendo guiado pelo especialista (seção 3.5), onde o *chatterbot* pode ser visto como uma ferramenta de auxílio para aquisição de perfil e o preenchimento da base de dados. Esse perfil é o conhecimento sobre o usuário e não o que ele sabe, seu conhecimento. É guardado então informação e não conhecimento. No nosso caso, o usuário pode ser visto como o especialista, uma vez que é ele que detém a informação de se deseja adquirir (o seu perfil).

O domínio específico escolhido para o nosso projeto foi o de filmes, tendo como auxílio uma base de dados cedida pelos desenvolvedores do GEP (Guia de Entretenimento Personalizado). O GEP é um sistema de recomendação que mantém uma gigantesca base de filmes (contendo dados como nome do filme, diretor, idioma, atores e sinopse), e o perfil dos usuários cadastrados (contendo dados como nome, *login* e avaliação em relação a cada filme que foi sugerido ao usuário pelo sistema). Essas informações são utilizadas pelo sistema para fornecer recomendações personalizadas sobre filmes. Este sistema utiliza técnicas da Inteligência Artificial para aprender automaticamente sobre as preferências dos usuários [Bezerra et al. 2002]. O GEP foi desenvolvido por um convênio entre o CIn-UFPE e o CESAR (Centro de Estudos e Sistemas Avançados do Recife)<sup>14</sup>.

---

<sup>14</sup> <http://www.cesar.org.br/>

Atualmente, o GEP monta o perfil dos usuários através de uma sessão de aquisição baseada em menus. Cada usuário deve fornecer seus dados pessoais, e avaliar alguns filmes sugeridos pelo sistema durante a primeira sessão de uso. Os filmes sugeridos para todos os usuários, nesta primeira sessão são sempre os mesmos. A consulta ao GEP também é feita através de menus: o usuário solicita ao sistema uma recomendação de filme, que é feita com base no seu perfil.

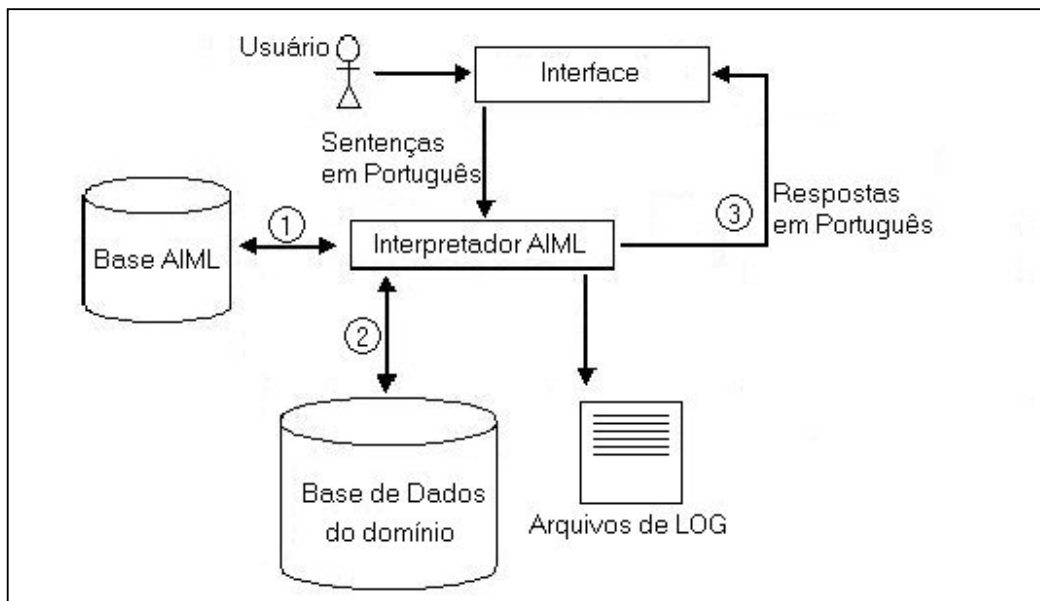
A integração do *chatterbot* desenvolvido ao GEP terá como principal contribuição a facilitação da interação entre usuário e sistema, tornando mais naturais tanto a aquisição do perfil quanto a consulta à base de filmes do sistema.

Não foi encontrado na literatura consultada nem na Web nenhum *chatterbot* com os mesmos objetivos e características que o desenvolvido neste trabalho. Isto atesta a originalidade do trabalho. Existem *chatterbots* que realizam aprendizado sobre um determinado domínio (e.g., [20q.net 2002]), mas não adquirem perfil dos usuários do sistema.

Nesse capítulo, veremos na seção 4.1 a arquitetura do sistema proposto, sendo explicado o seu funcionamento. Na seção 4.2, será apresentado o protótipo do projeto. Na seção 4.3, serão mostradas as novas tags AIML que foram implementadas para possibilitar o correto funcionamento do sistema. Na seção 4.4 teremos o sistema e como é o seu funcionamento. Na seção 4.5, os resultados obtidos. Por último, temos conclusões na seção 4.6.

## **4.1 Arquitetura do Sistema**

O sistema que implementa o *chatterbot* consta de vários módulos, todos cuidadosamente projetados e implementados com o objetivo de preservar a modularidade e a reusabilidade. A figura 4.1 mostra a arquitetura completa do sistema.



**Figura 4.1:** Arquitetura do sistema.

Na figura temos os números 1, 2 e 3 que correspondem ao fluxo de funcionamento do *chatterbot*. Logo que uma sentença é digitada pelo usuário, o interpretador busca na base AIML a categoria que casa com essa sentença digitada em (1). Depois do casamento com a categoria correspondente na base AIML, o interpretador sorteia uma resposta e, dependendo se a resposta contém ou não tags AIML, que iram consultar a base de dados, o passo (2) pode ou não vir a ser utilizado. Depois da resposta ser montada com sucesso, o interpretador a retorna ao usuário pela interface em (3).

Abaixo, cada módulo do sistema será descrito em detalhes.

#### 4.1.1 A Interface do sistema

A Interface foi toda codificada em HTML. É através dela que o usuário mantém seu diálogo com o *chatterbot*, primeiramente indicando o seu *login* ao sistema. As sentenças do usuário são inseridas através de uma barra onde ele digita suas perguntas e respostas.

Cada sentença de entrada é processada pelo interpretador AIML, que monta uma sentença para o *chatterbot* devolver ao usuário através da mesma interface. Essa interface também pode ser alterada de acordo com os padrões visuais que se queira ter.

#### 4.1.2 A Base de Dados do Domínio

Como já foi dito antes, a base de dados utilizada neste trabalho foi criada pelo grupo de desenvolvimento do GEP. Os dados contidos na base são utilizados durante o diálogo entre o robô e o usuário.

O banco de dados do GEP é composto por uma grande quantidade de tabelas e atributos que guardam informações sobre filmes e sobre o perfil de seus usuários. O nosso *chatterbot* utilizará apenas alguns dos dados sobre filmes contidos nesse banco (e.g., título, sinopse, diretores, atores, gênero). Algumas das informações disponíveis estão fora do escopo do nosso trabalho (e.g., país de origem do filme, idioma original e companhia produtora), e serão descartadas em nossa implementação. Além disso, o *chatterbot* utiliza uma das tabelas do banco para guardar os perfis dos usuários (*login*, avaliação sobre filmes, diretores e atores). Esta tabela será preenchida automaticamente pelo *chatterbot* através dos diálogos de aquisição de perfil.

Para a criação de um novo robô em um outro domínio específico, é necessário ter dados coerentes sobre esse domínio, que serão utilizados durante o diálogo com o usuário.

#### 4.1.3 A Base AIML ampliada

A base de categorias AIML utilizada pelo nosso *chatterbot* foi adaptada e ampliada a partir de um trabalho desenvolvido pelo Prof. André Neves, coordenador do laboratório de hipermídia da UFPE<sup>15</sup> e aluno de doutorado do CIn-UFPE. Neves criou uma base de categorias para diálogos em português a partir da base de categorias de ALICE, e utilizou essa base na criação de um *chatterbot* em AIML de propósito geral, o PixelBot<sup>16</sup>.

A base de categorias desenvolvida por Neves para o PixelBot trata sobre diversos assuntos, tais como música, livros, futebol, etc. Acrescentamos a essa base diversas categorias específicas sobre filmes (ver Apêndice A), a fim de possibilitar um

---

<sup>15</sup> <http://www.virtus.ufpe.br>

<sup>16</sup> <http://www.virtus.ufpe.br/pixelbot/>



diálogo natural com o usuário com o objetivo de aquisição de perfil. Além disso, foram também criadas novas tags, que realizam o acesso ao banco de dados sobre filmes, tanto para consulta de informações sobre filmes quanto para gravação do perfil do usuário (ver seção 4.3). A figura 4.2 abaixo mostra exemplos de categorias e tags para o domínio de filmes.

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>oi</pattern>
<template>
<srai>oi</srai>
</template>
</category>

<category>
<pattern>ola *</pattern>
<template>
<srai>oi</srai>
</template>
</category>
.
.
.
<category>
<pattern>oi</pattern>
<template>
<random>
<li>oi, <b><get name="name"/></b>. </li>
<li>oi <b><get name="name"/></b>, é muito bom encontrar você. gosto muito de filmes de ação e você ? gosta de que tipo de filmes ?</li>
<li>oi, espero que você converse bastante comigo sobre filmes. que tipo de filmes você mais curte ?</li>
<li>oi, que bom que você veio, <b><get name="name"/></b>. que tipo de filme você gosta ?</li>
<li>oi <b><get name="name"/></b>, vamos conversar sobre filmes ! vou começar perguntando que tipo de filmes você mais curte ?</li>
<li>olá, <b><get name="name"/></b>. qual gênero você mais gosta ?</li>
<li>olá, que bom que você está aqui, <b><get name="name"/></b>. que tipos de filmes você mais curte ?</li>
</random>
</template>
</category>

</aiml>

```

**Figura 4.2:** Arquivo de categoria “cumprimentar o robot”.

No exemplo acima, todas as sentenças digitadas pelo usuário que possuem a palavra “oi” ou “olá”, no início da sentença, são identificadas como sendo da categoria “cumprimentar o robot”. Depois da identificação, ocorre a seleção de uma das respostas que se encontram entre as tags: <template></ template >. Essas respostas serão devolvidas ao interpretador que, por sua vez, as retorna ao usuário.

Na figura 4.3, temos um exemplo de um arquivo de propriedades (*log*) contendo informações importantes para o sistema sobre o diálogo realizado.

