UNIVERSIDADE FEDERAL DE GOIÁS – UFG

CAMPUS CATALÃO – CaC

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO – DCC

Bacharelado em Ciência da Computação

Projeto Final de Curso

O Uso do *Processamento de Linguagem Natural* na Construção de *Chatterbots*

Autor: Eustáquio Cézar Pereira Filho

Orientador: Márcio Souza Dias

Catalão - 2009

Eustáquio Cézar Pereira Filho

O Uso do *Processamento de Linguagem Natural* na Construção de *Chatterbots*

Monografia apresentada ao Curso de
Bacharelado em Ciência da Computação da
Universidade Federal de Goiás, Campus Catalão,
como requisito parcial para obtenção do título de
Bacharel em Ciência da Computação

Área de Concentração: *Inteligência Artificial*
Orientador: Márcio Souza Dias

Catalão - 2009

C.Pereira , Eustáquio

O Uso do *Processamento de Linguagem Natural* na Construção de *Chatterbots*
Márcio Souza Dias- Catalão - 2009

Número de páginas: 46

Projeto Final de Curso (Bacharelado) Universidade Federal de Goiás, Campus Catalão, Curso de Bacharelado em Ciência da Computação, 2009.

Palavras-Chave: 1. *Chatterbot*. 2. *Processamento de Linguagem Natural*. 3. *Inteligência Artificial*

Eustáquio Cézar Pereira Filho

O Uso do *Processamento de Linguagem Natural* na Construção de *Chatterbots*

Monografia apresentada e aprovada em _____ de _____
Pela Banca Examinadora constituída pelos professores.

_____

Márcio Souza Dias – Presidente da Banca

_____

Luanna Lopes Lobato

_____

Vaston Gonçalves da Costa

Dedico este trabalho a Deus, em primeiro lugar;
aos meus pais Eustáquio e Márcia; aos companheiros
de trabalho que acreditaram em mim e me apoiaram;
a Mônica que em nenhum momento me deixou abater;
e a todos que estão ao meu lado nesta jornada.

AGRADECIMENTOS

Meus sinceros agradecimentos:

- ao meu orientador professor mestre Márcio Souza Dias, pela orientação, incentivo e companheirismo;
- aos companheiros de trabalho;
- e a todos os professores e colegas do departamento de Ciência da Computação.

“Nada é impossível para aquele que persiste.”
(Alexandre o Grande)

RESUMO

Filho, E.

O Uso do *Processamento de Linguagem Natural* na Construção de *Chatterbots*.
Curso de Ciência da Computação, Campus Catalão, UFG, Catalão, Brasil, 2009, 46p.

As pesquisas em *Processamento de Linguagem Natural*, uma subárea da *Inteligência Artificial,* têm se dedicado à análise e compreensão de idiomas. Elas consistem em desenvolver modelos computacionais para realização de tarefas que dependem de informações expressas em alguma língua natural. Deste modo, surgiram os *Chatterbots*, que são programas de computadores com o objetivo de simular uma conversação, dando a impressão que a conversa é entre humanos, e não entre uma pessoa e uma máquina.

Assim, o intuito deste trabalho é construir um *Chatterbot*, que designei de *Equus*, situado no contexto da eqüinocultura, com o objetivo de disseminar o conhecimento da cultura, da criação e manipulação de eqüinos. Este projeto mostra e estuda técnicas e ferramentas que serão usadas como subsídios para a construção do protótipo *Equus* e principalmente da ferramente de geração de conhecimento automático sobre a equinocultura.

Palavras-Chaves: *Chatterbot*, *Processamento de Linguagem Natural*, *Inteligência Artificial*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A evolução tecnológica, ao longo dos anos, ganhou força e movimentou o mercado, com tecnologia e ferramentas, nas soluções para tarefas, e automatizando processos. A *Inteligência Artificial (IA)* acompanhou esse processo, uma vez que é um campo de pesquisa da Ciência da Computação que visa buscar métodos computacionais para simular a capacidade humana de solucionar problemáticas ou reproduzir o pensamento do homem.

Após a Segunda Guerra Mundial, o matemático britânico Alan Turing, considerado o pai da computação, em 1950, propôs um teste cujo objetivo principal era determinar se uma máquina pode ou não pensar [Turing, 1950]. Esse teste baseava-se na distinção de uma conversa entre dois humanos e uma máquina, colocados em lugares distintos, caso uma terceira pessoa não identificasse qual era a máquina isso significaria que o programa de computador obteve sucesso, já que a máquina conseguiu parecer com um ser humano. Essa técnica se tornou um importante apoio para pesquisas em *IA*, pois alimentava a busca por novas tecnologias que pudessem ser aprimoradas.

Um subcampo da *IA* é o *Processamento de Linguagem Natural (PLN*) que estuda a aproximação do homem e da máquina numa interação mais natural. As principais aplicações da *PLN* estão na geração de uma linguagem natural, e na sua interpretação, simplificando o texto.

Assim, o objetivo deste trabalho é desenvolver um protótipo de um *Chatterbot* que possa ter conhecimentos específicos, no caso sobre a eqüinocultura, e que seja fácil a inserção de conhecimento, tornando-o popular e de fácil acesso às pessoas, que futuramente poderiam colaborar com o desenvolvimento e o aprimoramento do projeto. Para a construção do *Bot* foi adotada uma abordagem bastante simples, que consiste em proporcionar uma maior acessibilidade para os usuários, que por vezes deixam de usufruir de um programa pela complexidade que o mesmo exige. A criação de bots com conhecimento específico, sem dúvida é importante para simplificar sua utilização, podendo utilizá-los em FAQs, atendimento virtual, ensino a distância, consulta de conhecimentos e outros. A proposta de utilização nesses ambientes traz um maior nível de interação entre homem

e máquina, que é um ponto importante para impulsionar a contínua evolução computacional.

A estrutura do trabalho apresenta a seguinte sequência:

- Capítulo 2: descreve a evolução histórica e suas gerações, bem como os principais projetos desenvolvidos; apresenta o embasamento teórico sobre os *Chatterbots*, suas características e as áreas de atuação;
- Capítulo 3: fundamenta os conceitos que são utilizados para a construção de um *Chatterbot*, abordando a sua estrutura e técnicas de construção;
- Capítulo 4: apresenta o trabalho realizado, a construção das interfaces e da ferramenta para geração de conteúdo *AIML* (*Artificial Intelligence Markup Language).*
- Capítulo 5: aponta a conclusão do trabalho e sugere alguns pontos para trabalhos futuros.

Capítulo 2

Chatterbots:Evolução Histórica, Características e Usos

Inicialmente, os computadores eram tidos como objetos que efetuavam somente operações aritméticas e surpreendiam se realizassem qualquer atividade remotamente inteligente. Assim, quando em 1950, Alan Turing publicou um artigo "Computing Machinery and Intelligence" que destacava uma forma de assegurar se os computadores poderiam ou não pensar, foi o começo da evolução e a possibilidade de sonhar com modelos computacionais complexos capazes de surpreender a todos. O sistema baseava-se na impossibilidade de distinção entre ser humano e máquina por meio de um teste no qual o computador será aprovado caso um interrogador humano, após propor algumas perguntas por escrito, não descobrir se as respostas vêm de uma pessoa ou não. As pesquisas de *IA* têm dedicado pouco à aprovação no teste de Turing, acreditando que estudar os princípios básicos da inteligência seja mais importante do que produzir um exemplar que possa ter sucesso no teste [Russel e Norvig, 2002].

Em 1994, Michael L. Mauldin criou um termo para nomear um robô jogador cuja função principal é conversar: *Chatterbot* [Mauldin, 1994]. *Chatterbot* é um programa de computador que tenta simular o diálogo entre humanos, no qual o objetivo é responder as perguntas de maneira que se tenha a impressão de estar conversando com outra pessoa e não com um programa de computador [Teixeira, 2005]. Os Chatterbots constituem uma das diversas formas de humanização da máquina fornecidas pela *IA* .

Os *Chatterbots* podem possuir diversas nomenclaturas, sendo possível encontrar menções a *Chatter-bot*, *Chatbots*, *Bots* entre outras. No entanto, a origem do termo vem da palavra *Chat* (*Conversational Hypertext Access Technology* - Hipertexto de Conversação de Acesso à Tecnologia), e a palavra *Bot* que tem sua origem na abreviação da palavra checa Robota, que significa trabalho. A palavra robô teve origem na peça "R.U.R." de Karel Capek, escrita em 1921 [Leonhardt, 2005]. A sigla é uma abreviatura para "Rossum's Universal Robots" [Leonhardt, 2005]. Peça esta, que relata a história de um

cientista brilhante, Rossum, que cria uma substância química semelhante ao protoplasma, utilizada na construção de humanóides (robôs) com o objetivo que sejam inteligentes e façam todo o trabalho físico.

Por tudo explanado e para entender a criação de *Equus* é interessante destacar a evolução histórica dos primeiros Chatterbots criados que são: ELIZA, JULIA e ALICE.

2.1 ELIZA

Chaterbot criado em 1966, pelo professor Joseph Weizenbaum, no Massachussets Institute of Technology [Weizenbaum, 1966], sua implementação original foi em *SLIP (Serial Line Internet Protocol)* uma linguagem também criada por Weizenbaum e rodou em uma plataforma da IBM, conhecida como IBM 7094, um mainframe produzido pela IBM. Foi desenvolvido utilizando a idéia proposta na primeira geração de *Chatterbots*, o casamento de padrões. Tinha como objetivo simular a conversa de um psicanalista rogeriano [Rogers, 1951] com seu paciente. ELIZA então, esperava que o usuário (paciente) contasse e refletisse sobre os problemas que o incomodava e era estimulado a falar por meio de perguntas.

ELIZA possuía como arquitetura uma base de regras em que existiam as sentenças a serem analisadas, e um programa que captava a mensagem digitada pelo usuário. O procedimento do programa é bastante simples, a entrada fornecida pelo usuário é lida e se inspeciona a presença de uma palavra-chave, quando for encontrada, a sentença é transformada de acordo com uma regra associada a determinada palavra chave. Os problemas técnicos que ELIZA se preocupa podem ser resumidos em:

- Identificação da palavra-chave mais importante;
- Identificação do contexto mínimo que a palavra escolhida aparece;
- A escolha de uma regra de transformação adequada, e naturalmente fazer a transformação;
- A previsão de um mecanismo que permitirá uma resposta "inteligente" quando o texto não tiver palavra-chave;
- A previsão de mecanismos que facilitem a edição, particularmente a extensão de contexto.

A base de regras está estruturada por palavras-chave obtidas a partir do texto digitado pelo usuário, isto é feito em duas fases: a primeira são às regras de decomposição, responsável por separar a sentença de entrada em termos; a segunda é a de reconstrução, que retornava a saída para o usuário.

Inicialmente, ELIZA identifica as palavras mais importantes na frase e então aplica uma regra de modificação que é capaz de contextualizar as palavras, por exemplo: "você" seguido de "é" são classificados como uma afirmação. Para casos em que não se encontra nenhuma regra o programa retorna uma resposta já utilizada ou a uma livre de contexto. Na tabela 2.1 é mostrado um trecho do diálogo com ELIZA.

Considerando as duas fases para formulação da resposta, decomposição e reconstrução, pense na seguinte sentença sendo digitada pelo usuário: "It seems that you hate me", tem-se a decomposição mostrada na tabela 2.2.

O número N indica que a sentença vai ser composta por N palavras, assim o número zero (0) representa um termo composto por uma seqüência infinita de palavras. Palavras-chave como "you" e "me" determinam em qual termo da sentença elas devem aparecer. Na fase de reconstrução, um número N representa o í-esimo termo da decomposição, portanto tem-se a seguinte resposta: "What makes you think I hate you".

ELIZA: Hello. I am ELIZA. How can I help you?
You: I am tired
ELIZA: How long have you been tired?
You: I do not kwon
ELIZA: Why do you not kwon?

Tabela 2.1: Trecho de diálogo com ELIZA

(1) It seems that	(2) you	(3) hate	(4) me

Tabela 2.2: Decomposição da sentença

A idéia básica do método seria imaginar a sentença dita a um estrangeiro que não conhece muito bem a língua inglesa, mas possui uma ótima audição. Imagine a frase "I am very unhappy these days" , supondo que dita a ele e que apenas entenda "I am", uma possível resposta seria "How long have been very unhappy these days". O que ele fez foi aplicar uma espécie de modelo para a frase original, que específica que qualquer sentença da "I am" pode ser transformada em "How long have you been...".

Uma frase de entrada é varrida da esquerda para a direita, cada palavra é procurada em um dicionário de palavras-chave, se uma palavra é identificada como uma palavra-chave, então, aplica-se a regra da decomposição explicada anteriormente.

O programa se comporta melhor quando o correspondente humano é inicialmente instruído a falar, como se estivesse numa conversa com o psicoterapeuta (Teoria Rogeriana) [Rogers, 1951] . Esse modo de comunicação foi escolhido para a construção da ELIZA, porque a entrevista psiquiátrica é um dos poucos exemplos de linguagem categorizada

como didática natural em que um dos participantes da conversa não necessita de muito conhecimento do mundo real, já que a idéia é fazer o outro interlocutor falar.

Considerando o ambiente e o contexto da época em que ELIZA foi criada, pode-se ressaltar que apesar de simples era muito eficiente e retratava o ambiente de conversa em que foi proposta a sua construção, a psicologia Rogeriana.

As principais limitações de ELIZA são quanto a sua capacidade de memorização, não relacionando o que tinha sido falado antes. Outra limitação seria a construção de algumas respostas, pois ELIZA na tentativa de imprimir ao diálogo uma certa naturalidade, respondia ao interlocutor usando partes da própria entrada (Princípio da Teoria Rogeriana)[Rogers, 1951] , gerando às vezes diálogos um tanto confusos.

2.2 JULIA

O *Chatterbot* que marcou a segunda geração foi intitulado JULIA [Mauldin, 1994]. Criado por Michael Mauldin na Carnegie Mellon University, atuando como um personagem com a função de auxiliar outros usuários em um ambiente virtual, conhecido como TinyMUD (Multi-User Dungeons). Neste ambiente há vários usuários que controlam personagens que são jogados em uma rede com terminais que emulam ambientes através de interface de texto. JULIA vive nesta ambiente ajudando os usuários no mundo virtual, mapeando cavernas e enviando mensagens.

Uma característica interessante implementada em JULIA é a capacidade de lembrar informações sobre ações feitas anteriormente, para auxiliar na tomada de decisão do jogo. Sua versão original utilizava um algoritmo simples de interações do tipo “if-then-else”.

Posteriormente, foi introduzido um algoritmo baseado em redes neurais, que tinha por objetivo melhorar o desempenho das respostas. Dentro da rede, cada nó consiste em um conjunto de padrões, uma resposta simples, uma lista de nós ativos e outra de nós inibidos. Quando a entrada do usuário aciona algum padrão, os nós que contêm o modelo têm sua ativação estimulada e o de maior nível é selecionado, enviando sua resposta ao usuário. O código fonte de JULIA é proprietário, mas uma parte dele foi utilizada no desenvolvimento de outro *Chatterbot* de código aberto chamado COLIN [Foner, 1997].

JULIA possui uma personalidade curiosa, podendo apresentar sinais levemente alterados de humor e objetivos a serem alcançados, tais como, prestar informações aos jogadores, repassar mensagens de outros jogadores e explorar o ambiente virtual. Além disso, ela tem um mecanismo capaz de avaliar se está sendo útil em um ambiente. A partir do pressuposto de que ninguém interage com ela, por um longo período, decide explorar outros ambientes e procurar outros usuários.

2.3 ALICE

Um dos *Chatterbots* mais poulares ALICE (Artificial Linguistic Internet Computer Entity) [Wallace, 1995a], marca a terceira geração dos *Chatterbots* com a criação da *AIML* (*Artificial Inteligence Markup Language*), que é um padrão baseado no XML e apesar de ser simples, tem alcançado resultados melhores do que os *Chatterbots* anteriores. Para escrever em *AIML* é fundamental ser breve, conciso, interativo, gramaticalmente correto e estimular o humor para tornar mais natural a conversa e não apenas um jogo de perguntas e respostas [Wallace, 1995b].

O surgimento da *World Wide Web*, em 1994, gerou uma oportunidade de aumentar a divulgação e agregar maiores conhecimentos. Então, foram adotadas uma série de experimentos voltados para *Web*, como nível de interação com o usuário, e principalmente, a oportunidade de coletar amostras de uma linguagem natural em uma escala sem precedentes. Outra grande vantagem, com a utilização da plataforma *Web*, é a descentralização do conhecimento que agrega ajuda de vários outros desenvolvedores para a criação de interpretadores *AIML* em outras linguagens e de uma maneira geral, contribuindo para sua evolução [Wallace, 1995a].

A personalidade de ALICE está associada ao conjunto de padrões, pergunta-resposta, existentes nos arquivos *AIML*, denominados categorias. Cada categoria possui um estímulo ¡ pattern ¿ e uma resposta ¡ template ¿. A cada entrada do usuário, o *Bot* faz uma pesquisa na base *AIML* em busca de uma categoria e então, gera uma resposta correspondente ao determinado estímulo de entrada [Wallace, 1995b]. O modelo de aprendizagem utilizado é conhecido como aprendizado supervisionado, pelo fato de existir um responsável, denominado *Botmaster*, que pode inserir novos padrões uma vez que, não há um padrão específico permitindo um contínuo aperfeiçoamento dos conhecimentos do *Bot*. Com sua abordagem, ALICE também é capaz de colher informações das conversas adotando uma característica passiva-agressiva, na qual combina elementos de forma passiva com alguns de forma agressiva. Esse tipo de coleta permite verificar, por exemplo, um perfil dos usuários e quantificar algumas informações como porcentagem de faixas etárias.

Diferentemente de todos os outros *Chatterbots* citados anteriormente, a ênfase desse modelo está no *AIML*, e assim existem várias implementações do seu Kernel, que é o responsável pelo processamento da base de conhecimento, e como característica fundamental está o impulso no desenvolvimento de *Bots* com *AIML*. O Kernel foi desenvolvido em várias linguagens como: php, Java, C++, Perl, dentre outras, e sendo todas de código aberto. O *AIML* é capaz de armazenar informações sobre percepções e ações passadas, podendo utilizá-las para conduzir e melhorar a resposta final.

Os avanços para a construção de *Chatterbots* não param e muitos pesquisadores dedicam um aprimoramento ao método desenvolvido por Dr. Wallace (ALICE). Um

exemplo é a criação do *iAIML* [Neves e Barros, 2005], que consiste num mecanismo desenvolvido para tratamento de intenções em *Chatterbots* baseado na Teoria de Análises da Conversação, que propõe uma análise da conversação, tanto no nível local quanto no global. Nos experimentos realizados, 87 usuários mantiveram diálogos com dois *Chatterbots*: um em *AIML* e o outro em *iAIML*. O uso do *iAIML* comprovou ter 95% de grau de confiança, foram avaliados a qualidade do diálogo, o andamento da conversa e a coerência das réplicas [Neves e Barros, 2005].

A evolução histórica dos *Bots* seguiu de forma em que novas tecnologias foram sendo descobertas e implementadas. Hoje os esforços se concentram em aprimorar e viabilizar a construção de *Chatterbots* baseados no *AIML* e suas derivações. Trabalhos como construção de ferramentas e estratégias de geração de conhecimento automático somam as principais fontes de pesquisa.

(c)2001 Science Museum, London, all rights reserved
[ALICE 41124] [Dr. Richard S. Wallace]
Start at [2001/10/13 01:25:46]
*** JUDGE04 ***
JUDGE04[02:43:32] MY NAME IS JUDGE04
PROGRAM[02:43:32] That's a very nice name,
PROGRAM[02:43:32]Judge04.
JUDGE04[02:43:42] really why?
JUDGE04[02:43:42]
PROGRAM[02:43:42] For real. "judge04"just sounds nice.
JUDGE04[02:44:02] you are a pentium!!! aren't you
JUDGE04[02:44:02]
PROGRAM[02:44:02] Thanks and you are a delightful person too.

Tabela 2.3: Trecho do diálogo de Alice com o juíz

O Prêmio Loebner Prize é um concurso anual que avalia *Chatterbots* segundo o Teste de Turing. A tabela acima (2.3) mostra um trecho da conversa de ALICE com um dos juízes no Prêmio de 2001, em que o juíz chega a se confundir com sua naturalidade e capacidade de retornar respostas muito semelhantes a de um ser humano, além de não caracterizar em nenhum momento fuga de contexto [Loebner, 2009].

Existem alguns exemplares de *Bots* brasileiros, que em sua maioria são baseados na terceira geração, e utilizam como base para o desenvolvimento o *AIML*. Sendo assim a evolução histórica dos *Bots* segue de forma em que novas tecnologias foram sendo descobertas e implementadas. Hoje os esforços se concentram em aprimorar e viabilizar a construção de *Chatterbots* baseados no *AIML* e suas derivações, como no caso do objetivo

deste trabalho que é criar o *Equus*.

2.4 Características

Atualmente, a interação homem-máquina é baseada na passagem de informações por padrões de entrada que são processados e retornados. Contudo, estudos em *IA*, mais especificamente o *Processamento de Linguagem Natural*, busca uma interface de comunicação baseada na linguagem natural.

Uma primeira característica de um *Chatterbot* seria a capacidade de manter diálogo com seus usuários através de uma linguagem natural, para isto é necessário uma base de conhecimento e um interpretador capaz de analisar o padrão de entrada e consultar na base qual a melhor resposta. Na figura 2.1 é apresentado um modelo funcional básico de um *Chatterbot*.

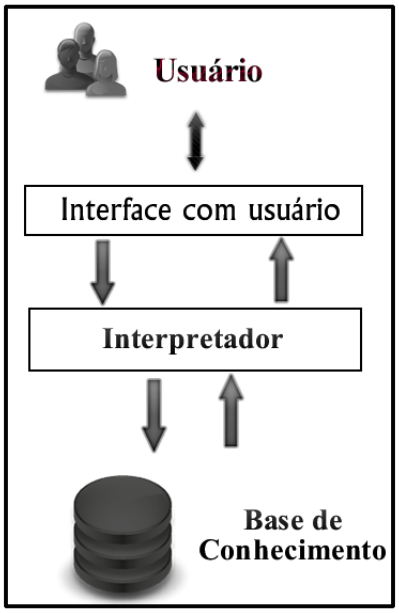

Figura 2.1: Modelo funcional básico de um *Chatterbot*

De uma maneira geral o conhecimento deve ser armazenado de alguma forma, assim como acontece quando aprendemos algo, precisamos armazenar este conhecimento para então utilizá-lo posteriormente. O diálogo entre humanos em sua língua natural realiza constantes consultas à base de conhecimento, aquela que criamos com os padrões passados durante a vida comunicativa. Para *Chatterbots* a idéia para a construção é a mesma, afinal precisamos ter um conhecimento prévio sobre os padrões e o conteúdo abordado durante a conversa. Então, para um diálogo entre homem e máquina, precisamos inicialmente de uma interface para transmitir os resultados, fluindo visualmente a comunicação. Contudo é necessário um interpretador capaz de avaliar a entrada fornecida pelo usuário e buscar

a melhor saída para aquela entrada na base de conhecimento já adquirida.