```
#Mon Feb 17 11:06:50 GMT-03:00 2003
name=Tiago
that.5=<b> Tiago </b>, voc\u00EA acha que eu sou inteligente ? entendo tudo sobre filmes, mas ainda n\u00E3o
consigo entender o que voc\u00EA disse.
that.4=<b> Tiago </b>, voc\u00EA gostaria de conversar sobre filmes ?
that.3=\ Eu gostei muito de\ O Presente Do Dem\u00F4nio \! Que nota de 1 a 5 voc\u00EA daria a ele ?
that.2=<b> Tiago </b>, voc\u00EA poderia fazer mais algum coment\u00E1rio sobre esse assunto ?
input.5=panico?
that.1=como ? voc\u00EA est\u00E1 falando sobre filmes? voc\u00EA poderia reformular sua senten\u00E7a <b>
Tiago </b>.
input.4=panico
input.3=gosto de filmes de terror
dado=1
input.2=0
input.1=o filme e muito ruim
topic=referencia a filmes de terror
filme=23512
```

**Figura 4.3:** Arquivo de propriedades

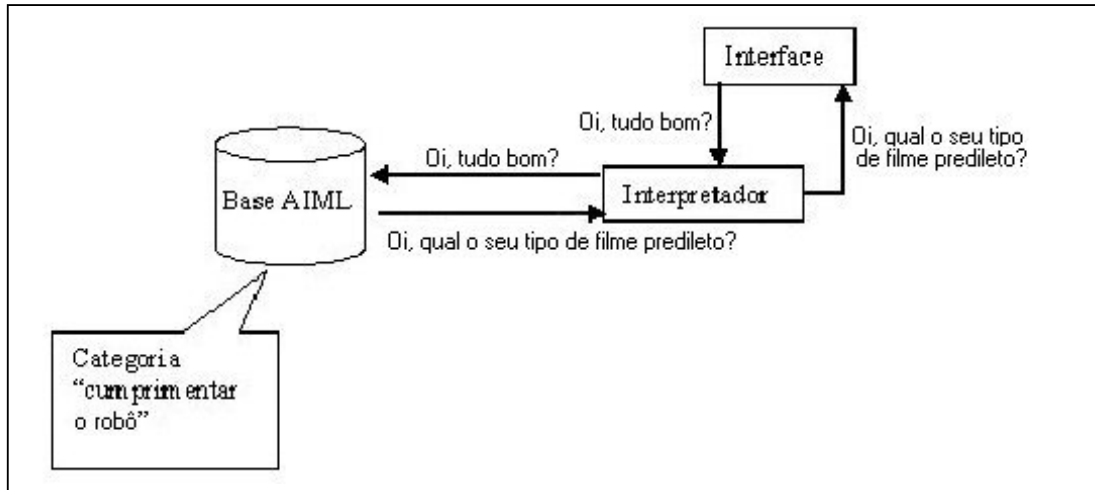
Algumas informações, como número de registro do filme em questão no diálogo (filme=23512), nome do usuário (name=Tiago) e sentenças digitadas, podem ser encontradas nesse arquivo.

Caso outro *chatbot* seja criado para outro domínio, novos arquivos de categorias devem ser inseridos com as sentenças necessários para o diálogo com o usuário.

#### 4.1.4 O Interpretador AIML

O interpretador AIML é um módulo fundamental para o funcionamento do *chatbot*. Ele tem como tarefa principal receber como entrada as sentenças digitadas pelos usuários, e buscar na base AIML uma categoria cujo padrão de entrada seja o mais adequado para a sentença atual. Em seguida, o interpretador monta a resposta que será devolvida ao usuário. Inicialmente, ele verifica se existem tags a serem processadas no template de saída dessa categoria. Em caso afirmativo, ele executa essas tags, buscando no banco de dados, quando necessário, informações adicionais para completar a sentença de saída. Por fim, o interpretador envia à interface a sentença montada a partir do

template de saída. A figura 4.4 mostra um exemplo de funcionamento do interpretador utilizado.



**Figura 4.4:** Exemplo de funcionamento do interpretador.

O interpretador utilizado neste trabalho estende o que está disponível para *download* no site dos criadores do AIML (o mesmo usado por ALICE). Foi necessário modificá-lo a fim de torná-lo capaz de: (1) processar as novas tags criadas para permitir acesso ao banco de filmes; (2) ser executado em qualquer máquina; e (3) ser acoplado a outra interface mais adequada ao nosso trabalho. Detalhes sobre a implementação desse interpretador serão mostrados na seção 4.2.4.

## 4.2 O Protótipo

Nesta seção, mostraremos os detalhes de implementação do nosso chatterbot. Veremos como os módulos de arquitetura apresentada na seção anterior foram implementados, e que tecnologias foram usadas ao longo de todo o projeto. Esses módulos estão fisicamente armazenados em diretórios separados, embaixo de um diretório principal contendo arquivos de configuração do interpretador.

### 4.2.1 A interface

A interface foi toda criada em HTML, como já dito anteriormente, sendo controlada por servlets Java. Através dela, as sentenças digitadas pelos usuários dão entrada no sistema e são processadas pelo interpretador, que retorna à interface a sentença "resposta" do robô. A cada nova sentença do usuário e resposta do robô, a tela do browser é atualizada, e essas sentenças são mostradas em conjunto com as outras já digitadas em interações anteriores.

#### **4.2.2 A Base de Dados sobre Filmes**

A base de dados original do sistema GEP está implementada como um banco de dados IBM DB2, na versão *Professional*, que é uma ferramenta de banco de dados *freeware* da IBM. Assim sendo, nossa escolha inicial foi manter esse mesmo sistema. Contudo, devido às muitas dificuldades encontradas para a instalação e o uso do DB2, os dados do banco original foram exportados para o sistema Microsoft Access, que oferece maior facilidade de manuseio e manutenção. Todas as tabelas do banco original foram exportadas e organizadas da mesma forma como se encontravam no DB2, para que o nosso sistema pudesse trabalhar no mesmo padrão que o GEP. O objetivo foi garantir uma futura inclusão do *chatbot* a esse sistema, sem que haja grandes mudanças no código já implementado.

A única mudança realizada na estrutura do banco original foi a criação de uma nova tabela para guardar perfis dos usuários. A tabela que guarda os perfis dos usuários no banco original possui apenas três atributos: código de registro do usuário, código de registro do filme, e avaliação dos filmes sugeridos pelo sistema. Contudo, o *chatbot* também dialoga e adquire informações sobre as avaliações do usuário em relação a diretores e atores. Assim, foi necessário criar uma outra tabela, onde os dados avaliados sobre filmes, diretores e atores são guardados.

#### **4.2.3 A Base AIML ampliada**

Como dito anteriormente, a base AIML é fisicamente armazenada em um diretório separado. Uma vez que temos mais de 4000 categorias individuais, decidimos

separá-las por “tipo” e armazená-las em arquivos menores dentro do diretório, a fim de facilitar a manutenção e atualização da base AIML.

Categorias referentes a cada tipo de filme (e.g., ação, aventura) estão agrupadas em arquivos separados, um para cada tipo de filme. Categorias referentes a diretor de filme estão em outro arquivo separado, assim como acontece com as categorias sobre atores (ver Apêndice A).

As categorias herdadas do PixelBot estão também separadas em diversos arquivos. Aqui, mantivemos a mesma organização utilizada na base original do PixelBot, onde as categorias são classificadas de acordo com o tipo de “intenção” do usuário (e.g., cumprimentar o robô, perguntar sobre algo, elogiar o robô, etc...) [Neves et al. 2002].

As categorias que compõem a base AIML ampliada são armazenadas em arquivos texto, podendo ser atualizadas ou modificadas através de qualquer editor de texto convencional.

#### 4.2.4 O Interpretador AIML

O interpretador foi todo escrito em Java, com o código aberto, sendo composto por: arquivos de configuração, arquivos contendo as definições das tags existentes no sistema, arquivos que registram as tags, classes referentes à estrutura do servidor utilizado, classes referentes à interface do sistema, entre outros.

Depois de se verificar o código do interpretador AIML original, constatou-se que alguns arquivos deveriam ser modificados para que o interpretador pudesse processar as tags criadas (seção 4.3), com novas classes Java implementando cada uma das novas tags. Dos muitos arquivos encontrados no interpretador original, apenas dois tiveram que ser alterados para o acréscimo e registro das novas tags: o *AIMLProcessorRegistry.java* e o *AIMLTag.java*. Esses arquivos são apresentados abaixo:

- *AIMLProcessorRegistry.java*: essa classe é responsável pelo registro de todas as tags que o interpretador possui. Aqui, encontram-se os endereços dos arquivos que guardam as definições das tags existentes no sistema. Acrescentamos aqui os endereços dos arquivos das novas tags criadas.

- *AIMLTag.java*: essa classe é responsável por processar as tags. É nessa classe onde está definido a sintaxe de cada tag, isto é, todas as possibilidades de representação que a tag pode assumir (por exemplo, `<star></star>` ou `<star/>`).

Depois de inserir as referências às novas tags nesses dois arquivos de controle, foram criados arquivos de extensão Java definindo o modo de processamento de cada nova tag (Apêndice B). Os novos arquivos criados seguiram a mesma estrutura de todos os arquivos que definem tags, contendo um verificador, que é responsável por avaliar se a tag que está sendo interpretada se encontra bem formada e se está definida dentro do sistema. Caso alguma dessas situações não se verifique, o sistema não reconhece a tag e não a executa, ignorando-a e retornando para o usuário apenas o texto que se encontra no template de resposta da categoria selecionada. Abaixo temos um exemplo do uso de uma tag não definida no sistema (figura 4.5):

```
<category>
<pattern>defina homens</pattern>
<template>ok, falemos de homens. mas, já vou avisando, <getname="name"/>, esta não é minha
    especialidade.</template>
</category>
```

**Figura 5:** Exemplo de tag não definida

No exemplo acima, a tag `<get name="name"/>` foi escrita com a sintaxe errada, sem a utilização de espaço em branco entre as palavras *get* e *name*. Assim, o sistema não conseguirá interpretá-la, e retornará ao usuário uma resposta sem definir o nome do usuário, como deveria ter ocorrido.

Se a tag for escrita corretamente e registrada no sistema, será executado todo o processamento proposto por essa tag, e um string (neste caso, o nome do usuário) substituirá a tag, retornando para o usuário uma sentença completa.

Algumas das tags criadas fazem acesso ao banco de dados do domínio para obter a informação que irá compor o restante do template que a contém.

Assim, nossos arquivos Java que definem as tags devem manter a mesma estrutura de outros arquivos de tags, isto é, conter um verificador e pesquisar se a tag está definida no sistema.

### 4.3 Tags criadas

Nesta seção, veremos detalhes sobre todas as novas tags criadas para esta aplicação. Algumas delas realizam consultas ao banco e retornam informações que são utilizadas em complemento a algum template encontrado na base AIML. Outras têm como função acrescentar alguma informação ao perfil do usuário, como por exemplo, a nota dada pelo usuário a um filme, diretor ou ator.

As novas tags foram escritas em Java, já que o interpretador AIML é todo escrito nessa linguagem, com a utilização do JDBC para acesso ao banco de dados. Os arquivos definindo o modo de processamento de cada nova tag são apresentados no Apêndice B.

#### Tag BuscaFilme

Essa tag é responsável por buscar no banco de dados um exemplo de filme referente ao estilo preferencial do usuário corrente. Quando se identifica, durante o diálogo, qual é o estilo de filme preferido pelo usuário, o robô busca no banco de dados, aleatoriamente, algum filme nesse estilo e prossegue com o diálogo.

A sintaxe desta tag é: `<buscafilme>TipoDeFilme</buscafilme>`. Deve existir no sistema uma tag BuscaFilme para cada tipo de filme no banco de dados (e.g., `<buscafilme>Ação</buscafilme>`, `<buscafilme>Aventura</buscafilme>`, etc).

No caso do diálogo a seguir (figura 4.6), a tag `<buscafilme>Ação</buscafilme>` procura no banco de dados, aleatoriamente, algum filme no estilo Ação, para que o *chatbot* possa conversar com o usuário sobre esse estilo de filme (figura 4.7).

<b>robô:</b> oi usuário, vamos conversar sobre filmes! vou começar perguntando que tipo de filmes você mais curte!
--

<p><b>usuário:</b> ação <b>robô:</b> você gostou de: Velocidade Máxima? Que nota de 1 a 5 você daria a ele ? <b>usuário:</b> 4</p>
--

Figura 4.6: Exemplo de sentença com estilo de filme isolado

Veja a seguir exemplos de categorias referentes ao tipo de filme “Ação”:

```
<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>ação</pattern>
<template>
<srai>acao</srai>
</template>
</category>

<category>
<pattern>adoro filmes de ação</pattern>
<template>
<srai>acao</srai>
</template>
</category>
...
<category>
<pattern>acao</pattern>
<template>
<random>
<li><b><get name="name"/></b>, qual a nota de 1 a 5 que você daria a: <buscafilme>Ação</buscafilme> ?</li>
<li>Eu gostei muito de: <buscafilme>Ação</buscafilme> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>filmes de ação são muito bons, eu os adoro! voce gostou de: <buscafilme>Ação</buscafilme> ? Que nota de 1 a 5 você daria a ele ?</li>
<li>filmes de ação são muito movimentados! voce gostou de: <buscafilme>Ação</buscafilme> ? Que nota de 1 a 5 você daria a ele ?</li>
<li>são os meus prediletos ! voce gostou de: <buscafilme>Ação</buscafilme> ? Que nota de 1 a 5 você daria a ele ?</li>
<li>voce gostou de: <buscafilme>Ação</buscafilme> ? Que nota de 1 a 5 você daria a ele ?</li>
</random>
</template>
</category>