De acordo com Russel e Norvig [Russel e Norvig, 2002] o agente é uma entidade que percebe seu ambiente através de sensores e atua sobre ele através de efetuadores. Outra definição é que o agente seja um sistema de computador baseado em hardware ou software que desfruta as propriedades de autonomia, capacidade social, reatividade e pró-atividade [Wooldridge, 1995]. Não existe uma definição consensual para agentes, pois cada autor define o termo de modo que melhor se relacione com seu trabalho. Enfim, a maioria dos autores enumeram características que devem estar presentes nos agentes, que são [Leonhardt, 2005]:

- Autonomia: um agente autônomo deve ter controle sobre suas ações. Um agente pode ser autônomo em relação a outros agentes ou a um ambiente;
- Pró-atividade: capacidade de tomar a iniciativa para atingir seus objetivos, não se limitando apenas a estímulos do ambiente;
- Reatividade: capacidade de reação a estímulos e mudanças dentro do ambiente no qual encontra-se inserido;
- Continuidade Temporal: possibilidade de permanecer continuamente ativo;
- Capacidade Social: a sociabilidade implica na comunicação de um agente com outros agentes ou com humanos. A capacidade de comunicação pode levar a uma necessidade de cooperação e negociação entre agentes, que, por sua vez, são características que devem estar presentes em agentes quando necessário;
- Capacidade de Adaptação: possibilidade de alterar o comportamento baseado na sua experiência. Este processo também é conhecido como aprendizagem;
- Mobilidade: capacidade do agente de se mover dentro de um ambiente;
- Flexibilidade: habilidade de escolher dinamicamente uma ação ou sequência de ações em resposta a um estado do ambiente no qual se encontra.

O agente não precisa ter todas essas características, basta que tenha uma ou mais, afinal a presença de todos estes atributos, depende do tipo de aplicação que o agente está envolvido. Existe um certo conflito em se tratando da relação conceitual entre *Chatterbots* e agentes. *Chatterbots* são classificados muitas vezes como agentes conversacionais, pois exibem um comportamento não verbal apropriado, simulando todo o poder de comunicação de um humano. Para análise de *Chatterbots* na perspectiva de agentes utilizamos o modelo proposto por Russel e Norvig [Russel e Norvig, 2002], o qual apresenta uma arquitetura genérica para um agente racional que utiliza um componente de raciocínio

para mapear cada possível seqüência de percepções na melhor ação a ser tomada. Uma arquitetura genérica é apresentada na figura 2.2.

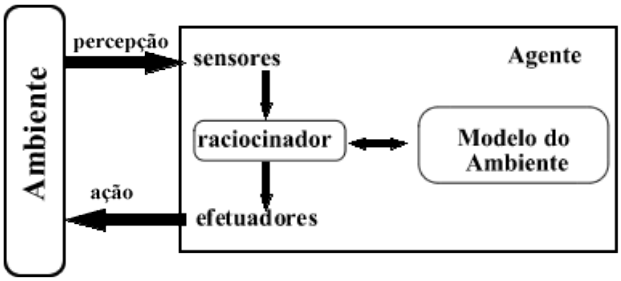

Figura 2.2: Arquitetura geral de um agente inteligente [Russel e Norvig, 2002]

Segundo Russel e Norvig é preciso se ter uma idéia bem definida sobre um conjunto de ações, percepções, objetivos e ambiente em que o agente atuará[Russel e Norvig, 2002]. Este conjunto é conhecido como PAGE (*Perceptions, Actions, Goals, Environment*). Os *Chatterbots* também podem ser caracterizados conforme esse conjunto. Galvão caracteriza as aplicações dos *Chatterbots* de acordo com a PAGE [Galvão, 2003], mostrada na figura 2.3.

Aplicação	**Percepções**	**Ações**	**Objetivos**	**Ambiente**
Entretenimento	Texto Escrito	Conversar sobre um tema qualquer ou guiar um usuário em um jogo, por exemplo	Divertir o usuário, geralmente simulando vida artificial	usuários em geral
Ensino a Distância (EAD)	Texto Escrito	Sugerir assuntos de discussão, tirar dúvidas ou efetuar demonstrações, por exemplo	Auxiliar na aprendizagem de um estudante	estudantes
Atendimento ao Consumidor	Texto Escrito	Responder dúvidas ou anotar reclamações, por exemplo	Minimizar custos de atendimento	consumidores
Comércio Eletrônico	Texto Escrito	Sugerir compra de produtos ou guiar o comprador pela loja, por exemplo	Maximizar vendas e satisfazer o cliente	compradores

Figura 2.3: Classificação dos *Chatterbots* segundo conjunto PAGE [Galvão, 2003]

Atualmente existe uma variedade de *Chatterbots*, que podem ser classificados em algumas categorias de acordo com sua finalidade, podendo existir *Bots* que pertencem a mais de uma categoria ao mesmo tempo. A seguir é apresentado exemplos de caracterização dos *Bots* de acordo com suas características [Leonhardt, 2005].

- **academic bots**: relacionados aos assuntos acadêmicos como sites de professores ou laboratórios de pesquisa.
- **design bots**: possuem ferramentas e habilidades para a produção de outros bots e agentes inteligentes.
- **commerce bots**: desempenham atividades de comércio na Internet
- **fun bots**: divertem usuários através de jogos, ambientes virtuais e personagens de realidade virtual.
- **government bots**: buscam informações em sites governamentais.
- **knowledge bots**: congrega agentes inteligentes, agentes de informação,agentes da web e muitas ferramentas inteligentes de busca.
- **news bots**: criam jornais personalizados e clips de artigos de jornais do mundo inteiro.
- **search bots**: buscas de bots e agentes inteligentes na Internet.
- **shopping bots**: fazem compras e comparações de preços para internautas.
- **stock bots**: monitoram o mercado de ações e mandam mensagens sobre os últimos preços e tendências.
- **update bots**: avisam ao usuário quando um site especificado foi atualizado ou modificado.
- **chatter bots**: são programas que simulam uma conversa com um ser humano.

Como visto há diversos campos que podem se beneficiar com a construção de *Chatterbots*, por isso a forma de conversação dos humanos vem sendo estudada e levada em consideração para a construção dos mesmos. Os *Chatterbots* fazem uso de diversas estratégias para manter e direcionar o diálogo analogamente ao ser humano, dando ilusão de inteligência, que são [Mauldin, 1994]:

- Manter a iniciativa do diálogo utilizando o constante questionamento;
- Inserir trechos da mensagem do usuário na respota;
- Aprofundar o diálogo com questionamentos, como por exemplo: 'Porque me pergunto isso?';
- Reconhecer quando uma conversa se torna repetitiva e iniciar um novo tópico;
- Fazer bom uso do humor para comentar algum assunto que esteja em foco;

2.5 Usos

Dentre os principais usos podemos destacar a utilização de *Chatterbots* para entretenimento, ensino à distância, atendimento ao consumidor e comércio eletrônico.

2.5.1 Entretenimento

O Entretenimento é uma das aplicações mais comuns para os *Chatterbots*. Para um bom desempenho deve-se ter um comportamento dinâmico e capaz de dar sensação da existência de uma personalidade. Como exemplos temos: ED[CONPET, 2008], ELIZA [Weizenbaum, 1966]. O ED é um projeto mantido pela Petrobrás que traz informações sobre o meio ambiente, conversando sobre uso eficiente de energia e combustíveis. É possível esclarecer dúvidas sobre esses assuntos, simulando um diálogo, e consequentemente, agregar conhecimento de uma maneira mais rápida. Como no caso de ELIZA que já foi citada anteriormente.

2.5.2 Ensino à distância

Um dos grandes desafios do Ensino à distância é manter os alunos estimulados com o aprendizado, que na maioria dos casos é pouco flexível. A utilização de Chatterbots nesse ambiente faz com que os alunos possam pesquisar informações de seu interesse num diálogo bem interessante pela interatividade.

2.5.3 Atendimento ao consumidor

Em toda organização existe a necessidade de uma área para atender os consumidores, no entanto o custo para manutenção de um serviço como este é bastante alto. Em geral a maioria das perguntas podem ser respondidas utilizando-se de uma base de conhecimento, ou seja, é possível apresentar esse serviço utilizando um *Chatterbot* com sua base de conhecimento formada por casos comuns a serem tratados em um serviço de atendimento ao consumidor. O grande diferencial para utilização de um sistema desses é a redução de gastos, pois poderia diminuir o custo com funcionários e equipamentos. Outro ponto forte é a aceleração do processo, em geral esses serviços são lentos, com fila de espera e com um *Chatterbot* seria acelerado e bem mais interessante.

2.5.4 Comércio eletrônico

Muitos usuários relutam em utilizar Comércio eletrônico por alguns fatores, tais como: dificuldade do consumidor em localizar o produto desejado; medo de que o cartão seja clonado e usado indevidamente; e ainda o fato de o consumidor não se sentir confortável

por não poder examinar o produto [Anderson, 1997]. Neste cenário o uso do *Chatterbots* pode humanizar o processo de e-commerce de modo que o consumidor não precise navegar entre páginas para chegar ao produto desejado, uma vez que ele poderá ser levado pelo *Chatterbot* diretamente até esse produto, com base no diálogo com o cliente, analogamente ao que acontece em uma loja física.

Capítulo 3

Fundamentação Teórica

A construção de *Chatterbot* envolve vários conceitos, técnicas e estruturas que auxiliam na sua execução, proporcionando um ambiente para a sua construção. Dentre as principais informações para a sua construção está sem dúvida a sua base de conhecimento, que é responsável pelo seu funcionamento e sua característica de conversação. Um dos principais problemas encontrados na construção de *Bot* é a falta de uma base de conhecimento consistente que permita um nível de interação que o torne distinto de uma máquina. Mas como essa distinção é bastante complexa, para que se tenha uma resposta satisfatória é necessário que o *Bot* tenha uma resposta na base de conhecimento, que esteja associada a entrada que é fornecida pelo usuário. Além do mais, o que distingue os seres humanos dos outros animais é o complexo sistema de mensagens estruturadas conhecido como a linguagem, o qual nos permite a comunicação sobre o que se sabe do mundo [Norvig, 2002].

3.1 Processamento de Linguagem Natural (PLN)

O processamento de linguagem natural consiste no desenvolvimento de modelos computacionais para a realização de tarefas que dependem de informações expressas em alguma língua natural. As pesquisa, em PLN estão voltadas em três aspectos da comunicação em língua natural [Pereira, 2007]:

- som: fonologia;
- estrutura: morfologia e sintaxe;
- significado: semântica e pragmática.

3.1.1 Fonologia

A fonologia é uma área da lingüística preocupada em avaliar e estudar os sons da língua, investigando o conhecimento fonológico dos falantes[Oliveira, 2007].

As aplicações hoje estão em uma vasta escala de aparelhos que saõ utilizados até no nosso dia a dia, um exemplo seria o reconhecimento de voz que é utilizado nos aparelhos celulares, para realizar uma chamada.

Outras aplicações deste estudo, que merecem destaque, são os conhecidos sistemas de síntese da fala, que por sua vez são capazes de criar um áudio baseado em um texto.

Os sistemas de diálogos em uma língua falada trazem um maior nível de interação e agrega outros valores do processamento de linguagem natural, pois envolvem a interação humano x máquina por meio de diálogos orais em linguagem natural.

3.1.2 Morfologia e Sintaxe

Uma análise morfológica é responsável por identificar palavras ou expressões em uma sentença, sendo que para a formação da mesma são utilizados delimitadores, espaços em branco e pontuação. Neste contexto, uma instância de uma palavra em uma sentença gramaticalmente válida pode ser substituída por outra do mesmo tipo, configurando uma sentença ainda válida (exemplo: substantivos, pronomes, verbos). Dentro de um mesmo tipo de palavra, existem grupos de regras que caracterizam o comportamento de um subconjunto de vocábulos da linguagem. Assim, a morfologia trata as palavras quanto a sua estrutura, forma, flexão e classificação, no que se refere a cada um dos tipos de palavras [Oliveira, 2007].

A sintaxe define a estrutura de uma frase, com base na forma como as palavras se relacionam nessa frase. O analisador sintático verifica a sentença a partir da sequência de *Tokens* recebidos e da sua adequação com a gramática da linguagem. Em outras palavras, o analisador é capaz de responder se a sentença está correta ou não de acordo com a gramática. Através da gramática da linguagem a ser analisada procura-se construir árvores de derivação para cada sentença, mostrando como as palavras estão relacionadas entre si.

3.1.3 Semântica e pragmática

A semântica se refere ao significado das sentenças. É um ponto em que se é possível tratar as ocorrências de ambiguidades no contexto abordado e a diferenciação entre o significado e o sentido. A compreensão da relação entre as palavras é tão importante quanto a compreensão das próprias palavras. Enfoques formais para a semântica tentam descrever o sentido de uma frase, mediante a tradução de sua estrutura sintática para uma

fórmula lógica-semântica. Como não existe uma correspondência imediata entre sintaxe e semântica, uma mesma estrutura sintática pode dar origem a diferentes representações semânticas [Oliveira, 2007].

A pragmática estuda a linguagem no contexto de sua utilização, é importante fazer uma interpretação do todo e não mais analisar o significado de suas partes, do ponto de vista léxico e gramatical. Um grande problema enfrentado no processamento de linguagem natural é a ambiguidade em estruturas complexas como anáforas e elipses [Oliveira, 2007].

3.2 Base de Dados

Base de Dados é conhecida como um sistema que utiliza uma origem de informação que mapeia todo o seu conteúdo em uma coleção de dados.As informações são armazenadas de forma persistente para que se possa realizar consultas e busca de conhecimento.

Fischler e Firschein definem o conhecimento como a informação armazenada, ou os modelos usados pela pessoa ou máquina para interpretar, predizer e responder apropriadamente ao mundo exterior [Fischler e Firschein, 1987]. Atentando para a definição do conhecimento foi observado que a base de conhecimento de um *Chatterbot* é de importância central, pois irá pré-determinar a qualidade de reconhecimento de uma determinada entrada e produzir uma resposta mais eficiente e coerente no contexto da conversação. Mesmo os *Bots* que utilizam técnicas simples de identificação das sentenças dos usuários, mas que possuem por sua vez bases de conhecimento robustas, conseguem apresentar um resultado interessante e até superior aos que utilizam técnicas mais complexas.

A construção de grande parte de *Chatterbots*, que atualmente possuem aceitação e desempenho em testes importantes, utilizam o *AIML* como base de conhecimento, o padrão foi desenvolvido por Dr. Wallace [Wallace, 2001] e utilizado na implementação de ALICE que basicamente estrutura as informações de forma simples baseado no XML. O padrão XML (*eXtended Markup Language*) foi desenvolvido em 1998 pelo consórcio W3. Sua proposta inicial e principal motivação era a criação de uma estrutura em que fosse possível vincular informações relativas a ela, e ao significado dos dados, tornando-o autodescritivo. E ainda, podendo categorizar os dados.

Essa facilidade em estruturar a informação fez com que para os projetos de *Chatterbot* fosse criado um padrão baseado no XML, o conhecido *AIML*. O XML é, portanto, mais uma meta-linguagem de marcação de texto. Formalmente pode-se dizer que é um conjunto de regras para definir *tags* semânticas que quebram um documento em partes e identifica diferentes partes do documento. É uma meta-linguagem de marcação que define as sintaxes pelas quais outras linguagens de marcação, específicas a um domínio, podem ser escritas [Harold, 2004]

A criação do *AIML* veio permitir que os padrões da base de estímulo-resposta possam

ser hospedados e processados pela *Web*, aumentando sua portabilidade. Sua idéia inicial era um sistema capaz de facilitar a implementação de um *Bot* baseado no ALICE, dentre os seus principais objetivos tem-se[Bush, 2001]:

- Fácil aprendizado;
- Utilização de um conceito mínimo , necessário para permitir o funcionamento de um sistema de estímulo-resposta;
- Compatível com XML;
- Facilidade de escrever em *AIML* e documentar o processo;
- Os objetos devem ser de fácil compreensão humana;
- O projeto *AIML* deve ser formal e conciso.

O *AIML* descreve uma classe de objetos de dados chamados de objetos *AIML*, que são constituídos de unidades denominadas categorias, e também é possível descrever parcialmente o comportamento que o *Bot* deverá assumir. O desenvolvimento do padrão foi iniciado por Dr. Richard Wallace e a comunidade ALICEBOT (software livre) durante 1995 e 2000 [Bush, 2001].

3.2.1 Estrutura

No *AIML* sua estrutura é basicamente igual ao XML, e segue todos os padrões de caracteres, formação, comentários e etc. O objeto *AIML* é formado por uma estrutura lógica e uma estrutura física. Sendo a estrutura física composta por unidades chamadas de categorias, e a estrutura lógica é formada por elementos e referências de caracteres, que são indicados em marcação explícita. Pode-se inserir comentários e instruções de processamento, assim como descrito na especificação XML, que não são tratados pelo interpretador *AIML*.

Para a criação de um documento em *AIML* deve-se respeitar as instruções e padrões adotados pelo XML, contribuindo para a construção de um arquivo *AIML* dentro dos padrões.

Formação Documento XML

Um documento XML é considerado bem formado se ele respeitar as seguintes situações:

- Reúne-se todas as restrições definidas na especificação do XML[Bray et al., 2000];

- Cada uma das entidades analisadas, que está relacionada no documento é bem formada;
- Contém um ou mais elementos;
- Elementos delimitados pelo início e fim de tags;
- Há exatamente um elemento, chamado de raíz, ou elemento do documento.

Elementos *AIML*

A estrutura do documento *AIML* se inicia com a declaração que informa a versão do XML, e assim tem-se o elemento *AIML* composto por:

```
<?xml version= number ?>
      <aiml version= \emph{number}>
      ...
      </aiml>
```

Figura 3.1: Estrutura Documento *AIML*

Um objeto *AIML* deve ter um atributo de versão, indicando a versão do *AIML* que o objeto requer. Para esta versão do *AIML*, a versão deve ser 1.0. Um elemento *AIML* ocorre como uma *tag* filha da *tag* < aiml >, que é chamado de um elemento de nível superior, a *tag* filha é chamada de categoria, <category>.

Outro elemento *AIML* é o "*topic*" representado por <topic>, nele há informações sobre a categoria. Sendo considerado um elemento opcional de alto nível, podendo existir um ou mais elementos na categoria.

As *tags AIML* são responsáveis pela estruturação da informação na base de conhecimento do *Bot*, através delas é possível retornar estímulos a determinadas entradas fornecidas pelo usuário. Uma entrada fornecida por um usuário é comparada aos padrões descritos na linguagem e, com base neste processo, são selecionadas ou construídas as respostas e todas as informações são estruturadas com base no *AIML*. A estrutura básica para a construção de um arquivo *AIML* utiliza as seguintes *tags*[Bush, 2001]:

- <topic> < /topic>: é um elemento opcional de alto nível que contém *tags* categoria, e por sua vez tem um atributo de nome necessário que deve conter uma expressão de padrão simples, e permite conter um ou mais elementos na categoria.
- <category> < /category>: é um nível superior (ou de segundo nível, se contido dentro de um tópico), que contém exatamente um padrão. A *tag* não possui atributos e corresponde a uma unidade de conhecimento.

- <pattern> < /pattern>: conteúdo entre as tags é uma expressão de padrão misto, em cada categoria existe exatamente um padrão e que deve ser sempre o primeiro elemento filho. Esta *tag* não possui atributos.

- <template> < /template>: a informação existente entre as *tags* é um conjunto de respostas que serão acionadas para determinada entrada do usuário, e também não possui atributos.

As *tags* citadas acima correspondem a estrutura principal de um arquivo *AIML*, no entanto, ainda existem outras *tags* que são utilizadas para determinados fins.

Caracteres Especiais

Um padrão de entrada delimitado pelas *tags* $< pattern >< /pattern >$ pode usar o caractere especial estrela (*) para casar com sentenças variadas fornecidas pelo usuário. Para recuperar o conteúdo que casou com o caractere especial se faz uso do elemento $< star >$ de *AIML*, representado na figura 3.2:

```
<?xml version= number ?>
   <aiml version= 1.0>
     <pattern>O que é *</pattern>
         <template>
               Eu não sei o que é <star />
         </template>
  </aiml>
```

Figura 3.2: Categoria com caractere especial (*)

Na figura 3.2, o elemento $< star >$ seria substituído pelo que vier depois da frase "O que é". Assim, se o usuário digitasse "O que é um carro", o *Bot* responderia "Eu não sei o que é um carro".

Intenções

Em linguagem natural é possível utilizar diversas frases com a mesma semântica. Frases como "Vocês tem filme de ação?", "Eu queria um filme de ação?"e "Gostaria de um filme de ação", têm um mesmo sentido pragmático e, portanto, devem ser tratadas por um *Chatterbot* de maneira idêntica.

```
<?xml version= number ?>
  <aiml version= 1.0>
     <pattern>Vocês tem filme de *</pattern>
          <template>
               <srai>Eu queria um filme de <star /></srai>
          </template>
  </aiml>
```

Figura 3.3: Categoria com *tag* < *srai* >

A figura 3.3 utiliza o elemento *AIML* <srai>, que faz uma chamada recursiva a outra padrão existente, dispensando a criação de novos templates para o mesmo sentido pragmático.

Tratamento de repetições

Utilizando-se da *tag* < *that* >, o *AIML* prevê o tratamento de repetição de sentenças, comparando sua forma. Deste modo as sentenças podem ser mais significativa.

Por exemplo:

```
Bot: Hoje eu estou feliz.
Homem: Isso é maravilhoso.
Bot: Mas, será feliz amanhã?
Homem: Ninguém pode dizer.
```

```
<category>
   <pattern> Isso é maravilhoso </ pattern>
   <that> Hoje eu estou feliz. </ that>
   <template> Mas, será feliz amanhã? </ template>
</ category>
```

Figura 3.4: Categoria utilizando tag < *that* >

Em *AIML* utilizando a *tag* < *that* > pode-se escrever um padrão para o diálogo acima, representado na figura 3.4.

3.3 Interpretador *AIML*

O desenvolvimento do interpretador para *AIML* foi iniciado por Richard Wallace- [Wallace, 2001]. Embora já como *software* livre, atraiu poucos participantes até migrar

para a sua primeira versão em JAVA, conhecida como Program A. Depois de algum tempo foi implementado interpretadores em outras linguagens, o que popularizou o *AIML* disponibilizando outros interpretadores [Alice, 2009].

O interpretador *AIML* é um módulo responsável por identificar a entrada do usuário na base de conhecimento *AIML* e retornar a resposta adequada. Os interpretadores utilizados atualmente são [Alice, 2009]:

- *Program M*: é implementado em uma linguagem conhecida como *SETL* (*Set Theory and Mathematical Logic*).
- *Program Z*: é implementado em Common Lisp.
- *Program N*: é implementado em C++.
- *Program D*: é implementado em JAVA.
- *Program E*: mais conhecido como "PHiliP".
- *Program V*: é uma implementação em Perl.
- *Program P*: mais conhecido como PASCALice, tendo sido desenvolvido em Delphi.
- *Program Y*: mais conhecido como PyAIML, é implementado em Python.
- *Program #*: é implementado em .NET.
- *Program R*: é implementado em Ruby.