</aiml>
```

Figura 4.7: Arquivo de categoria: “filme ação”

Foi necessário criar categorias específicas para cada estilo de filme. Certamente, seria mais simples e elegante criar apenas uma categoria para processar o estilo de filme escolhido pelo usuário, sendo esse valor passado como um parâmetro. Contudo, isto não foi possível, uma vez que uma sentença do usuário contendo apenas o estilo do filme



(digitada, por exemplo, como resposta a uma pergunta do robô sobre o estilo preferido do usuário – figura 4.6) iria “casar” com o padrão de entrada `<pattern>*</pattern>`, que já é utilizado pelo interpretador AIML na identificação da categoria “desconhecido”. Essa categoria casa com sentenças digitadas pelo usuário que não são reconhecidas por nenhuma outra categoria na base AIML, retornando alguma resposta *default*, a fim de manter a continuidade do diálogo.

Assim sendo, um diálogo como o mostrado na figura 4.6 não seria processado corretamente caso não existissem categorias individuais para cada estilo de filme. O estilo de filme informado seria ignorado, não sendo utilizado para buscar um exemplo na base de filmes e, assim, dar andamento ao diálogo de aquisição de perfil.

Apenas para ilustrar a discussão, veremos a seguir como todos os estilos de filme poderiam ser tratados em um mesmo arquivo de categorias com o auxílio do "\*" e da tag `<srai>`, caso não houvesse problemas com a categoria “desconhecido” (figura 4.8).

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>adoro filmes de *</pattern>
<template>
<srai><star/></srai>
</template>
</category>

<category>
<pattern>amo filmes de *</pattern>
<template>
<srai><star/></srai>
</template>
</category>
...
<category>
<pattern>ação</pattern>
<template>
<random>
<li>são os meus prediletos, você gosta muito de filmes de ação ?</li>
...
<li>voce gostou de: <buscafilme>Ação</buscafilme> ? Que nota de 1 a 5 você daria a ele ?</li>
</random>
</template>
</category>
...
</aiml>

```

**Figura 4.8:** Arquivo de categoria “estilos de filmes”

Na figura 4.8, podemos verificar que as categorias que tratam padrões de entradas não atômicas utilizam a tag `<srai>` para finalizar o processamento da entrada. Todas essas categorias passam adiante (para o interpretador) apenas o estilo do filme (e.g., ação, drama), uma vez que é essa informação que determina o tipo do filme a ser recuperado do banco de dados (utilizado para preencher o template de saída a ser devolvido ao usuário). Assim sendo, mesmo quando a entrada não é atômica, contendo mais informação do que apenas o estilo de filme, não é possível cobrir todos esses estilos de uma só vez (i.e., em um mesmo arquivo de categorias) sem o uso do padrão `<pattern>*</pattern>`. No exemplo da figura 4.8, a solução adotada foi tratar somente filmes de ação, incluindo uma categoria com o padrão `<pattern>ação</pattern>`, que devolve templates apenas sobre esse estilo de filme.

Devido a essa limitação, não foi possível tratar em um único arquivo de categorias todas as variações de entradas atômicas, como “Ação”, “Aventura”, “Suspense”, etc. Por causa disso, houve a separação do arquivo inicial de categorias “estilos de filmes” em 21 arquivos diferentes, cada um referente a um estilo de filme disponível na base de dados.

### **Tag Sinopse**

Esta tag tem a função de retornar para o usuário a sinopse do filme em discussão. Caso o robô retorne algum filme desconhecido pelo usuário e este pergunte, por exemplo, “Que filme é esse?”, ou “Como é esse filme?”, o robô automaticamente retorna a sinopse do filme em questão.

A sintaxe dessa tag é: `<sinopse>sinopse</sinopse>`. A figura 4.9 abaixo traz um exemplo do uso dessa tag (arquivo de categorias “sinopse de filme”).

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>como e esse filme</pattern>
<template>
<srai>sinopse</srai>
</template>
</category>
...
<category>
<pattern>sinopse</pattern>
<template>
<random>
<li>A descrição dele é: <sinopse>sinopse</sinopse> </li>
<li>Ele é mais ou menos assim: <sinopse>sinopse</sinopse> </li>
<li>Fala sobre: <sinopse>sinopse</sinopse> </li>
</random>
</template>
</category>

</aiml>

```

**Figura 4.9:** Arquivo de categoria: “sinopse de filme”

Depois de retornar a sinopse do filme, o diálogo com o usuário prossegue, podendo incluir perguntas sobre, por exemplo, o diretor do filme.

### **Tag Diretor**

A tag Diretor é responsável por retornar ao usuário o diretor do filme sobre o qual se está dialogando. O sistema pode guardar no perfil do usuário a nota dada a esse diretor pelo usuário (através da tag “NotaDada”), para uma futura indicação de filmes de acordo com o diretor que melhor foi avaliado pelo usuário.

Notas de 1 a 5 podem ser dadas aos diretores, e essas informações são armazenadas no banco de dados juntamente com o filme correspondente.

A sintaxe dessa tag é: <diretor>diretor</diretor>. A figura 4.10 abaixo traz um exemplo do uso dessa tag (arquivo de categorias “diretor de filme”).

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>qual e o diretor</pattern>
<template>
<srai>diretor</srai>
</template>
</category>

<category>
<pattern>qual e o diretor desse filme</pattern>
<template>
<srai>diretor</srai>
</template>
</category>
...
<category>
<pattern>diretor</pattern>
<template>
<random>
<li>Ele é o: <diretor>diretor</diretor> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>É o: <diretor>diretor</diretor> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>O diretor é: <diretor>diretor</diretor> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>O nome dele é: <diretor>diretor</diretor> ! Que nota de 1 a 5 você daria a ele ?</li>
</random>
</template>
</category>

</aiml>

```

**Figura 4.10:** Arquivo de categoria: “diretor de filme”

## Tag Ator

A tag Ator tem a função de retornar ao usuário algum ator do filme sobre o qual se está dialogando. Como no caso da tag Diretor, aqui o sistema também pode armazenar, através da tag “NotaDada”, as avaliações dadas aos atores pelos usuários, para uma futura recomendação de filme com base nessas informações. Assim também como no caso anterior, notas de 1 a 5 podem ser dadas aos atores, e essas informações são guardadas no perfil de usuários.

A sintaxe dessa tag é: <ator>ator</ator>. A figura 4.11 abaixo traz um exemplo do uso dessa tag (arquivo de categorias “ator de filme”).

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>qual e o ator</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>qual e o ator principal</pattern>
<template>
<srai>ator</srai>
</template>
</category>

...
<category>
<pattern>ator</pattern>
<template>
<random>
<li>Ele é o: <ator>ator</ator> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>É o: <ator>ator</ator> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>O ator é: <ator>ator</ator> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>O nome dele é: <ator>ator</ator> ! Que nota de 1 a 5 você daria a ele ?</li>
</random>
</template>
</category>

</aiml>

```

**Figura 4.11:** Arquivo de categoria: “ator de filme”

## Tag NotaDada

Essa tag realiza a função de gravar no perfil dos usuários as notas dadas por eles para os filmes, diretores e atores. Ao contrário das outras tags, esta tem apenas a função de inserir informações no banco de dados, e não de realizar consultas para retorno de algum resultado.

A sintaxe dessa tag inclui as seguintes variações: <notadada>1</notadada>, <notadada>2</notadada>, <notadada>3</notadada>, <notadada>4</notadada> ou <notadada>5</notadada>. Assim como no caso da tag BuscaFime, aqui também não foi possível tratar todas as notas em um único arquivo de categorias, uma vez que isso iria requerer o uso do padrão <pattern>\*</pattern>. Assim sendo, foram criados os arquivos

de categoria “nota dada 1”, “nota dada 2”, “nota dada 4” e “nota dada 5”. Abaixo temos um exemplo de utilização dessa tag no arquivo de categorias “nota dada 3” (figura 4.12).

```
<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>3</pattern>
<template>
<srai>3</srai>
</template>
</category>

<category>
<pattern>acho que 3</pattern>
<template>
<srai>3</srai>
</template>
</category>

....
<category>
<pattern>3</pattern>
<template>
<random>
<li>Boa nota! <notadada>3</notadada></li>
<li>Obrigado por ter dado uma nota, <b><get name="name"/></b>! <notadada>3</notadada></li>
<li>Ok ! <notadada>3</notadada></li>
<li>Valeu! <notadada>3</notadada></li>
</random>
</template>
</category>