3.3.1 O algoritmo Graphmaster

O Graphmaster é o algoritmo utilizado por ALICE para avaliar as sentenças de entrada e ainda, buscar na base *AIML* uma resposta que satisfaça a entrada. A base *AIML* é carregada na forma de um Graphmaster, que é basicamente um grafo[Wallace, 1995b].

A estrutura do Graphmaster consiste de uma coleção de nós chamados Nodemappers, que são responsáveis por mapear os ramos de cada nó. As ramificações são palavras sozinhas ou os caracteres curingas. A raiz do Graphmaster é um Nodemapper, e existe uma associação (um nó) para cada uma das primeiras palavras de todos os padrões (cerca de 40.000 no caso da ALICE) da base *AIML* identificadas na $tag < pattern >$. O número de nós folha no grafo é igual ao número de categorias, e cada nó folha contém o conteúdo armazenado na $tag < template >$[Wallace, 2001] [Wallace, 2001].

Para ilustrar melhor o seu funcionamento, suponha que a entrada fornecida pelo usuário comece com a palavra "CAVALO" Primeiro verifica se o nó do pai (Nodemaper), contém o caractere especial "_", então procure o subgrafo a partir do nó que corresponde

a "_". Tente todas as palavras restantes para a sentença após a palavra "CAVALO" para ver se alguma casa. Do contrário, volte ao Nodemapper. Agora tenta-se procurar pela palavra "CAVALO", voltando ao nó raiz e procura-se novamente o subgrafo que inicia-se com o termo, caso não encontre, volta-se ao nó raiz e busca pelo caractere especial "" e tenta-se todas as palavras e caracteres restantes da sentença de entrada após a palavra "CAVALO", para ver se há algum casamento.

A idéia do algoritmo é buscar uma resposta na base, analisando a entrada do usuário e o uso de caracteres coringas disponíveis na linguagem *AIML*.

Uma metáfora conveniente para os padrões de *AIML*, e talvez também uma alternativa para o armazenamento de dados de padrões e modelos, é o sistema de arquivos. Se você usa Windows, Unix ou Mac o mesmo princípio é válido. O sistema de arquivo tem uma raíz, como 'c:\' ou '/'. A raíz tem algumas ramificações que são arquivos, e algumas que são pastas. As pastas, por sua vez, têm filhos que podem ser ambos, pastas e arquivos. Se visualizarmos a estruturação do sistema de arquivos como uma árvore, tem-se que os nós folha da árvore são os arquivos. Cada arquivo tem um nome de caminho, que explicita a sua posição exata dentro da árvore [Wallace, 2001]. Utilizando o caminho

```
''C:\meusdocumentos\minhasimagens\01.txt''
```

tem-se que o arquivo 01.*txt* está localizado abaixo de um determinado conjunto de ramos da raiz.

O Graphmaster está organizado exatamente da mesma maneira. Pode-se escrever um padrão como "I LIKE TO *" como "G: / I / LIKE / TO / star". Todos os outros padrões que começam com "I" também vão para o "G: / I / pasta". Todos os padrões que começam com "I Like" vão para o "G: / I / LIKE / subpasta". Pode-se verificar que a pasta "G: / I / LIKE / TO / star" tem um único arquivo chamado "template.txt" que contém a resposta para essa entrada [Wallace, 2001].

O algoritmo de correspondência especifica um procedimento eficaz para a consulta dos arquivos para um determinado arquivo chamado "template.txt", cujo o nome do caminho distingue todos os template.txt existentes, retornado apenas o template que se encaixa com o caminho associado no texto de entrada.

Além do mais, pode-se visualizar a "compreensão" do Graphmaster na hierarquia do sistema de arquivos, em que todos os padrões de prefixo comum tornam-se ramos da raiz. É evidente que esta hierarquia é o melhor método de armazenamento de uma matriz linear simples, ou o armazenamento de dados de padrões [Wallace, 2001].

De maneira geral é possível resumir o funcionamento do Graphmaster Pattern Matching, como o funcionamento de um dicionário ou enciclopédia. Se você quiser procurar uma palavra ou frase, não comece no início ou no final, a busca é através de cada entrada até encontrar uma correspondência, primeiro vá para a seção que corresponda a primeira

letra ou palavra. Então, pula-se para outra seção que contém um início de um conjunto com uma próxima letra ou palavra, e então continua-se neste processo até localizar a palavra ou frase.

Toda essa explicação da parte estrutural de um *Chatterbots* é de extrema relevância para que o protótipo *Equus* seja construído, no qual, utilizamos o algoritmo graphmaster para implementar o interpretador que busca informações na base de conhecimento que é em AIML.

Capítulo 4

Equus: Características e Construção

4.1 Motivação

A idéia de construção de um *Chatterbot* é uma forma de gerar um conhecimento que pode ser acessível por várias pessoas em tempo real com uma grande interatividade, estimulando a propagação do conhecimento.

Bots em sua forma mais simples podem atender uma infinidade de situações que vão desde atendimento a um cliente, como a uma importante fonte de pesquisa. A construção de mecanismos para maximizar a eficiência dos *Bots* tem conquistado grandes pesquisas.

A motivação principal deste trabalho é propor uma solução de um *Bot* que seja de fácil implementação, que utilize apenas ferramentas livres, hospedagem grátis e mão-de-obra vasta nas ferramentas e soluções, para que seja possível sua expansão e aprimoramento. Sendo assim, é disponibilizado um interpretador de AIML, uma base de conhecimento genérica, responsável por armazenar informações de conhecimento geral e uma base de conhecimento específica gerada com o auxílio de uma ferramenta para a criação de conhecimento específico sobre a equinocultura.

4.2 Conhecimento Específico

A criação de um *Bot* com um conhecimento genérico demanda tempo devido a uma capacidade de agregar conhecimento que seja manual e supervisionada pelo *botmaster*. Mesmo com uma grande quantidade de informação, o *Bot* ainda não conseguirá abordar todo o contexto de muitos assuntos, então, a elaboração de um sistema com um conhecimento específico auxilia por reduzir os padrões nos documentos *AIML*, além de se tornar uma importante fonte de consulta de informações sobre um determinado assunto.

De maneira geral, quando se ataca um contexto genérico, corre-se um grande risco de diminuir o nível de interatividade entre o *Bot* e o usuário, que pode ser minimizado

quando aborda apenas uma pequena área do conhecimento, agregando maior valor e facilidade quanto a construção de um *Bot* com conhecimentos completos sobre determinados assuntos.

Como constatei a inexistência de um modelo de *Chatterbot* operante na equinocultura, que representa atualmente uma importante faixa do agronegócio brasileiro, decidi criar o *Equus*.

A equinocultura, resultado da junção dos termos equino + cultura, no qual equino é um mamífero quadrúpede (cavalo) e cultura sendo o ato, o efeito ou o modo de cultivar a criação de certos animais [Buarque, 1989]. E, sendo assim, considero que o maior benefício desta abordagem é o uso de técnicas de *IA* na aplicação de um *Chatterbot* em um contexto ainda pouco explorado.

4.3 Equus

O *Chatterbot* foi denominado *Equus*, que significa cavalo em latim. Seu uso está associado a nomes científicos (Equus caballus, Equus quagga). E ainda, existe uma definição criada por Monty Roberts, conhecido mundialmente de"O Encantador de Cavalos" [Roberts, 1996], como uma linguagem de comunicação corporal, baseada no conceito de "Avançar-Recuar" adotadas pelos indígenas norte-americanos.

O objetivo é utilizar conceitos de *IA* na construção de um mecanismo que provê conhecimento aos apaixonados por essa cultura, que fascina muitas pessoas. O nome do *Bot* (*Equus*) denota um nível de interação para os amantes da equinocultura e uma importante forma de difundir o conhecimento.

Utilizando as técnicas de construção de um *Bot* apresentadas anteriormente e auxiliado pela ferramenta de geração de conteúdo *AIML*, o intuito é desenvolver um produto que difunda conhecimento a população e seja de fácil construção viabilizando o seu uso em outras áreas.

4.4 O interpretador

O interpretador utilizado foi o program E, conhecido como "*PHiliP*", uma implementação em PHP disponível pela comunidade alicebot.org [A.L.I.C.E, 2009]. A utilização da plataforma em PHP tem suas vantagens quanto a portabilidade, facilidade de hospedagem do serviço e crescimento da comunidade e utilizadores. Um dos principais objetivos do projeto é ter de posse um sistema facilmente escalável e portátil, que garanta maior visibilidade, sendo voltado para *Web.*

Equus utiliza o interpretador para prover a comunicação entre o usuário e a base de

conhecimento em *AIML*. O interpretador é construído embasado no algoritmo Graphmaster que busca a melhor resposta para informar ao usuário.

4.4.1 Software Livre

De modo mais preciso, existem quatro liberdades que os usuários de Software Livre podem usufruir [Foudation, 2008]:

- Liberdade de executar o programa, para qualquer propósito;
- Liberdade de estudar como o mesmo funciona e adaptá-lo para as suas necessidades;
- Liberdade de redistribuir cópias de modo a ajudar o próximo;
- Liberdade de aperfeiçoar o programa e liberar os seus aperfeiçoamentos de modo que toda comunidade se beneficie.

4.5 FGCE: Ferramenta de Geração de Conteúdo Específico

A ferramenta construída agrega um mecanismo facilitador para a construção e manipulação das bases *AIML*, agilizando o processo de criação de conhecimento para o *Bot*. A geração do conteúdo não é automática, mas contribui na construção de bases de conhecimentos específicos.

Para o funcionamento da FGCE utiliza-se um dicionário de termos técnicos associados ao tema desejado, no caso deste projeto, termos relacionados a equinocultura, contexto de atuação do equus. A figura 4.1 representa a idéia conceitual da FGCE.

Os elementos básico do processo para a criação de conteúdo em *AIML* são: o texto de entrada, o dicionário de dados e a criação de uma categoria *AIML*.

A funcionalidade do protótipo é auxiliar na construção de bases de conhecimento específico em *AIML*. Como entrada, utiliza-se o texto e então o processamento consiste na busca por palavras-chave associadas no dicionário de termos técnicos. Para um melhor entendimento apresenta-se em etapas o funcionamento da ferramenta:

- Texto de entrada, o usuário da ferramenta fornece um texto contendo informações sobre o conhecimento que ele deseja adicionar ao bot, lembrando que o dicionário deve estar atualizado com o novo conhecimento que se deseja adicionar.
- O processamento inicial consiste em encontrar a ocorrência dos termos técnicos no texto de entrada, que foram adicionados no dicionário de termos técnicos.

- A cada termo encontrado no texto o programa gera uma estrutura AIML, ressaltando as frases identificadas no texto que tratam do mesmo contexto de conhecimento do *Bot*, delimitado pelo dicionário de dados.

- Se o programa não encontra esses termos então não cria nenhuma estrutura em *AIML*, apenas finaliza a execução do processo.

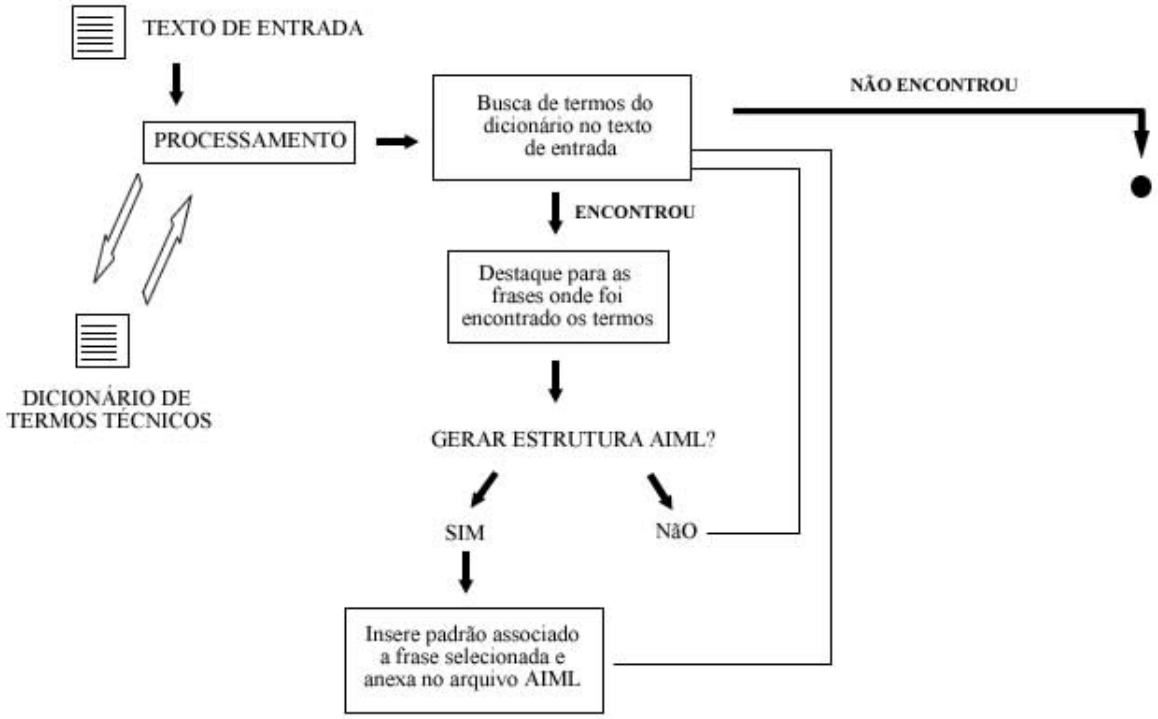

Figura 4.1: Funcionamento da ferramenta de geração de conteúdo *AIML*

O processo é bastante simples e funciona mais como um módulo para estender o potencial de ação do *Botmaster* e, assim, inserir novos conhecimentos. É implementado em PHP, e anexado ao projeto inicial do interpretador, somando uma nova funcionalidade de construção de *Chatterbots* com conhecimento específico.

Outra ação desempenhada pelo usuário da FGCE é a associação de padrões para as categorias previamente criadas e fornecidas pelo sistema, de acordo com o retorno do processamento do texto de entrada e o dicionário de termos técnicos. Esta etapa envolve conhecimento sobre *AIML* por parte do utilizador, em que é definido se determinada entrada do usuário irá estimular a saída extraída do texto.

Para a sua construção, inicialmente cria-se um documento texto contendo as palavras-chave, termos técnicos sobre determinada área do conhecimento, no caso utiliza-se a equinocultura. O dicionário possui estrutura simples e as palavras são separadas por espaços " ", é salvo como ".txt" sendo selecionado o arquivo de dicionário no momento em

que se fornece o texto de entrada, isto viabiliza a existência de vários arquivos separados com contextos de atuação diferentes, se assim for desejado.

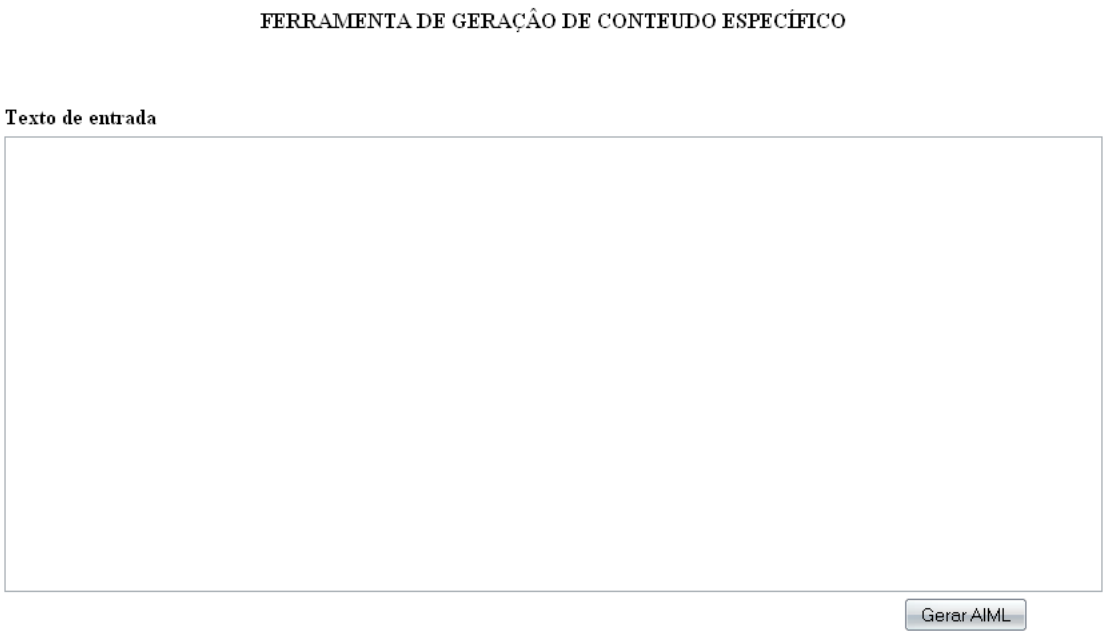

Figura 4.2: Interface FGCE

O texto de entrada é inserido na caixa de texto, disponível na interface figura 4.2, e então, inicia-se o processamento clicando em gerar *AIML*. O resultado do processamento é salvo em um arquivo aiml.

Considere o seguinte texto de entrada:

```
"A equitação é um esporte muito difundido. Ocupa entretanto,
uma posição muito especial entre os esportes devido ao fato
de seu instrumento ser uma criatura viva e animada.
O cavaleiro deve possuir não apenas qualidades físicas, tais
como força, destreza, resistência, e intelectuais, tais como
capacidade de decisão, paciência, presença de espírito e
audácia, como também conhecimentos sobre fenômenos físicos,
fisiológicos e psicológicos.
Montar a cavalo é aplicar com precisão as leis da mecânica
estática e dinâmica, bem como noções de anatomia, fisiologia
do próprio corpo e do cavalo".
```

Suponhamos um dicionário composto por: cavalo, cavaleiro, equitação, instrutor, montar. Ao fornecer ambas entradas no texto, o dicionário será gerado em *AIML* demonstrando os seguintes padrões:

```
<aiml version=1.0>

  <category>
   <pattern> EQUITAÇÂO * </pattern>
   <template> A equitação é um esporte muito difundido </ template>
  </category>

  <category>
   <pattern> CAVALEIRO * </pattern>
   <template> O cavaleiro deve possuir não apenas qualidades
              físicas, tais como força, destreza, resistência,
              mas também intelectuais, tais como capacidade de decisão,
              paciência, presença de espírito e audácia, e mais
               ainda, conhecimentos sobre fenômenos físicos,
               fisiológicos e psicológicos.
   </ template>
  </category>

  <category>
   <pattern> MONTAR CAVALO * </pattern>
   <template> Montar a cavalo é aplicar com precisão as leis
              da mecânica estática e dinâmica, bem como noções
              de anatomia, fisiologia do próprio corpo e  do
              cavalo.
   </ template>
  </category>

</aiml>
```

Então, o *Botmaster* agora possui condições de avaliar os padrões e trabalhar as entradas da melhor maneira possível quando for o caso. Inserir palavras para melhor filtrar o casamento da entrada do usuário com o conteúdo gerado.

É importante lembrar que a ferramenta de geração de conteúdo para o seu sucesso, depende da interação humana na avaliação final dos resultados, para que possa atender

a sua finalidade. No entanto, ela consegue extrair o conhecimento específico do texto acelerando o processo de desenvolvimento do *Bot*.

4.6 Interface

A interface é a forma mais importante de interação entre homem-máquina, é o ponto de partida para uma comunicação eficiente, se tem falha neste ponto compromete-se todo o processo. A figura 4.3 e 4.4 representa a interface do *Equus*.

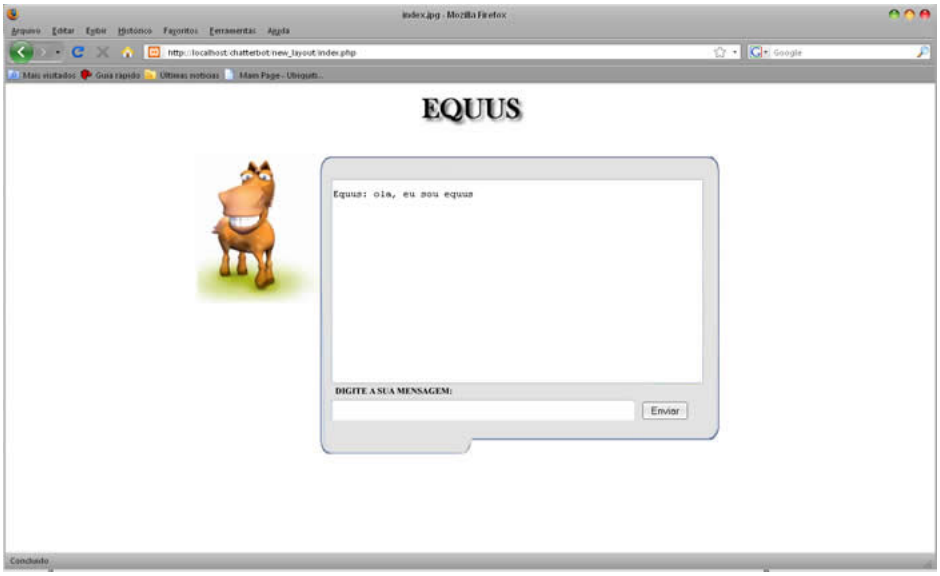

Figura 4.3: Interface Equus no Navegador

A portabilidade do PHP e HTML, garantem execução da interface em diferentes sistemas operacionais e em diferentes navegadores, podendo ser utilizado sem quaisquer pré-requisitos do usuário. Isto é uma das principais preocupações com o projeto, afinal a diversidade atual de navegadores e sistemas operacionais estão em constante evolução, e deste modo ter um sistema desenvolvido com essa preocupação garante a confiança dos usuários, pois eles sabem que independentemente da plataforma que são utilizadas poderão usufruir do mesmo.

A figura 4.3 mostra a execução do *Equus* em um navegador Mozilla Firefox, mantido pela comunidade livre. Já a figura 4.4 aproxima a interface para observação dos detalhes e a estruturação gráfica que se assemelham aos mensageiros que comumente são utilizados por todos.