</aiml>
```

**Figura 4.12:** Arquivo de categoria: “nota dada 3”

Depois de armazenar a nota no banco de dados, o diálogo prossegue normalmente, e mais avaliações podem ser feitas pelos usuários sobre outros filmes, atores ou diretores.

#### 4.4 Testes e exemplo de uso do chatterbot

Os testes com o *chatterbot* para aquisição de perfil foram realizados com o auxílio de cerca de 20 usuários. Vários diálogos bem diversificados serviram de base ao

nosso estudo, para que pudéssemos inicialmente melhorar o grau de acerto e coerência do nosso *chatterbot* e, mais tarde, testar seu desempenho. Na fase de ajustes do robô, os erros identificados nos diálogos serviram de guia para modificarmos a base de falas, bem como para adicionarmos novas tags ao sistema.

Diálogos com usuários variados, como estudantes do CIn-UFPE (graduandos, mestrandos e doutorandos), e usuários externos (adolescentes e adultos), serviram de base para os testes realizados ao longo dessa fase, que foram importantes para que fosse possível avaliarmos o nosso sistema com maior precisão.

#### 4.4.1 Iniciando o diálogo com o Robô

A primeira tela da interface do robô solicita ao usuário que digite algum nome, para então iniciar o seu diálogo com o robô. Essa primeira tela serve de entrada ao sistema, pois a partir desse ponto, o usuário já terá um nome, que será associado a ele ao longo de todo o diálogo. Esse nome também será guardado no banco de dados, caso o usuário avalie algum filme, ator ou diretor que o sistema o apresentou, iniciando a montagem do seu perfil.

Essa primeira tela é bastante simples, e consta de uma barra de texto onde o usuário digitará o seu nome e entrará no sistema. A primeira tela é mostrada na figura 4.13.



**Figura 4.13:** Tela de entrada do sistema

A segunda tela apresenta o diálogo do usuário com o robô. É aqui que o usuário digita suas sentenças (perguntas e respostas) e recebe outras sentenças do robô. A aquisição de perfil é, então, realizada com base nesse diálogo. Na figura 4.14, é mostrada a tela de diálogo.



**Figura 4.14:** Tela de Diálogos

#### 4.4.2 O Processamento dos diálogos

Muitos diálogos foram realizados e registrados no log do sistema, e alguns serão utilizados como exemplos para podermos demonstrar todo o funcionamento e processamento do sistema.

Na figura 4.15, temos um exemplo de diálogo entre um usuário e o robô.



1. **robô:** oi usuário , eu sou um assistente virtual para conversar na internet sobre filmes. atualmente, estou tentando melhorar o meu português, e você pode me ajudar falando comigo sobre cinema e filmes.
2. **usuário:** oi robô!
3. **robô:** oi usuário, vamos conversar sobre filmes! vou começar perguntando que tipo de filmes você mais curte!
4. **usuário:** aventura
5. **robô:** filmes de aventura às vezes são muito engraçados! Eu gostei muito de “Os Contrabandistas”! Que nota, de 1 a 5, você daria a ele?
6. **usuário:** nunca assisti!
7. **robô:** eu não tenho certeza de que estou entendendo você completamente.
8. **usuário:** como é esse filme?
9. **robô:** Fala sobre: Durante a vigência da lei seca, dupla de aventureiros dividem os riscos de contrabandear bebidas pela fronteira no barco Lucky Lady, e o amor da dançarina Claire.
10. **usuário:** deve ser legal!
11. **robô:** você se sente preparado para discutir sobre filmes ?
12. **usuário:** dou nota 4
13. **robô:** Muito boa nota!

**Figura 4.15:** Diálogo entre o robô e um usuário

Ao entrar no sistema, o robô inicia o diálogo se apresentando e mostrando ao usuário qual assunto é sua especialidade. Assim, o usuário será informado de que o robô apenas conversa sobre um determinado domínio, que no nosso caso é sobre filmes (sentença 1). O diálogo prossegue, e o usuário cumprimenta o robô (sentença 2). Em seguida, o interpretador identifica que a sentença digitada pertence à categoria cuja intenção é “cumprimentar o robô”, e alguma resposta é escolhida e retornada ao usuário (sentença 3) para que o diálogo prossiga.

Em (4), o usuário responde a pergunta feita anteriormente, informando qual estilo de filme ele prefere. Com isso, o arquivo de categoria em AIML “filmes\_de\_aventura” é acessada pelo sistema, identificada pelo padrão de entrada digitado em (4), e é selecionada alguma das respostas disponíveis nessa categoria (sentença 5). Todas as respostas contidas nessa categoria fazem acesso ao banco de dados sobre filmes, através da tag <buscafilme>Aventura</buscafilme>, que retorna algum filme nesse estilo. O número de registro do filme escolhido é guardado em um arquivo de *log*, o arquivo de propriedades, contendo dados relevantes do usuário (e.g., nome, *log* dos diálogos). Esse arquivo é criado quando o usuário entra no sistema pela primeira vez.

Depois que o robô retorna ao usuário alguma das respostas contidas no arquivo de categorias “filmes\_de\_aventura”, o usuário responde em (6). O robô não possui nenhuma categoria que contenha o padrão de entrada digitado pelo usuário em (6), e responde em (7) com algum dos padrões de resposta da categoria “desconhecido”, onde o robô guarda respostas simples para padrões desconhecidos.

Em (8), o usuário continua a interação e pergunta como é o filme que o robô indicou. Em (9), o robô acessa a categoria “sinopse\_dos\_filmes” e faz uma busca no banco de dados de filmes através da tag <sinopse>sinopse<sinopse>, no campo DS\_DESC da tabela PROGRAM (figura 4.16), retornando a descrição do filme contida no banco de dados, com alguma sentença que completa a sua resposta com a sinopse do filme.

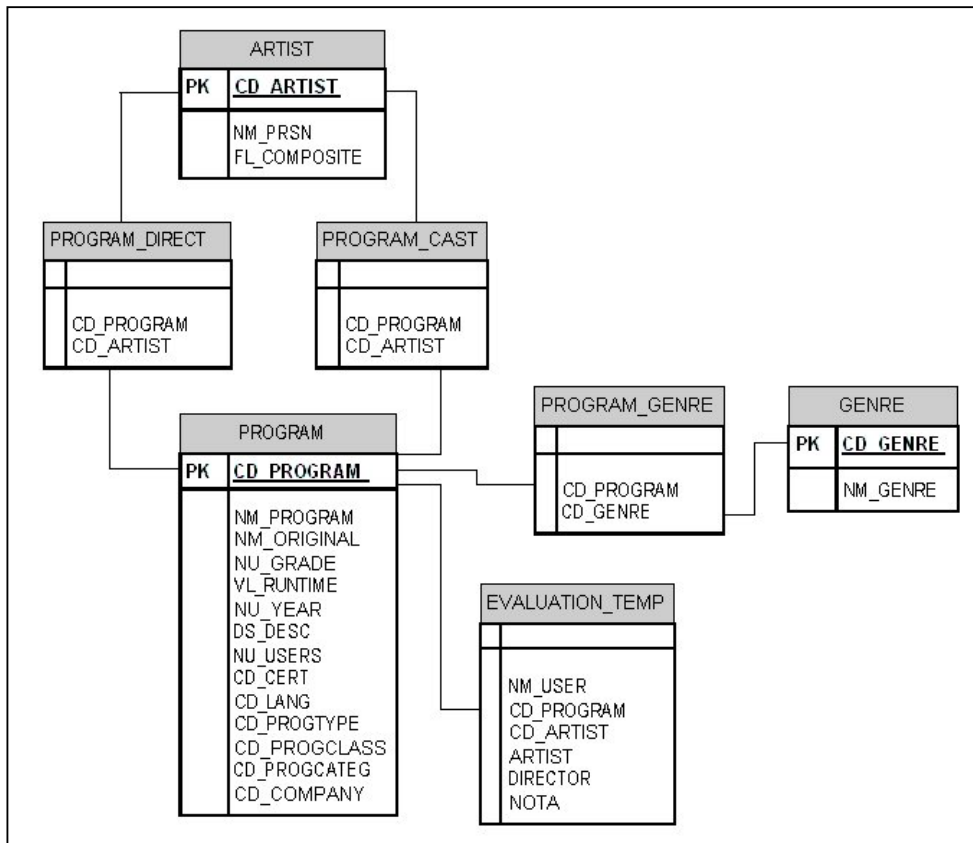
O usuário continua o diálogo com o robô (sentença 10), mas novamente o robô não encontra em nenhuma de suas categorias o padrão digitado pelo usuário. Novamente, ele busca uma saída na categoria “desconhecido” (sentença 11).

Em (12), o usuário dá uma nota ao filme mencionado anteriormente, e finalmente em (13) o robô identifica o padrão digitado pelo usuário em (12) como sendo da categoria “nota\_dada\_4”, e retorna algumas das respostas contidas nesse arquivo, acessando posteriormente o banco de dados para armazenar na tabela EVALUATION\_TEMP (figura 4.16) o nome do usuário, número de registro do filme e a nota dada.

#### **4.4.3 O Banco de Dados**

Durante o processamento dos diálogos, apenas a tabela EVALUATION\_TEMP é disponibilizada para armazenamento de informações. É a única tabela que o sistema utiliza para guardar os perfis dos usuários. Todas as outras tabelas são usadas apenas para consulta e utilização de seus dados para montar os diálogos do robô.

Na figura 4.16, é mostrada a estrutura do banco de dados utilizado pelo nosso *chatbot*. O banco de dados mantido pelo sistema GEP é bem mais extenso, mas aqui apenas as tabelas utilizadas pelo *chatbot* são mostradas.



**Figura 4.16:** Tabelas do Banco de dados

A tabela PROGRAM contém todos os cadastros de filmes com suas descrições e outras informações relevantes. Na tabela ARTIST, estão cadastrados todos os atores e diretores encontrados em todos os filmes da tabela PROGRAM. A tabela PROGRAM\_CAST relaciona todos os filmes com os seus respectivos atores, e a tabela PROGRAM\_DIRECT relaciona todos os filmes com seus diretores.

A tabela GENRE contém o cadastro de todos os estilos de filmes disponíveis na tabela PROGRAM. A tabela PROGRAM\_GENRE relaciona os filmes com os seus estilos de gênero. Finalmente, a tabela EVALUATION\_TEMP guarda os perfis dos usuários adquiridos durante os diálogos.

#### 4.5 Resultados obtidos

Como dito na seção anterior, o *chatbot* protótipo foi avaliado através da análise dos *logs* de diálogos de usuários do CIN-UFPE (graduandos, mestrandos e doutorandos), e usuários externos (adolescentes e adultos). Foram cerca de 20 usuários

que utilizaram o *chatterbot* que rodava em máquinas nos laboratórios do CIn. Todos os *logs* eram guardados em um diretório separado na máquina onde o *chatterbot* estava em funcionamento. O uso do sistema pelos usuários era realizado de forma individual e só depois de alguns dias da divulgação, aos usuários, sobre a existência do *chatterbot* para que fosse testado, eram recolhidos os *logs* para análise. Os resultados apresentados aqui foram medidos de forma manual, com a análise individual de cada *log*.

Como já dito, a base de categorias utilizada neste protótipo foi oriunda do trabalho de Neves. A base contém falas sobre temas variados, uma vez que o robô original é de propósito geral. Porém, essa base é relativamente pequena, e assim, algumas sentenças do usuário não são reconhecidas pelo sistema, sendo enquadradas na categoria “desconhecido”.

O trabalho desenvolvido aqui estendeu essa base original, incluindo categorias sobre o domínio de filmes apenas, e com um objetivo explícito de realizar aquisição de perfil. Assim sendo, a análise de desempenho realizada por nós considerou apenas as intenções onde o tema era sobre o domínio de filmes. Essa análise procurou medir a taxa de acerto do robô na tarefa de aquisição de perfil.

Seguindo esse critério, observamos que cerca de 50% das sentenças do usuário contendo informações relevantes à aquisição de perfil foram corretamente processadas pelo robô, e as informações fornecidas foram corretamente armazenadas no BD, montando assim o perfil do usuário.

Em muitos *logs* observados, constata-se que os usuários fizeram perguntas ao robô, ao invés de deixarem que o robô comandasse o diálogo. Esse foi um dos fatores que contribuiu para o desempenho do robô não ter sido melhor. Este problema pode ser facilmente resolvido pela inclusão de novas categorias que tratam os padrões de perguntas dos usuários, sendo capazes de responder a essas perguntas, bem como de extrair informações a partir delas. Infelizmente, não houve tempo de incluir essas novas categorias ainda. Assim sendo, esse aumento da base de categorias fica indicado como trabalho futuro.

Apesar desse problema, que tem soluções facilmente implementáveis, o *chatterbot* realizou de forma correta sua tarefa principal, que é a aquisição de perfil do usuário. Observando os *logs* dos diálogos, verificamos que a aquisição de perfil foi

realizada com sucesso em todos os casos onde os usuários deixaram o robô guiar o diálogo, como no exemplo da figura 4.15, tendo o seu perfil guardado de forma correta. Quando era verificado durante os *logs* que o sistema realizara alguma pergunta com a intenção de aquisição de perfil e o usuário interagiu de forma correta respondendo de acordo com o que o fora perguntando, geralmente com alguma nota, era também verificado no banco de dados se de fato a nota do usuário para aquele filme, ator ou diretor havia sido guardada, e em todos os casos isso ocorreu de forma correta.

## 4.6 Conclusões

Vimos neste capítulo a descrição geral do *chatterbot* criado para aquisição de perfil no domínio de filmes, incluindo sua arquitetura, detalhes de implementação, as tags criadas para esse domínio, e um exemplo de uso do sistema. O robô apresentou um desempenho satisfatório na tarefa de aquisição de informação (por volta de 50%). Para melhorar seu desempenho, seria necessário aumentar a base AIML com categorias contendo novos padrões de entrada que atualmente não são reconhecidas. Como já foi comentado, isso será realizado em trabalhos futuros.

Um *chatterbot* com as mesmas características que este não foi encontrado na literatura consultada nem na Web. Assim sendo, acreditamos que o nosso trabalho é o único a tratar aquisição de perfil de usuário através de diálogos em português.

Esse *chatterbot* será incorporado ao sistema GEP, a fim de possibilitar a aquisições automática de perfil através de diálogos em português, facilitando assim a interação do usuário com o sistema GEP. Algumas informações adquiridas pelo robô, como as notas dadas sobre diretores e atores, não são atualmente tratadas no sistema GEP, que só considera notas dadas sobre filmes. Isso tem dificultado a elaboração de uma recomendação mais exata pelo GEP. Assim sendo, a intenção dos desenvolvedores do GEP é incorporar as novas informações adquiridas pelo robô ao sistema, a fim de melhorar sua precisão.

Não foi possível incorporar ao robô a função de, através de diálogos, recomendar filmes a partir do BD existente, uma vez que não tivemos acesso ao módulo de processamento de recomendação implementado pelo sistema GEP. Essa extensão, que

possibilitaria que toda a interação com o GEP fosse feita através de diálogos em português, está indicada como trabalho futuro no capítulo de conclusão.

Além dos resultados práticos obtidos com a implementação do *chatterbot*, esse trabalho trouxe também muito aprendizado, em especial sobre o JDBC do Java, sobre o banco de dados IBM DB2 e Microsoft Access, e sobre o interpretador AIML, que teve seu código alterado para lidar com as novas categorias e tags criadas. Apesar das dificuldades encontradas durante o desenvolvimento do projeto, conseguimos concluir as metas que foram propostas para o nosso sistema.

## 5 Conclusão

---

Esta dissertação de mestrado apresentou um sistema capaz de dialogar com os seus usuários sobre um domínio específico e realizar aquisição automática do perfil desses usuários. Para isto, o sistema utiliza um banco de dados sobre filmes e perfis de usuários, contando também com uma base de falas construída com o auxílio da linguagem de marcação AIML. Tratando-se de um protótipo inicial, o *chatterbot* apresentou um desempenho satisfatório na realização da tarefa principal para a qual ele foi proposto: aquisição de perfil.

Veremos a seguir as principais contribuições deste trabalho, as maiores dificuldades encontradas em seu desenvolvimento, e indicações de trabalhos futuros.

### 5.1 Contribuições

A principal contribuição deste trabalho foi, sem dúvida, a construção do *chatterbot* de aquisição automática de perfil através de diálogos. Como já dito, trata-se de um trabalho original, cuja relevância pode ser constatada tanto na área de *chatterbots* quanto na de aquisição automática de perfil.

Outra contribuição relevante é a possibilidade real de o *chatterbot* ser acoplado ao sistema GEP, que atualmente se comunica com seus usuários através de menus. Assim, os usuários do GEP poderão se comunicar com o sistema através de diálogos, tornando a interação mais natural e simples. Além disso, o *chatterbot* também adquire as preferências dos usuários em relação a diretores e atores, incluindo essa informação no perfil de cada usuário. Essa nova informação irá possibilitar uma análise mais detalhada das preferências do usuário, a fim de que o GEP possa realizar recomendações mais exatas e coerentes.

A base de diálogos do *chatterbot* foi implementada em AIML, a tecnologia mais moderna e bem sucedida para construção desse tipo de sistema. Além de ser extensível, possibilitando a fácil inclusão de novas categorias, essa base poderá também ser utilizada

por outros sistemas neste mesmo domínio (com ou sem o objetivo de fazer aquisição de perfil). Por fim, a nosso trabalho pode servir de exemplo para a construção de bases para outros sistemas em domínios diferentes.

O sistema foi projetado buscando modularidade, extensibilidade e facilidade de uso. Foi utilizado um interpretador desenvolvido em Java, sendo essa linguagem orientada a objetos, portátil, e que permite a modularidade e a extensibilidade. Além disso, Java é altamente adequada para o desenvolvimento de aplicações na Web.

Por fim, citamos também a apresentação de um estudo detalhado dos vários tipos de *chatbots* encontrados na Web e as tecnologias utilizadas em seu desenvolvimento ao longo dos anos (Capítulo 2). O estudo sobre aquisição de conhecimento (Capítulo 3) trouxe experiência nessa área, possibilitando a construção adequada do nosso *chatbot*. A experiência adquirida nessas áreas será útil para a construção de outros sistemas que incorporem essa técnica em sua implementação.

## **5.2 Dificuldades Encontradas**

### **Banco de Dados utilizado**

Como já mencionado anteriormente, o banco de dados sobre filmes foi criado inicialmente com a ajuda do SGBD da IBM, o DB2. Devido a muitos problemas para utilizar esse banco de dados (tais como encontrar uma versão adequada para trabalhar com o banco, já que outras versões eram incompatíveis com o arquivo de *BackUp* disponível; resgatar todos os dados guardados no arquivo de *BackUp*, e encontrar um *driver* adequado ao seu funcionamento com o JDBC), resolvemos migrar toda a base para o banco de dados Microsoft Access, já que seu funcionamento e sua manutenção são bem mais simples.

## **5.3 Trabalhos Futuros**



O *chatterbot* implementado forneceu resultados bastante satisfatórios para a tarefa de aquisição de conhecimento de perfil de usuário. Entretanto, algumas extensões podem ser efetuadas neste sistema, de modo a torná-lo mais eficiente para a tarefa proposta, ou a fim de incluir novas funcionalidades. A não realização das melhorias aqui propostas deveu, principalmente, às limitações de tempo.

### **Extensão da Base de Falas**

Um dos fatores que mais contribuiu para o desempenho do robô não ter sido melhor foi o fato de muitos usuários perguntarem ao robô informações sobre suas preferências, não deixando que o robô guiasse o diálogo, de forma a realizar a aquisição de perfil.

Uma extensão da base de categorias, contendo falas capazes de responder perguntas dirigidas ao robô e, ao mesmo tempo, capazes de guiar o diálogo para a aquisição, seria de ganho considerável, melhorando a aquisição de perfil e o desempenho geral do *chatterbot*. Essa extensão seria guiada por uma análise detalhada dos arquivos de *log* de uso do sistema, que indicaria que tipos de perguntas os usuários comumente fazem ao robô.

### **Recomendação de Filmes**

Não foi possível implementar o módulo de recomendação de filmes no sistema, uma vez que não tivemos acesso ao módulo de processamento de recomendação implementado pelo sistema GEP. Esta é a mais importante extensão relacionada ao *chatterbot*, já que a aquisição de perfil dos usuários tem por objetivo guiar o processo de recomendação personalizada de filmes.

Esse módulo deverá funcionar como uma interface em linguagem natural entre usuário e sistema GEP. A partir da sentença do usuário, o *chatterbot* deverá ser capaz de consultar o módulo de recomendação de filmes do GEP, retornando ao usuário uma resposta contendo o filme indicado de acordo com o seu perfil. Com isso, toda a

comunicação entre usuário e sistema GEP será feita em linguagem natural, tornando o uso do sistema mais amigável.

### **Criação de Chatterbots para Novos Domínios**

A partir do trabalho desenvolvido aqui, seria possível criar novos *chatterbots* para aquisição de perfil em outros domínios. Para isto, seria necessária a criação de um novo banco de dados (ou utilização de algum banco já existente) sobre o domínio em questão. Assim, poderíamos ter robôs capazes de conversarem sobre os mais diversos temas nas áreas de entretenimento, ajuda ao usuário, comércio, etc.

Seria interessante, por exemplo, criar um *chatterbot* capaz de conversar sobre tipos de jogos de computador, realizando, da mesma forma como apresentado neste trabalho, a aquisição de perfil dos usuários sobre estilo de jogo, empresa produtora, qualidade gráfica, enredo, inteligência computacional dos jogos, etc. Essas informações seriam, então, utilizadas para a recomendação personalizada de jogos. Outra idéia seria criar um robô capaz de conversar sobre desenhos animados na TV, adquirindo o perfil de crianças, e futuramente realizando recomendação personalizada de desenhos.

## Referências Bibliográficas

---

[20q.net 2002] Online em [www.20q.net](http://www.20q.net)

[BOOSE 1986] Boose, J. H. *Expertise Transfer for Expert Systems Design*. New York: Elsevier, 1986.

[BOOSE e BRADSHAW 1987] Boose, J. H., and J.M. Brandshaw. "A Knowledge Acquisition Workbench for Eliciting Decision Knowledge." In *Proceedings, 20<sup>th</sup> HICSS*. Hawaii, January 1987.

[BOOSE 1989] Boose, J. H. "A Survey of Knowledge Acquisition Techniques and Tools." *Knowledge Acquisition* 1 (March 1989).

[BOOSE e GAINES 1990] Boose, J. H., and B. R. Gaines, eds. *The Foundations of Knowledge Acquisiton*. New York: Academic Press, 1990.

[BREUKER 1991] Breuker, J. A. *KADS: A Modelling Approach to Knowledge Engineering*. Paper submitted to the journal Knowledge Acquisition.

[BRIAN 2002] Online em <http://www.strout.net/info/science/ai/brian>

[CHAT 2002] Online em <http://debra.dgbt.doc.ca/chat/chat.html>

[COLBY 1975] K. Colby, *Artificial Paranoia: A Computer Simulation of Paranoid Process*, Pergamon Press., New York, 1975.

[CYBELLE 2002] Online em <http://cybelle.cjb.net/inf>

[DIEDERICH et al.1987] Diederich, J., et al. "Kriton: A Knowledge Acquisition Tool for Expert Systems." *International Journal of Man-Machine Studies* (January 1987).

[DR. ABUSE 2002] Online em <http://www.ctv.es/USERS/jboronat//drabuse.html>

[ELOISA 2002] Online em <http://www.eloisa.it/>

[EMYCIN 2002] Online em <http://www.wkonline.com/d/EMYCIN.html>

[ESTADÃO 2002] Online em <http://www.estadao.com.br/magazine/materias/2002/jan/11/87.htm>

[FONER 1994] Foner, L. N. What's an agent, anyway? A sociological case study. [online] Disponível na Internet via WWW. URL:

<http://foner.www.media.mit.edu/people/foner/Julia/Julia.html>. Arquivo capturado em 3 de abril de 1999.

[FREILING et al. 1985] Freiling, M., et al. "Starting a Knowledge Engineering Project: A Step by Step Approach." *AI Magazine* (Fall 1985).

[HART 1986] Hart, A. *Knowledge Acquisition for Expert Systems*. New York: McGraw-Hill, 1986.

[HAYES-ROTH et al. 1983] Hayes-Roth, F., et al. *Building Expert Systems*. Reading, Mass.: Addison-Wesley, 1983.

[HUTCHENS 1998] Hutchens, J. L.. How Hex Works, online em <http://www.amristar.com.au/~hutch/hex/How.html>

[JABBERWACKY 2002] Online em <http://www.jabberwacky.com/>

[JOHN LENNON 2002] Online em <http://www.triumphpc.com/john-lennon/index.shtml>

[KIM e COURTNEY 1988] Kim, J., e J. F. Courtney. "A Survey of Knowledge Acquisition Techniques and Their Relevance to Managerial Problem Domains." *Decision Support Systems* 4 (October 1988).

[KLONE 2002] Online em <http://www.bigscience.com>

[LAVEN 2001] Laven, S. online em <http://www.simonlaven.com>

[LOEBNER 2001] Loebner Prize Contest. online em <http://www.loebner.net>

[MAULDIN 1999] Mauldin, Michael L. Chatterbots, tinymuds and the Turing test: entering the Loebner Prize Competition. [online] Disponível na Internet via WWW. URL: <http://indian.monterey.edu/sbsc/sbsc368a/aaai94.htm>. Arquivo capturado em 12 de maio de 1999.

[McGRAW et. al. 1989] McGraw, K. L., and B. K. Harbison-Briggs. *Knowledge Acquisition, Principles and Guidelines*. Englewood Cliffs, N. J.: Prentice-Hall, 1989.

[MEGA-HAL 2002] Online em <http://www.futuriai.com/files/mhal.html>

[MORRIS 1964] Morris, C. - *Signs, Language, and Behavior*. New York, NY: Braziller, 1946

[NEVES et al. 2002] Neves, A. M. M., Diniz, I., Barros, F. A., Natural Language Communication via AIML Plus Chatterbots In: *V Symposium on Human Factors in Computer Systems (IHC 2002)*, 2002, Fortaleza - CE. Proc. of the IHC 2002 - V Symposium on Human Factors in Computer Systems 2002. , 2002. p.387 – 387

- [NICOLE 2002] Online em <http://www.neuromidia.com/index.html>
- [PANTANO 2000] Online em <http://www.cnet.com/techtrends/0-1544320-8-2862007-2.html>
- [SEARLE 1995] Searle John R. *Expressão e significado – estudos da teoria dos atos de fala* – Martins Fontes, 1995.
- [SHALLOWRED 2002] Online em <http://isis.neuromidia.com/html2/blank.htm>
- [TURBAN 1992] Turban, E. *Expert Systems and Applied Artificial Intelligence*. Macmillan Publishing Company, 1992.
- [TURING 1950] Turing, A. M. Computing machinery and intelligence, *Mind*, Vol. 59, No. 236, 433–460.
- [TUTHILL 1990] Tuthill, G. S. *Knowledge Engineering*. Blue Ridge Summit, Pa.: TAB Books, 1990.
- [WALLACE 2001] Wallace, R. S. Don't read me - A.L.I.C.E. and AIML documentation, online em <http://www.alicebot.com/dont.html>
- [WEINTRAUB 1986] Weintraub, J. "History of the PC Therapist" Online em <http://www.loebner.net/Prizef/weintraub-bio.html>
- [WEIZENBAUM 1966] Weizenbaum, J. Eliza - a computer program for the study of natural language communication between man and machine, *Communication of the Association for Computing Machinery* 9: 36–45.
- [WOLFGRAM 1987] Wolfram, D. D., et al. *Expert Systems*. New York: John Wiley & Sons, 1987.
- [ZABAWARE 2002] ZABAWARE – Ultra Hal Representative, Disponível na Internet via WWW. URL: <http://www.ultrahal.com>.

# Apêndice A

Devido a grande quantidade de arquivos de falas da base AIML, apenas alguns exemplos foram escolhidos para serem mostrados.

## 1. Arquivo: cumprimentar\_o\_robot.aiml