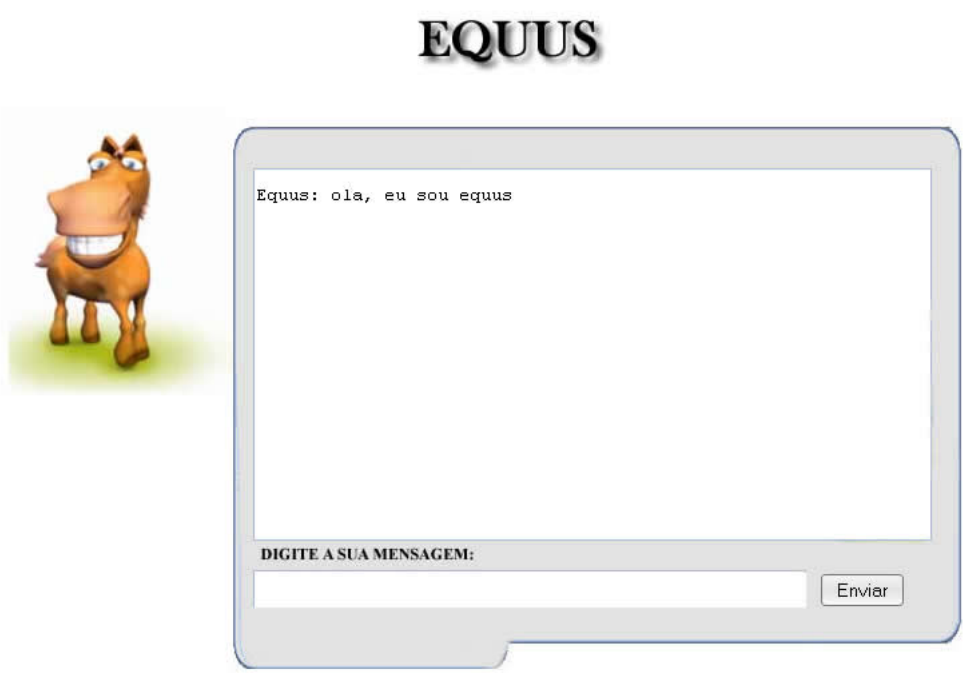

Figura 4.4: Interface Equus

4.7 Base de Conhecimento

Base *AIML* constitui parte imprescindível na execução do *Chatterbot Equus*, nela existe todo o conhecimento que o *Bot* será capaz de processar e informar aos usuários. A sua construção segue duas etapas principais, uma utilizando a base de conhecimento padrão, gerada para atender um contexto comum de entradas com mais frequência de utilização. A outra etapa é o uso da ferramenta construída para gerar conhecimento *AIML*, de acordo o conhecimento específico que o *Bot* atuará.

Ambas as etapas são de contínuo processo de atualização que com o tempo serão inseridos mais padrões e, consequentemente, mais informações poderão ser tratadas pelo *Bot*.

A segunda etapa é totalmente auxiliada pelo uso da ferramenta FGCE, em que cria-se um dicionário com termos específicos e então, fornece textos no mesmo contexto do dicionário para a geração de arquivos *AIML*.

O *Equus* é uma iniciativa para difundir uma área de conhecimento, que atualmente conta com poucos padrões de conhecimento, porém com a criação dele espero que, em breve, seja possível contar com a sua ajuda para esclarecer e popularizar informações sobre equinocultura. Sua arquitetura simples garante a sua funcionalidade, e o uso de sua idéia para a construção de outros *Bots* com conhecimento em outras áreas, trafegando e disponibilizando informações por toda *Web*.

4.8 Modelo funcional *Equus*

Equus é divido em 4 funções principais que juntas traduzem o funcionamento do *Chatterbot*, a figura 4.5 representa o modelo funcional do *Equus*

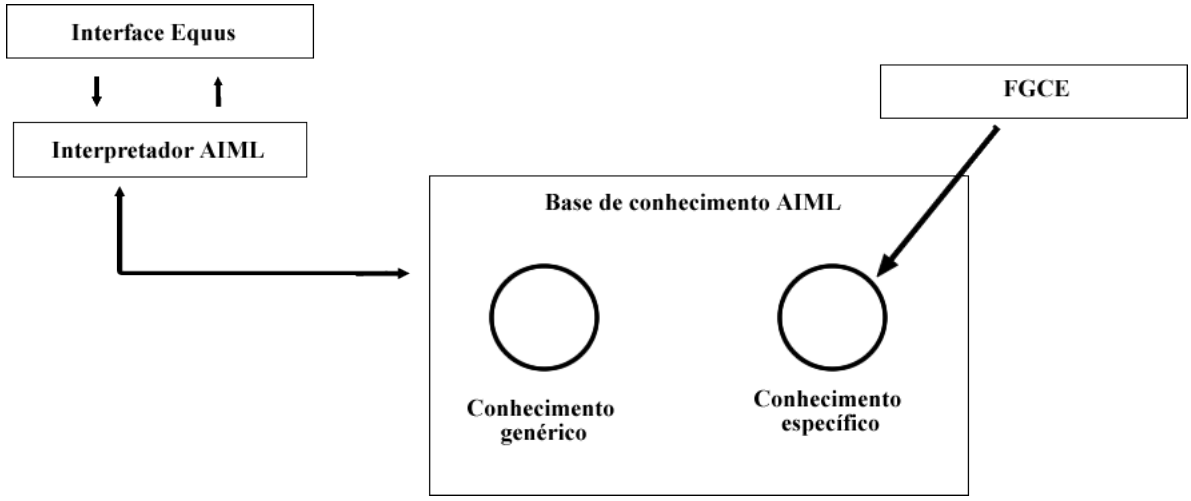

Figura 4.5: Modelo funcional *Equus*

Na figura acima é possível visualizar os componentes que formam o *Chatterbot*, bem como o tráfego da informação por seus componentes.

Inicialmente temos a interface com o usuário que é responsável por receber a mensagem do usuário e mostrar a resposta estimulada pela respectiva entrada. A entrada do usuário é utilizada pelo Interpretador para buscar na base de conhecimento *AIML* uma melhor resposta. Por sua vez a base de conhecimento do *Equus* foi construída dividindo-se de acordo com o tipo de conhecimento armazenado, basicamente temos uma base menor construída para tratar de conhecimentos genéricos, alguns diálogos cotidianos.

A outra base de conhecimento é gerada através da FGCE que consegue abstrair de um contexto específico pré determinado conhecimento para a base *AIML*, tornando o processo de criação de um *Bot* simplificada para uma área de conhecimento específica.

Equus foi modelado para que uma implementação para outra área de atuação seja prática e rápida tornando seu uso facilitado utilizando todas ferramentas livres e tendo o tempo de criação da base de conhecimento reduzido com o uso da FGCE.

Capítulo 5

Conclusão

Neste trabalho procurei demonstrar o desenvolvimento de um *Chatterbot* desde a evolução histórica dos Bots, suas características e usos até a construção do *Equus*. A idéia é modificar a abordagem da construção de um *Bot* criando ferramentas como FGCE, facilitando a sua popularização e, consequentemente, a sua evolução. E ainda, na criação de novas técnicas e abordagens em cima de um futuro desenvolvimento do *Equus*. O intuito deste foi criar um *Bot* com conhecimento específico, e para isso utilizei um ferramenta para gerar *AIML* de áreas específicas de conhecimento. Foi possível a geração desse conteúdo ainda que de forma simples, pois há uma exigência de uma revisão do *Botmaster*, que corresponde a um importante passo para a construção de uma solução que consiga recuperar informações a partir de padrões textuais.

Em se tratando da aceitação do uso de *Chatterbots*, uma tecnologia cujo principal atrativo é o uso da linguagem natural e a sua facilidade de interação com o usuário, verifico que o uso da solução representa um enorme potencial no sentido de aumentar o interesse e a curiosidade dos usuários.

O presente trabalho salienta, que a *IA* tem buscado desenvolver diferentes meios de fazer com que um computador possa realizar suas tarefas de forma racional, não mais se prendendo a um conjunto pré-programado de instruções, e deste modo, envolvendo maior interatividade.

Ao longo de várias décadas, desde o surgimento dos primeiros computadores, a compreensão de línguas naturais tem representado um grande desafio. Mesmo em recortes para domínios e cenários muito particulares, a compreensão ainda é um grande empecilho.

Os esforços não param e dentre os pontos mais interessantes para se estudar e solucionar, podemos citar:

- A criação automática de conhecimento.
- Estruturar semanticamente o conhecimento, observando desvios de linguagem.

- Sistemas de busca e recuperação de documentos ou informações a partir de padrões textuais.

- Avançar no processamento da língua portuguesa, criando mecanismos facilitadores para extração e manipulação do conhecimento.

A criação de uma ferramenta, como a FGCE, usada no *Equus* para auxiliar na montagem da sua base de conhecimento foi um dos principais objetivos deste trabalho, pois desse modo obteve-se uma solução eficiente para a criação do conhecimento, de um sistema capaz de aprender e de gerar conhecimento automático. O objeto de análise auxilia na construção dessa base, mas não exclui o auxílio do *Botmaster* para manipulá-la. Seu uso potencializa a construção do *Bot*, mas ainda não consegue gerar automaticamente conhecimento. No caso da construção do *Equus*, ela auxiliou na informação sobre um contexto que não é de conhecimento comum, e então, associei dados e informações escritas por outras pessoas e inseri na sua base de conhecimento.

A construção da base de conhecimento, referente a equinocultura, só foi possível devido ao uso da ferramenta em que apenas foi gerada as informações iniciais para alimentar o dicionário de termos técnicos. É importante que seja feito aprimoramentos na ferramenta e que ela consiga gerar conhecimento sem a interação com o *Botmaster*, consolidando um modelo perfeito de aprendizagem.

Dentre trabalhos futuros que posso considerar, avalio a criação de uma ferramenta que forneça uma forma de geração automática do *AIML*, que ao fornecer um texto de entrada a ferramenta consiga não só criar uma estrutura *AIML* correspondente, mas também analisar semanticamente o contexto e as informações apresentadas. Como a implementação de mecanismos de TTS(Text To Speech), para tradução do texto de saída do *Bot* em voz, aumentando o nível de interação sensorial de uma conversação. A criação de um sistema gerador de *Bots*, que contribua para popularizar a sua utilização e a fabricação de conhecimento espontâneo, em que os próprios utilizadores ajudariam a montar uma base de conhecimento maior e mais funcional.

O *Equus* é uma importante ferramenta para disseminar o conhecimento sobre equinocultura, apesar de possuir poucos padrões, com o tempo pode-se chegar a um nível maior de conhecimento agregado ao *Bot*. Há alguns termos relacionados a equinocultura que ele ainda não conhece, mas que serão adicionados com o passar do tempo, pois a preocupação não era exatamente o quanto ele sabe, mas em como ele irá adquirir esse conhecimento. Para um melhor funcionamento é importante que se trabalhe na sua base de conhecimento, adicionando ao *Equus* mais informações.

Assim, o *Chatterbot Equus* se tornará um programa importante para aqueles que procuram conhecimento específico sobre a equinocultura de maneira prática e ao mesmo tempo interessante.

Referências

A.L.I.C.E (2009). Aiml implementations. Disponível em http://www.alicebot.org/downloads/programs.html. Aceso em 01 de novembro de 2009.

Anderson, C. (1997). A survey of electronic commerce in search of the perfect market. Disponível em http://www.economist.com/surveys/showsurvey.cfm?issue=19970510. Aceso em 10 de junho de 2008.

Bray, T., Paoli, J., Sperberg-McQueena, C. M., e Maler, E. (2000). Extensible markup language (xml) 1.0 (second edition). Disponível em http://www.w3.org/TR/2000/REC-xml-20001006. Aceso em 01 de novembro de 2009.

Buarque, A. (1989). Dicionário da língua portuguesa.

CONPET, E. (2008). Publicação eletrônica. Disponível em http://www.ed.conpet.gov.br/converse.php. Aceso em 10 de junho de 2008.

Foner, L. N. (1997). Entertaining agents: A sociological case study. In Johnson, W. L. e Hayes-Roth, B., editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 122–129, Marina del Rey, CA, USA. ACM Press.

Galvão, A. M. (2003). Persona-aiml: uma arquitetura para desenvolver chatterbots com personalidade. Dissertação (Mestrado em Ciência da Computação) Universidade Federal de Pernambuco.