```
?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>oi</pattern>
<template>
<srai>oi</srai>
</template>
</category>

<category>
<pattern>oi *</pattern>
<template>
<srai>oi</srai>
</template>
</category>

<category>
<pattern>ola</pattern>
<template>
<srai>oi</srai>
</template>
</category>

<category>
<pattern>ola *</pattern>
<template>
<srai>oi</srai>
</template>
</category>

<category>
<pattern>oi</pattern>
<template>
<random>
<li>oi, <b><get name="name"/></b>. </li>
<li>oi <b><get name="name"/></b>, é muito bom encontrar você. gosto muito de filmes de ação e você ?
gosta de que tipo de filmes ?</li>
<li>oi, espero que você converse bastante comigo sobre filmes. que tipo de filmes você mais curte ?</li>
<li>oi, que bom que você veio, <b><get name="name"/></b>. que tipo de filme você gosta ?</li>
<li>oi <b><get name="name"/></b>, vamos conversar sobre filmes ! vou começar perguntando que tipo de
filmes você mais curte ?</li>
```

```

<li>olá, <b><get name="name"/></b>. qual gênero você mais gosta ?</li>
<li>olá, que bom que você está aqui, <b><get name="name"/></b>. que tipos de filmes você mais curte
?</li>
</random>
</template>
</category>

</aiml>

```

## 2. Arquivo: elogiar\_o\_robot.aiml

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>encantada</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>estou encantada</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>estou encantada com voce</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>estou encantado</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>estou encantado com voce</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>fico encantada</pattern>
<template>
<srai>encantado</srai>
</template>

```

```
</category>

<category>
<pattern>fico encantada com voce</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>fico encantado</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>fico encantado com voce</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>fiquei encantada</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>fiquei encantada com voce</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>fiquei encantado</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>fiquei encantado com voce</pattern>
<template>
<srai>encantado</srai>
</template>
</category>

<category>
<pattern>prazer</pattern>
<template>
<srai>encantado</srai>
</template>
```



```

</category>

<category>
<pattern>encantado</pattern>
<template>
<random>
<li>foi gentil de sua parte dizer isso, <b><get name="name"/></b>.</li>
<li>obrigado.</li>
<li>ohhh... assim você me deixa sem jeito...</li>
<li>o prazer é todo meu, <b><get name="name"/></b>.</li>
<li>puxa, fiquei sensibilizado com estas palavras.</li>
<li>também estou encantado.</li>
<li>também estou encantado com você.</li>
<li>você é muito gentil, <b><get name="name"/></b>.</li>
<li>você é tão educado.</li>
</random>
</template>
</category>

</aiml>

```

### 3. Arquivo: filme\_policial.aiml

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>adoro filmes policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>adoro filmes policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>amo filmes policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>amo filmes policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>

```

```
<category>
<pattern>eu gosto de policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>eu gosto de policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>filme policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>filme policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>filmes policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>filmes policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>gosto de filme policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>gosto de filme policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>
```

```
<category>
<pattern>gosto de filmes policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>
```

```
<category>
<pattern>gosto de filmes policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>
```

```
<category>
<pattern>gosto de policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>
```

```
<category>
<pattern>gosto de policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>
```

```
<category>
<pattern>policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>
```

```
<category>
<pattern>policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>
```

```
<category>
<pattern>prefiro filme policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>
```

```
<category>
<pattern>prefiro filme policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>
```

```

<category>
<pattern>prefiro filmes policiais</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>prefiro filmes policial</pattern>
<template>
<srai>policial</srai>
</template>
</category>

<category>
<pattern>policial</pattern>
<template>
<random>
<li><b><get name="name"/></b>, qual a nota de 1 a 5 que você daria a:
<buscafilme>Policial</buscafilme> ?</li>
<li>Eu gostei muito de: <buscafilme>Policial</buscafilme> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>filmes policiais são bem interessantes ! você gostou de: <buscafilme>Policial</buscafilme> ? Que nota
de 1 a 5 você daria a ele ?</li>
<li>gosto de alguns. você gostou de: <buscafilme>Policial</buscafilme> ? Que nota de 1 a 5 você daria a
ele ?</li>
<li>gosto muito, mas prefiro filmes de ação ! Eu gostei de: <buscafilme>Policial</buscafilme> ! Que nota
de 1 a 5 você daria a ele ?</li>
</random>
</template>
</category>

</aiml>

```

#### 4. Arquivo: filme\_de\_aventura.aiml

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>adoro filmes de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
<pattern>amo filmes de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>

```

```
<pattern>aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
<pattern>eu gosto de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
<pattern>filme de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
<pattern>filmes de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
<pattern>gosto de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
<pattern>gosto de filme de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
<pattern>gosto de filmes de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
<pattern>prefiro filme de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
```

```

<pattern>prefiro filmes de aventura</pattern>
<template>
<srai>aventura</srai>
</template>
</category>

<category>
<pattern>aventura</pattern>
<template>
<random>
<li><b><get name="name"/></b>, qual a nota de 1 a 5 que voce daria a:
<buscafilme>Aventura</buscafilme> ?</li>
<li>Eu gostei muito de: <buscafilme>Aventura</buscafilme> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>filmes de aventura às vezes são muito engraçados ! Eu gostei muito de:
<buscafilme>Aventura</buscafilme> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>filmes de aventura são muito legais ! você gostou de: <buscafilme>Aventura</buscafilme> ? Que nota
de 1 a 5 você daria a ele ?</li>
<li>gosto muito, mas prefiro filmes de ação ! você gostou de: <buscafilme>Aventura</buscafilme> ? Que
nota de 1 a 5 você daria a ele ?</li>
<li>você gostou de: <buscafilme>Aventura</buscafilme> ? Que nota você daria a ele, de 1 a 5 ?</li>
</random>
</template>
</category>

</aiml>

```

## 5. Arquivo: filme\_de\_guerra.aiml

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>adoro filmes de guerra</pattern>
<template>
<srai>guerra</srai>
</template>
</category>

<category>
<pattern>adoro filmes de guerras</pattern>
<template>
<srai>guerra</srai>
</template>
</category>

<category>
<pattern>amo filmes de guerra</pattern>
<template>
<srai>guerra</srai>
</template>
</category>

<category>

```

<pattern>amo filmes de guerras</pattern>

<template>

<srai>guerra</srai>

</template>

</category>

<category>

<pattern>eu gosto de guerra</pattern>

<template>

<srai>guerra</srai>

</template>

</category>

<category>

<pattern>eu gosto de guerras</pattern>

<template>

<srai>guerra</srai>

</template>

</category>

<category>

<pattern>filme de guerra</pattern>

<template>

<srai>guerra</srai>

</template>

</category>

<category>

<pattern>filme de guerras</pattern>

<template>

<srai>guerra</srai>

</template>

</category>

<category>

<pattern>filmes de guerra</pattern>

<template>

<srai>guerra</srai>

</template>

</category>

<category>

<pattern>filmes de guerras</pattern>

<template>

<srai>guerra</srai>

</template>

</category>

<category>

<pattern>gosto de filme de guerra</pattern>

<template>

<srai>guerra</srai>

</template>

</category>

<category>

```
<pattern>gosto de filme de guerras</pattern>
<template>
<srai>guerra</srai>
</template>
</category>
```

```
<category>
<pattern>gosto de filmes de guerra</pattern>
<template>
<srai>guerra</srai>
</template>
</category>
```

```
<category>
<pattern>gosto de filmes de guerras</pattern>
<template>
<srai>guerra</srai>
</template>
</category>
```

```
<category>
<pattern>gosto de guerra</pattern>
<template>
<srai>guerra</srai>
</template>
</category>
```

```
<category>
<pattern>gosto de guerras</pattern>
<template>
<srai>guerra</srai>
</template>
</category>
```

```
<category>
<pattern>guerra</pattern>
<template>
<srai>guerra</srai>
</template>
</category>
```

```
<category>
<pattern>guerras</pattern>
<template>
<srai>guerra</srai>
</template>
</category>
```

```
<category>
<pattern>prefiro filme de guerra</pattern>
<template>
<srai>guerra</srai>
</template>
</category>
```

```
<category>
```



```

<pattern>prefiro filme de guerras</pattern>
<template>
<srai>guerra</srai>
</template>
</category>

<category>
<pattern>prefiro filmes de guerra</pattern>
<template>
<srai>guerra</srai>
</template>
</category>

<category>
<pattern>prefiro filmes de guerras</pattern>
<template>
<srai>guerra</srai>
</template>
</category>

<category>
<pattern>guerra</pattern>
<template>
<random>
<li><b><get name="name"/></b>, qual a nota de 1 a 5 que você daria a:
<buscafilme>Guerra</buscafilme> ?</li>
<li>Eu gostei muito de: <buscafilme>Guerra</buscafilme> ! Que nota você daria a ele ? De 1 a 5 ?</li>
<li>filmes de guerra às vezes são muito violentos! você gostou de: <buscafilme>Guerra</buscafilme> ?
Que nota você daria a ele ? De 1 a 5 ?</li>
<li>filmes de guerra são muito violentos ! você assistiu: <buscafilme>Guerra</buscafilme> ? Que nota
você daria a ele ? De 1 a 5 ?</li>
<li>gosto um pouco, mas prefiro filmes de ação ! Eu gostei de: <buscafilme>Guerra</buscafilme> ! Que
nota você daria a ele ? De 1 a 5 ?</li>
<li>você gostou de: <buscafilme>Guerra</buscafilme> ? Que nota você daria a ele ? De 1 a 5 ?</li>
</random>
</template>
</category>

</aiml>

```

## 6. Arquivo: sinopse\_de\_filme.aiml

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>como e esse filme</pattern>
<template>
<srai>sinopse</srai>
</template>
</category>

<category>
<pattern>como ele e</pattern>

```

```

<template>
<srai>sinopse</srai>
</template>
</category>

<category>
<pattern>fala sobre o que</pattern>
<template>
<srai>sinopse</srai>
</template>
</category>

<category>
<pattern>fale sobre ele</pattern>
<template>
<srai>sinopse</srai>
</template>
</category>

<category>
<pattern>qual a descricao</pattern>
<template>
<srai>sinopse</srai>
</template>
</category>

<category>
<pattern>qual a descricao do filme</pattern>
<template>
<srai>sinopse</srai>
</template>
</category>

<category>
<pattern>qual a sinopse</pattern>
<template>
<srai>sinopse</srai>
</template>
</category>

<category>
<pattern>que filme e esse</pattern>
<template>
<srai>sinopse</srai>
</template>
</category>

<category>
<pattern>sinopse</pattern>
<template>
<random>
<li>A descrição dele é: <sinopse>sinopse</sinopse> </li>
<li>Ele é mais ou menos assim: <sinopse>sinopse</sinopse> </li>
<li>Fala sobre: <sinopse>sinopse</sinopse> </li>
</random>
</template>

```

```
</category>
```

```
</aiml>
```

## 7. Arquivo: diretor\_de\_filme.aiml

```
<?xml version="1.0" encoding="windows-1252"?>
```

```
<aiml version="1.0">
```

```
<meta content="pt" name="language"/>
```

```
<category>
```

```
<pattern>qual e o diretor</pattern>
```

```
<template>
```

```
<srai>diretor</srai>
```

```
</template>
```

```
</category>
```

```
<category>
```

```
<pattern>qual e o diretor desse filme</pattern>
```

```
<template>
```

```
<srai>diretor</srai>
```

```
</template>
```

```
</category>
```

```
<category>
```

```
<pattern>qual e o diretor do filme</pattern>
```

```
<template>
```

```
<srai>diretor</srai>
```

```
</template>
```

```
</category>
```

```
<category>
```

```
<pattern>qual e o nome do diretor</pattern>
```

```
<template>
```

```
<srai>diretor</srai>
```

```
</template>
```

```
</category>
```

```
<category>
```

```
<pattern>quem dirigiu</pattern>
```

```
<template>
```

```
<srai>diretor</srai>
```

```
</template>
```

```
</category>
```

```
<category>
```

```
<pattern>quem dirigiu ele</pattern>
```

```
<template>
```

```
<srai>diretor</srai>
```

```
</template>
```

```
</category>
```

```
<category>
```

```
<pattern>quem dirigiu esse filme</pattern>
```

```
<template>
```

```
<srai>diretor</srai>
</template>
</category>

<category>
<pattern>quem e o diretor</pattern>
<template>
<srai>diretor</srai>
</template>
</category>

<category>
<pattern>quem e o diretor desse filme</pattern>
<template>
<srai>diretor</srai>
</template>
</category>

<category>
<pattern>quem escreveu esse filme</pattern>
<template>
<srai>diretor</srai>
</template>
</category>

<category>
<pattern>quem foi o diretor dele</pattern>
<template>
<srai>diretor</srai>
</template>
</category>

<category>
<pattern>quem foi o diretor desse filme</pattern>
<template>
<srai>diretor</srai>
</template>
</category>

<category>
<pattern>diretor</pattern>
<template>
<random>
<li>Ele é o: <diretor>diretor</diretor> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>É o: <diretor>diretor</diretor> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>O diretor é: <diretor>diretor</diretor> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>O nome dele é: <diretor>diretor</diretor> ! Que nota de 1 a 5 você daria a ele ?</li>
</random>
</template>
</category>

</aiml>
```

**8. Arquivo: ator\_de\_filme.aiml**

```
<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>qual e o ator</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>qual e o ator desse filme</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>qual e o ator do filme</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>qual e o ator principal</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>qual e o ator principal desse filme</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>qual e o ator principal do filme</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>qual e o nome do ator</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>qual e o nome do ator principal</pattern>
```

```
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem atuou ele</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem atuou ele</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem e o ator</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem e o ator desse filme</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem e o ator principal</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem e o ator principal desse filme</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem foi o ator dele</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem foi o ator desse filme</pattern>
```

```
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem foi o ator principal dele</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem foi o ator principal desse filme</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem trabalhou</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem trabalhou</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem trabalhou nesse filme</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>quem trabalhou nesse filme</pattern>
<template>
<srai>ator</srai>
</template>
</category>

<category>
<pattern>ator</pattern>
<template>
<random>
<li>Ele é o: <ator>ator</ator> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>É o: <ator>ator</ator> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>O ator é: <ator>ator</ator> ! Que nota de 1 a 5 você daria a ele ?</li>
<li>O nome dele é: <ator>ator</ator> ! Que nota de 1 a 5 você daria a ele ?</li>
</random>
```

```
</template>
</category>

</aiml>
```

## 9. Arquivo: nota\_dada\_1.aiml

```
<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>1</pattern>
<template>
<srai>1</srai>
</template>
</category>

<category>
<pattern>acho que 1</pattern>
<template>
<srai>1</srai>
</template>
</category>

<category>
<pattern>acho que nota 1</pattern>
<template>
<srai>1</srai>
</template>
</category>

<category>
<pattern>doria 1</pattern>
<template>
<srai>1</srai>
</template>
</category>

<category>
<pattern>doria nota 1</pattern>
<template>
<srai>1</srai>
</template>
</category>

<category>
<pattern>dou 1</pattern>
<template>
<srai>1</srai>
</template>
</category>

<category>
<pattern>dou nota 1</pattern>
```



```

<template>
<srai>1</srai>
</template>
</category>

<category>
<pattern>nota 1</pattern>
<template>
<srai>1</srai>
</template>
</category>

<category>
<pattern>1</pattern>
<template>
<random>
<li>Acho que não merecia tão pouco, mas você é quem manda! <notadada>1</notadada></li>
<li>A pior nota! Mas é sua opinião que vale aqui! <notadada>1</notadada></li>
<li>Hiiii, <b><get name="name"/></b>, tô vendo que você não gosta mesmo!
<notadada>1</notadada></li>
<li>Ok! <notadada>1</notadada></li>
</random>
</template>
</category>

</aiml>

```

**10. Arquivo: nota\_dada\_2.aiml**

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>2</pattern>
<template>
<srai>2</srai>
</template>
</category>

<category>
<pattern>acho que 2</pattern>
<template>
<srai>2</srai>
</template>
</category>

<category>
<pattern>acho que nota 2</pattern>
<template>
<srai>2</srai>
</template>
</category>

<category>

```

```
<pattern>daria 2</pattern>
<template>
<srai>2</srai>
</template>
</category>

<category>
<pattern>daria nota 2</pattern>
<template>
<srai>2</srai>
</template>
</category>

<category>
<pattern>dou 2</pattern>
<template>
<srai>2</srai>
</template>
</category>

<category>
<pattern>dou nota 2</pattern>
<template>
<srai>2</srai>
</template>
</category>

<category>
<pattern>nota 2</pattern>
<template>
<srai>2</srai>
</template>
</category>

<category>
<pattern>2</pattern>
<template>
<random>
<li>É sua opinião, mas tudo bem! <notadada>2</notadada> </li>
<li>Nota regular, mas tudo bem! <notadada>2</notadada></li>
<li>Ok ! <notadada>2</notadada></li>
<li>Valeu! <notadada>2</notadada></li>
</random>
</template>
</category>

</aiml>
```

**11. Arquivo: nota\_dada\_3.aiml**

```
<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
```

```
<pattern>3</pattern>
<template>
<srain>3</srain>
</template>
</category>

<category>
<pattern>acho que 3</pattern>
<template>
<srain>3</srain>
</template>
</category>

<category>
<pattern>acho que nota 3</pattern>
<template>
<srain>3</srain>
</template>
</category>

<category>
<pattern>daria 3</pattern>
<template>
<srain>3</srain>
</template>
</category>

<category>
<pattern>daria nota 3</pattern>
<template>
<srain>3</srain>
</template>
</category>

<category>
<pattern>dou 3</pattern>
<template>
<srain>3</srain>
</template>
</category>

<category>
<pattern>dou nota 3</pattern>
<template>
<srain>3</srain>
</template>
</category>

<category>
<pattern>nota 3</pattern>
<template>
<srain>3</srain>
</template>
</category>

<category>
```

```
<pattern>3</pattern>
<template>
<random>
<li>Boa nota! <notadada>3</notadada></li>
<li>Obrigado por ter dado uma nota, <b><get name="name"/></b>! <notadada>3</notadada></li>
<li>Ok ! <notadada>3</notadada></li>
<li>Valeu! <notadada>3</notadada></li>
</random>
</template>
</category>

</aiml>
```

## 12. Arquivo: nota\_dada\_4.aiml

```
<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>4</pattern>
<template>
<srai>4</srai>
</template>
</category>

<category>
<pattern>acho que 4</pattern>
<template>
<srai>4</srai>
</template>
</category>

<category>
<pattern>acho que nota 4</pattern>
<template>
<srai>4</srai>
</template>
</category>

<category>
<pattern>daria 4</pattern>
<template>
<srai>4</srai>
</template>
</category>

<category>
<pattern>daria nota 4</pattern>
<template>
<srai>4</srai>
</template>
</category>

<category>
```

```

<pattern>dou 4</pattern>
<template>
<srai>4</srai>
</template>
</category>

<category>
<pattern>dou nota 4</pattern>
<template>
<srai>4</srai>
</template>
</category>

<category>
<pattern>nota 4</pattern>
<template>
<srai>4</srai>
</template>
</category>

<category>
<pattern>4</pattern>
<template>
<random>
<li>Gostei da sua nota, <b><get name="name"/></b> ! <notadada>4</notadada> </li>
<li>Muito boa nota! <notadada>4</notadada></li>
<li>Ok ! <notadada>4</notadada></li>
<li>Que bom, <b><get name="name"/></b>! <notadada>4</notadada> </li>
</random>
</template>
</category>

</aiml>

```

**13. Arquivo: nota\_dada\_5.aiml**

```

<?xml version="1.0" encoding="windows-1252"?>
<aiml version="1.0">
<meta content="pt" name="language"/>

<category>
<pattern>5</pattern>
<template>
<srai>5</srai>
</template>
</category>

<category>
<pattern>acho que 5</pattern>
<template>
<srai>5</srai>
</template>
</category>

```

```
<category>
<pattern>acho que nota 5</pattern>
<template>
<srai>5</srai>
</template>
</category>

<category>
<pattern>doria 5</pattern>
<template>
<srai>5</srai>
</template>
</category>

<category>
<pattern>doria nota 5</pattern>
<template>
<srai>5</srai>
</template>
</category>

<category>
<pattern>dou 5</pattern>
<template>
<srai>5</srai>
</template>
</category>

<category>
<pattern>dou nota 5</pattern>
<template>
<srai>5</srai>
</template>
</category>

<category>
<pattern>nota 5</pattern>
<template>
<srai>5</srai>
</template>
</category>

<category>
<pattern>5</pattern>
<template>
<random>
<li>Ok ! <notadada>5</notadada></li>
<li>Ótima nota ! <notadada>5</notadada></li>
<li>Que ótimo, <b><get name="name"/></b> ! <notadada>5</notadada></li>
<li>Você realmente gosta muito ! <notadada>5</notadada></li>
</random>
</template>
</category>

</aiml>
```

## Apêndice B

Aqui apresentamos o código fonte das tags implementadas para o sistema.

### Tag BuscaFilme:

```
package org.alicebot.server.core.processor;

import org.alicebot.server.core.PredicateMaster;
import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;
import org.alicebot.server.core.util.Toolkit;

import java.util.StringTokenizer;

import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;

public class BuscaFilmeProcessor extends AIMLProcessor
{
    String url = "jdbc:odbc:Gep";
    String username = "tjmm";
    String password = "123456";
    String query1 = "SELECT CD_PROGRAM FROM PROGRAM_GENRE WHERE
CD_GENRE = ";
    String query2 = "SELECT * FROM PROGRAM WHERE CD_PROGRAM = ";

    String palavra1;
    String palavra2;

    public static final String label = "buscafilme";
    private static final String SPACE = " ";
    private static final String serverBusca = "";

    public String process(int level, String userid, XMLNode tag,
AIMLParser parser) throws AIMLProcessorException
    {
        if (tag.XMLType == XMLNode.TAG)
        {
            String response = parser.evaluate(level++, userid,
tag.XMLChild);

            if (response.equals(EMPTY_STRING))
            {
                return response;
            }

            StringTokenizer tokenizer = new StringTokenizer(response,
SPACE);