Leonhardt, M. D. (2005). Doroty: um chatterbot para treinamento de profissionais atuantes no gerenciamento de redes de computadores.

Loebner, H. (2009). The loebner prize in artificial intelligence. Disponível em http://www.loebner.net/Prizef/loebner-prize.html. Última consulta em 29 de outubro de 2009.

Mauldin, M. L. (1994). Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. Disponível em http://www.lazytoad.com/lti/pub/aaai94.html. Acesso em 15 de maio de 2008.

Neves, A. M. M. e Barros, F. A. (2005). iaiml: Um mecanismo para tratamento de intenção em chatterbots. Disponível em www.sbc.org.br/bibliotecadigital/download.php?paper=355. Aceso em 21 de outubro de 2009.

Oliveira, F. A. D. (2007). Processamento de linguagem natural: princípios básicos e a implementação de um analisador sintático de sentenças da língua portuguesa. Disponível em http://www.inf.ufrgs.br/gppd/disc/cmp135/trabs/992/Parser/parser.html. Aceso em 29 de outubro de 2009.

Rogers, C. (1951). Client-centered therapy: Its current pratice, implications and theory. Boston: Houghton Mifflin.

Russel, S. e Norvig, P. (2002). Artificial inteligence: A modern approach. Pearson Education, Inc.

Teixeira, S. (2005). Chatterbots:uma proposta para a construção de bases de conhecimento. Disponível em http://www.multicast.com.br/sergio/tuxbot-dissertacao-mestrado-sergio-teixeira.pdf. Acesso em 15 de fevereiro de 2008.

Wallace, R. (1995a). Alice. artificial linguistic internet computer entity - the a.l.i.c.e a.i. foundation. Disponível em http://www.alicebot.org. Acesso em 18 de setembro de 2009.

Wallace, R. (1995b). The anatomy of a.l.i.c.e. Disponível em http://www.alicebot.org/anatomy.html. Acesso em 30 de setembro de 2009.

Wallace, R. (2001). Aiml pattern matching simplified. Disponível em http://www.alicebot.org/documentation/matching.html. Aceso em 01 de novembro de 2009.

Weizenbaum, J. (1966). Eliza–a computer program for the study of natural language communication between man and machine. Disponível em http://i5.nyu.edu/ mm64/x52.9265/january1966.html#reference4. Aceso em 30 de março de 2008.

Wooldridge, M. (1995). Intelligent agents: theory and practice. page 115 152. The knowledge engineering review.

Apêndice A

Código Fonte

A.1 Interface Equus

```
<?

//require_once "admin/botloader.php";
$HTTP_POST_VARS['botname'] = 'Equus';

//echo "".$HTTP_POST_VARS['botname'] ."";
error_reporting(0);

/**
* Include the guts of the program.
*/
include "respond.php";




$numselects=0;


session_start();
$myuniqueid=session_id();
```

```
$botresponse=replybotname($HTTP_POST_VARS['input'],
                $myuniqueid,$HTTP_POST_VARS['botname']);


$str_response = "\n Equus: " . $botresponse->response . "\n";
if(!(isset($HTTP_POST_VARS['input']))){
         $str_response="\n Equus: ola, eu sou equus!\n";
$HTTP_POST_VARS['str_conversa'] = $str_response;
}else{
$str_conversa = " User: ".$HTTP_POST_VARS['input'].
                                    "\n".$str_response;
$HTTP_POST_VARS['str_conversa'] = $HTTP_POST_VARS['str_conversa']
                                                        .$str_conversa;
}



?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- saved from url=(0014)about:internet -->
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>::::EQUUS::::</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<style type="text/css">td img {display: block;}</style>

<script type="text/javascript">
   window.onload=function WindowLoad(event)
   {
    var objControl=document.forms["FrmTestScroll"].elements["conteudo"];
    objControl.scrollTop = objControl.scrollHeight;
   }
 </script>

</head>
<body bgcolor="#ffffff">
```

```
<form name="form1" method="post" action="index.php" id="FrmTestScroll">

<table border="0" cellpadding="0" cellspacing="0" width="800" align="center">

  <tr>
   <td><img src="spacer.gif" width="21" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="176" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="16" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="413" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="11" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="80" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="1" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="27" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="11" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="44" height="1" border="0" alt="" /></td>
   <td><img src="spacer.gif" width="1" height="1" border="0" alt="" /></td>
  </tr>

  <tr>
   <td colspan="10" align="center" valign="middle">
     <img src="titulo.jpg" border="0" alt="EQUUS"  align="middle"/>
   </td>
   <td> </td>
  </tr>
  <tr>
   <td colspan="8"> </td>
   <td rowspan="7" colspan="2"> </td>
   <td> </td>
  </tr>
  <tr>
   <td rowspan="2"> </td>
   <td rowspan="2"><img name="equus" src="equus.jpg" width="176"
                    height="217" border="0" id="equus" alt="" />
   </td>
   <td colspan="6"><img name="index_r3_c3" src="index_r3_c3.jpg"
                    width="548" height="47" border="0" id="index_r3_c3"
                    alt="" />
   </td>
```

```
   <td><img src="spacer.gif" width="1" height="47" border="0" alt="" /></td>
  </tr>
  <tr>
   <td rowspan="5"><img name="index_r4_c3" src="index_r4_c3.jpg" width="16"
                   height="373" border="0" id="index_r4_c3" alt="" />
   </td>
   <td rowspan="2" colspan="4" bgcolor="e3e3e3">
      <!-- TEXTAREA -->
      <textarea name="conteudo" rows="15" cols="60"  style="border: none;" >
                <? print "\n".$HTTP_POST_VARS['str_conversa'];  ?>
      </textarea>
   </td>
   <td rowspan="5"><img name="index_r4_c8" src="index_r4_c8.jpg" width="27"
                   height="373" border="0" id="index_r4_c8" alt="" />
   </td>
   <td><img src="spacer.gif" width="1" height="170" border="0" alt="" /></td>
  </tr>
  <tr>
   <td rowspan="5" colspan="2"> </td>
   <td><img src="spacer.gif" width="1" height="104" border="0" alt="" /></td>
  </tr>
  <tr>
   <td colspan="4"><img name="index_r6_c4" src="index_r6_c4.jpg" width="505"
                   height="27" border="0" id="index_r6_c4" alt="" />
   </td>
   <td><img src="spacer.gif" width="1" height="27" border="0" alt="" /></td>
  </tr>
  <tr>
   <td bgcolor="e3e3e3">
    <!--caixa de texto inserir mensagem -->
        <input type="hidden" name="<?=session_name()?>" value="<?=$uid?>">
<input type="hidden" name="botname" value="<?=
           $HTTP_POST_VARS['botname']?>">
        <input type="hidden" name="str_conversa" value="<?=
           $HTTP_POST_VARS['str_conversa']?>">
        <input type="text" name="input" size="60" border="0"
        style="border: none;">
   </td>
```

```
   <td rowspan="2"><img name="index_r7_c5" src="index_r7_c5.jpg"
                   width="11" height="72"
                   border="0" id="index_r7_c5" alt="" />
   </td>
   <td bgcolor="#E3E3E3">
   <!-- Botão input -->
     <input type="submit" name="Enviar" value="Enviar">
   </td>
   <td rowspan="2"><img name="index_r7_c7" src="index_r7_c7.jpg" width="1"
                    height="72" border="0" id="index_r7_c7" alt="" />
   </td>
   <td><img src="spacer.gif" width="1" height="26" border="0" alt="" /></td>
  </tr>
  <tr>
   <td><img name="index_r8_c4" src="index_r8_c4.jpg" width="413" height="46"
         border="0" id="index_r8_c4" alt="" />
   </td>
   <td><img name="index_r8_c6" src="index_r8_c6.jpg" width="80" height="46"
        border="0" id="index_r8_c6" alt="" />
   </td>
   <td><img src="spacer.gif" width="1" height="46" border="0" alt="" /></td>
  </tr>
  <tr>
   <td colspan="7"> </td>
   <td> </td>
   <td><img src="spacer.gif" width="1" height="90" border="0" alt="" /></td>
  </tr>
</table>

</form>

</body>
</html>
```

A.2 Interface FGCE

```
<table width="700" border="0" align="center">
  <form action="gera_aiml.php" name="entrada"  method="post">
  <tr>
    <td colspan="2"><div align="center"><strong>FERRAMENTA DE
        GERA&Ccedil;&Acirc;O DE CONTEUDO ESPEC&Iacute;FICO
        </strong></div></td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
    <td width="566"><strong>Texto de entrada</strong></td>
    <td width="124"> </td>
  </tr>
  <tr>
    <td colspan="2" align="center">

        <textarea name="texto" cols="100" rows="20"></textarea>    </td>
  </tr>
  <tr>
    <td> </td>
    <td><input type="submit" name="Gerar AIML"  title="Gerar AIML"
            value="Gerar AIML"/>
    </td>
  </tr>
 </form>
</table>
```

A.3 Processamento FGCE

```
<?

/***
* Autor: Eustáquio Cezar Pereira Filho
*
* ===Ferramente de geração de conhecimento automático==
*       ===          Universidade Federal de Goias         ===
 */
 function str2($a , $b){
   return stristr($a, $b);
 }
 //importanto os termos para o dicionário
$lines_dic = file("dic.txt");
 //importando
$texto = $_POST["texto"];

if($texto == ''){
  echo"<center><br><bR><br><b>É necessário fornecer um texto
        de entrada.</b><br><br><a href='index.php'>voltar</a></center>";
}else{


//enviar texto ao banco de dados dividindo em sentenças delimitadas por .

$texto_fragmentado = explode(".", $texto);




/*
* Imprimir os dois arrays de entrada
*/

for($n=0; $n < count($texto_fragmentado); $n++) {
echo " $n ".$texto_fragmentado[$n] ."<br>";
} // fim o for
```

```
echo"<bR><br><br><br><br>";

for($n=0; $n < count($lines_dic); $n++) {
echo " $n ".$lines_dic[$n] ." - ";
} // fim o for


/*
*  Efetua a busca dos termos técnicos na string de entrada
*/

$n_strings='';
$n_dic='';
$cont=0;
for($i=0; $i < count($texto_fragmentado); $i++) {

for($j=0; $j < count($lines_dic); $j++) {

     $temp1= $texto_fragmentado[$i];
   $temp2= $lines_dic[$j];
   $temp="";
   $temp = str2($temp1, $temp2);

if( $temp <> ""){
   $n_strings[$cont] =  $texto_fragmentado[$i];
   $n_dic[$cont] = $lines_dic[$j];
   $cont++;
}//fim if

} // fim for


} // fim for

/*
*  Gera estrutura AIML corresponde aos casamentos encontrados
*/
```

```
if($cont <> 0){
$str = "<?xml version='1.0' encoding='ISO-8859-1'?>
<aiml version='1.0'>";

$hora = date('i')."_FGCE.aiml";


for($n=0; $n < $cont; $n++) {
echo " $n ".$n_strings[$n] ." - <b>".$n_dic[$n]."</b><br>";
$str .= "\n                <category>
  <pattern> * ".$n_dic[$n]."  </pattern>
  <template> ".$n_strings[$n]."</template>
</category>\n";
}

$str.="\n        </aiml>";
// Abre ou cria o arquivo 01.aiml
// "a" representa que o arquivo é aberto para ser escrito
$fp = fopen("$hora", "a");

// Escreve "exemplo de escrita" no bloco1.txt
$escreve = fwrite($fp, $str);

// Fecha o arquivo
fclose($fp);
}else{
      echo"<br><br><br><center><b>Não foi encontrado nenhum padrão
             possível!</b></center>";
}

}//fim else
?>
```