            StringBuffer result = new StringBuffer(response.length());
            String word = tokenizer.nextToken();

            if(word.equals("Ação"))
```

```

    {
        query1 = query1 + "716";
    }
    else if (word.equals("Documentário"))
    {
        query1 = query1 + "2";
    }
    else if (word.equals("Suspense"))
    {
        query1 = query1 + "6";
    }
    else if (word.equals("Drama"))
    {
        query1 = query1 + "14";
    }
    else if (word.equals("Aventura"))
    {
        query1 = query1 + "29";
    }
    else if (word.equals("Comédia"))
    {
        query1 = query1 + "49";
    }
    else if (word.equals("Guerra"))
    {
        query1 = query1 + "65";
    }
    else if (word.equals("Musical"))
    {
        query1 = query1 + "111";
    }
    else if (word.equals("Terror"))
    {
        query1 = query1 + "131";
    }
    else if (word.equals("Infantil"))
    {
        query1 = query1 + "193";
    }
    else if (word.equals("Erótico"))
    {
        query1 = query1 + "231";
    }
    else if (word.equals("Policial"))
    {
        query1 = query1 + "239";
    }
    else if (word.equals("Romance"))
    {
        query1 = query1 + "466";
    }
    else if (word.equals("Western"))
    {
        query1 = query1 + "1037";
    }
    else if (word.equals("Épico"))
    {

```



```

        query1 = query1 + "1045";
    }
    else if (word.equals("Fantasia"))
    {
        query1 = query1 + "4334";
    }
    else if (word.equals("Ficção Científica"))
    {
        query1 = query1 + "5341";
    }
    else if (word.equals("Faroeste"))
    {
        query1 = query1 + "5576";
    }
    else if (word.equals("Religioso"))
    {
        query1 = query1 + "13342";
    }
    else if (word.equals("Animação"))
    {
        query1 = query1 + "122972";
    }
    else if (word.equals("Desenho"))
    {
        query1 = query1 + "123101";
    }
    else if (word.equals("Show"))
    {
        query1 = query1 + "123516";
    }
    else if (word.equals("Ficção"))
    {
        query1 = query1 + "125476";
    }
    }

    palavra1 =
AcessoJDBC.ConsultarRandomico(query1,url,username,password,1);

    PredicateMaster.set("filme", palavra1, userid);

    query2 = query2 + palavra1;

    PredicateMaster.set("dado", "1", userid);

    palavra2 =
AcessoJDBC.Consultar(query2,url,username,password,2);

    return palavra2;

    }
    else
    {
        throw new AIMLProcessorException("<buscafilme></buscafilme>
must have content!");
    }
}
}

```

## Tag Sinopse:

```
package org.alicebot.server.core.processor;

import org.alicebot.server.core.PredicateMaster;
import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;
import org.alicebot.server.core.util.Toolkit;

import java.util.StringTokenizer;

import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;

public class SinopseProcessor extends AIMLProcessor
{
    String url = "jdbc:odbc:Gep";
    String username = "tjmm";
    String password = "123456";
    String query = "SELECT DS_DESC FROM PROGRAM WHERE CD_PROGRAM = ";

    String descricao;

    public static final String label = "sinopse";

    //Convenience constants.
    private static final String SPACE = " ";

    public String process(int level, String userid, XMLNode tag,
AIMLParser parser) throws AIMLProcessorException
    {
        if (tag.XMLType == XMLNode.TAG)
        {
            query = query + PredicateMaster.get("filme", userid);

            descricao =
AcessoJDBC.Consultar(query,url,username,password,1);

            return descricao;

        }
        else
        {
            throw new AIMLProcessorException("<sinopse></sinopse> must
have content!");
        }
    }
}
```

## Tag Diretor:

```

package org.alicebot.server.core.processor;

import org.alicebot.server.core.PredicateMaster;
import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;
import org.alicebot.server.core.util.Toolkit;

import java.util.StringTokenizer;

import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;

public class DiretorProcessor extends AIMLProcessor
{
    String url = "jdbc:odbc:Gep";
    String username = "tjmm";
    String password = "123456";
    String query = "SELECT CD_ARTIST FROM PROGRAM_DIRECT WHERE
CD_PROGRAM = ";
    String query2 = "SELECT NM_PRSN FROM ARTIST WHERE CD_ARTIST = ";
    String result1, result2;
    String diretor;

    public static final String label = "diretor";

    // Convenience constants.
    private static final String SPACE = " ";

    public String process(int level, String userid, XMLNode tag,
AIMLParser parser) throws AIMLProcessorException
    {
        if (tag.XMLType == XMLNode.TAG)
        {
            query = query + PredicateMaster.get("filme", userid);

            result1 =
AcessoJDBC.Consultar(query,url,username,password,1);

            query2 = query2 + result1;

            PredicateMaster.set("diretor", result1, userid);

            PredicateMaster.set("dado", "2", userid);

            result2 =
AcessoJDBC.Consultar(query2,url,username,password,1);

            return result2;

        }

        else
        {
            throw new AIMLProcessorException("<diretor></diretor> must
have content!");
        }
    }
}

```

```

    }
}

```

### Tag Ator:

```

package org.alicebot.server.core.processor;

import org.alicebot.server.core.PredicateMaster;
import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;
import org.alicebot.server.core.util.Toolkit;

import java.util.StringTokenizer;

import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;

public class AtorProcessor extends AIMLProcessor
{
    String url = "jdbc:odbc:Gep";
    String username = "tjmm";
    String password = "123456";
    String query1 = "SELECT CD_ARTIST FROM PROGRAM_CAST WHERE
CD_PROGRAM = ";
    String query2 = "SELECT NM_PRSN FROM ARTIST WHERE CD_ARTIST = ";
    String result1, result2;

    public static final String label = "ator";

    // Convenience constants.
    private static final String SPACE = " ";

    public String process(int level, String userid, XMLNode tag,
AIMLParser parser) throws AIMLProcessorException
    {
        if (tag.XMLType == XMLNode.TAG)
        {
            query1 = query1 + PredicateMaster.get("filme", userid);

            result1 =
AcessoJDBC.ConsultarRandomico(query1,url,username,password,1);

            query2 = query2 + result1;

            PredicateMaster.set("ator", result1, userid);

            PredicateMaster.set("dado", "3", userid);

            result2 =
AcessoJDBC.Consultar(query2,url,username,password,1);

            return result2;
        }
    }
}

```

```

        else
        {
            throw new AIMLProcessorException("<ator></ator> must have
content!");
        }
    }
}

```

### Tag NotaDada:

```

package org.alicebot.server.core.processor;

import org.alicebot.server.core.PredicateMaster;
import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;
import org.alicebot.server.core.util.Toolkit;

import java.util.StringTokenizer;
import java.lang.Integer;
import java.lang.Object;

import org.alicebot.server.core.parser.AIMLParser;
import org.alicebot.server.core.parser.XMLNode;

public class NotaDadaProcessor extends AIMLProcessor
{
    String url = "jdbc:odbc:Gep";
    String username = "tjmm";
    String password = "123456";
    String genericQuery = "INSERT INTO EVALUATION_TEMP (NM_USER,
CD_PROGRAM, CD_ARTIST, ARTIST, DIRECTOR, NOTA) VALUES ('";

    int word;

    public static final String label = "notadada";

    private static final String SPACE = " ";

    public String process(int level, String userid, XMLNode tag,
AIMLParser parser) throws AIMLProcessorException
    {
        if (tag.XMLType == XMLNode.TAG)
        {
            String response = parser.evaluate(level++, userid,
tag.XMLChild);

            if (response.equals(EMPTY_STRING))
            {
                return response;
            }

            StringTokenizer tokenizer = new
StringTokenizer(response, SPACE); //

```

```

        StringBuffer result = new
StringBuffer(response.length());          // Pega a nota da fala, nos
arquivos .AIML
        word = Integer.parseInt(tokenizer.nextToken());
//
        switch (Integer.parseInt(PredicateMaster.get("dado",
userid)))
        {
            case 1:
                genericQuery = genericQuery +
PredicateMaster.get("name", userid) + "', '" +
PredicateMaster.get("filme", userid) + "', '" + 0 + "', '" + 0 + "', '" +
0 + "', '" + word + "'");
AcessoJDBC.InsereRegistro(genericQuery,url,username,password);
                break;
            case 2:
                genericQuery = genericQuery +
PredicateMaster.get("name", userid) + "', '" +
PredicateMaster.get("filme", userid) + "', '" +
PredicateMaster.get("diretor", userid) + "', '" + 0 + "', '" + 1 + "', '"
+ word + "'");
AcessoJDBC.InsereRegistro(genericQuery,url,username,password);
                break;
            case 3:
                genericQuery = genericQuery +
PredicateMaster.get("name", userid) + "', '" +
PredicateMaster.get("filme", userid) + "', '" +
PredicateMaster.get("ator", userid) + "', '" + 1 + "', '" + 0 + "', '" +
word + "'");
AcessoJDBC.InsereRegistro(genericQuery,url,username,password);
                break;
            default:
                break;
        }
        return "";
    }
    else
    {
        throw new
AIMLProcessorException("<notadada></notadada> must have content!");
    }
}
}

```

**Classe Java responsável por toda a comunicação com o Banco de dados (arquivo AcessoJDBC.java):**

```
package org.alicebot.server.core.processor;
```

```

import java.sql.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class AcessoJDBC
{
    private static Statement statement;
    private static ResultSet resultset;
    private static Connection connection;

    private static String Valor;

    private static int Fetch = 0;

    //*****
    //
    // Método que cria uma Conexão com o Banco de Dados
    //
    //*****
    private static void Conectar(String url, String username, String
password)
    {
        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            connection = DriverManager.getConnection(url,username,password);

        } // Fim do try

        catch (ClassNotFoundException cnfex)
        {
            System.err.println("Falhou ao carregar o driver JDBC/ODBC");
            cnfex.printStackTrace();
            System.exit(1);

        }

        catch (SQLException sqllex)
        {
            System.err.println("Nao pode conectar!");
            sqllex.printStackTrace();

        }

    }

    //*****
    //
    // Método que Insere um novo registro no Banco de Dados
    //
    //*****
    public static void InsereRegistro(String consulta, String endereco,
String login, String senha)
    {
        try
        {

```

```

        Conectar(endereco, login, senha);

        connection.nativeSQL(consulta);
        statement = connection.createStatement();
        statement.executeUpdate(consulta);
        Desconectar();
    } // Fim do try

    catch (SQLException sqlex)
    {
        System.err.println("Nao pode conectar!");
        sqlex.printStackTrace();

    }
}

//*****
//
// Método que realiza uma consulta no Banco de Dados
//
//*****
public static String Consultar(String consulta, String endereco,
String login, String senha, int coluna)
{
    try
    {
        Conectar(endereco, login, senha);

        statement = connection.createStatement();
        resultset = statement.executeQuery(consulta);

        if (resultset.next())
        {
            Valor = resultset.getString(coluna);
            Desconectar();
        }

    } // Fim do try

    catch (SQLException sqlex)
    {
        System.err.println("Nao pode conectar!");
        sqlex.printStackTrace();

    }
    finally
    {
        return Valor;
    }

}

//*****
//
// Método que realiza uma consulta Rândomica no Banco de Dados
//
//*****

```



```

public static String ConsultarRandomico(String consulta, String
endereco, String login, String senha, int coluna)
{
    try
    {
        Random random = new Random();
        Vector vetor = new Vector();
        int Randomico;

        Conectar(endereco, login, senha);

        statement = connection.createStatement();
        resultset = statement.executeQuery(consulta);

        while (resultset.next())
        {
            vetor.add(resultset.getString(1));
        }

        Fetch = vetor.size(); // Pega o tamanho
do Vetor
        Randomico = random.nextInt(Fetch); // Pega um Valor
Randômico para ser usado na busca de um filme ou ator qualquer!!!
        Valor = vetor.get(Randomico).toString();
        Desconectar();

    } // Fim do try

    catch (SQLException sqllex)
    {
        System.err.println("Nao pode conectar!");
        sqllex.printStackTrace();

    }
    finally
    {
        return Valor;
    }

}
//*****
//
// Método que fecha as Conexões com o Banco de Dados
//
//*****
private static void Desconectar() throws SQLException
{
    statement.close();
    connection.close();
}
} // Fim da Classe AcessoJDBC

```