UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CENTRO INTEGRADO DE NOVAS TECNOLOGIAS NA EDUCAÇÃO
ESPECIALIZAÇÃO EM INFORMÁTICA NA EDUCAÇÃO

ABEL CORRÊA

ROBÔ DE CONVERSAÇÃO APLICADO A EDUCAÇÃO A DISTÂNCIA COMO TUTOR INTELIGENTE

PORTO ALEGRE
2010

Abel Corrêa

ROBÔ DE CONVERSAÇÃO APLICADO A EDUCAÇÃO A DISTÂNCIA COMO TUTOR INTELIGENTE

Monografia apresentada à Universidade Federal do Rio Grande do Sul como requisito parcial para obtenção do título de Especialista em Informática na Educação.

Orientadora: Rosa Maria Vicari

Porto Alegre
2010

DEDICATÓRIA

Dedico este trabalho a minha mãe Nica
e à memória de minha avó Ertilia e uma das
professoras mais espetaculares que conheci,
minha tia Vera Lúcia.

AGRADECIMENTOS

Agradecendo em ordem cronológica acho que primeiramente ao meu colega de SEAD, Gerson Milan, pois foi ele que me indicou a participar do curso de especialização em Informática na Educação, também ao Prof. Dr. Sérgio Franco e Dr. Silvestre Novak pelo apoio juntamente com os meus colegas de trabalho, apoio este que se mostrou presente nos períodos onde eu precisava me retirar para participar das aulas presenciais ou realizar alguma tarefa, e através do projeto do edital onde este trabalho está sendo aplicado.

À todos os meus colegas pelo apoio e carinho, sem eles esse trabalho não seria possível. À Adriana, Isis, Sandra pela ajuda sempre presente nas questões da Educação, Pedagogia, trabalhos em grupo, conversas.

Agradeço também à Profª Liane Tarouco e toda equipe do CINTED, pela disponibilidade em me atender e fornecer a base de dados AIML da Profª Elektra, o chatterbot do CINTED, para que eu pudesse ter uma idéia inicial para este trabalho.

E finalmente a Profª Rosa Vicari por aceitar ser a minha orientadora, e por ter dado todo o apoio nesse sentido, através das palavras de incentivo, das correções e indicações, também à Michele Leondhardt, que apesar de não conhecê-la pessoalmente me ajudava enviando materiais, artigos e trabalhos anteriores para servir de referência.

À Deus que me inspirou e todos que de alguma forma colaboraram para o desenvolvimento do projeto, família, amigos, professores, colegas de aula e de trabalho, a vocês, meu sincero agradecimento.

“Qualquer tecnologia suficientemente
avançada é indistinguível da magia.”

Arthur C. Clarke

RESUMO

O presente trabalho tem por objetivo a análise de técnicas de inteligência artificial para aplicação na área da educação, mais precisamente na área de tutoria com o uso de tutores inteligentes na forma de robôs de conversação *(chatterbots)* afetivos com conhecimento em áreas específicas. Priorizamos *chatterbots* que estão baseados no papel do tutor presencial em cursos à distância e do tutor presencial em cursos presenciais e motivado pelo crescimento do uso e desenvolvimento das tecnologias inovadoras nos cursos EAD. O uso desta tecnologia visa, entre outras aplicações, evitar a evasão dos alunos nessa modalidade de ensino.

Palavras-chave: inteligência artificial, *chatterbots*, afetividade, educação a distância, tutor presencial

ABSTRACT

The present work aims at the analysis of artificial intelligence techniques for application in education, specifically in the area of tutoring with the use of intelligent tutors in the form of afective chatterbots with knowledges in specifical fields. We prioritize chatterbots based on the presencial tutor´s paper in learning distance courses and presencial tutor in presencial courses and motivated by the growing of use and development of innovative technologies at the learning distance courses. The use of this technology aims, among others applications, to prevent the escape of students in this teaching method.

Keywords: artificial intelligence, chatterbots, afectivity, learning distance, presencial tutors.

LISTA DE ABREVIATURAS E SIGLAS

AIML – *Artificial Intelligence Markup Language*
EAD – Educação a Distância
F – Falante
FAQ – *Frequently Asqued Question*
HTML – *HiperText Markup Language*
IA – Inteligência Artificial
iAIML – *Artificial Intelligence Markup Language* com Intenção
O – Ouvinte
OCC – *Ortony, Clore and Collins*
P – Proposição
PHP – *Hipertext PreProcessor*
S – Sintagma
SN – Sintagma Nominal
SV – Sintagma Verbal
XML – *eXtensible Markup Language*
XML – R – *eXtensible Markup Language – Reader*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A comunicação através da internet e as tecnologias computacionais abriram espaço para a Educação a Distância como nova e promissora modalidade de ensino, buscando atender áreas que antes não tinham acesso ao ensino superior, bem como pessoas que não tem condições de deslocamento até uma instituição de ensino.

É nessa modalidade de ensino que entram tecnologias como ambientes virtuais de aprendizagem, funcionando como salas de aula virtuais, dispondo de recursos síncronos e assíncronos, possibilitando a interação entre professor/tutor/aluno, e fazendo com que o conteúdo permaneça acessível, e o aluno seja o construtor do seu próprio conhecimento, tornando o professor o mediador desse processo auxiliado pelo tutor e outras tecnologias.

O tutor no ensino a distância vem para auxiliar o aluno, atuando diretamente com este, se relacionando com ele e acompanhando o andamento do aluno no processo de aprendizagem, mediando a interação do professor com o aluno, e do aluno com o material didático, tanto presencialmente como na comunicação a distância.

Sendo o tutor um papel fundamental na educação a distância, faz-se necessário o desenvolvimento de uma tecnologia de tutoria inteligente virtual, não com o fim de substituir o tutor real, mas sim de auxiliá-lo uma vez que um sistema de tutor inteligente acompanharia o aluno ininterruptamente.

É nesse contexto que aparecem os robôs de conversação, *chatterbots*, uma alternativa na área de inteligência artificial que pode ser utilizado para os mais diversos fins, na forma de FAQ (*Frequently Asked Questions*), lazer, tomada de decisões, educação.

Os robôs de conversação utilizam a linguagem natural para interagir com o usuário sendo assim um agente inteligente de conversação.

1.1 Objetivos da Pesquisa

O presente trabalho tem como objetivo geral o estudo e implementação de um *chatterbot* para aplicação na área da educação, com a possibilidade de servir como um tutor virtual inteligente de determinada disciplina, estando disponível permanentemente para o aluno, motivado pelo desenvolvimento constante de novas tecnologias aplicadas na educação e visando motivar ainda mais a procura por cursos na modalidade a distância.

Com base nisso, tem-se como objetivos específicos do trabalho:

- A análise do papel do tutor na educação a distância;
- O estudo aprofundado dos robôs de conversação e como estes podem auxiliar no processo de ensino – aprendizagem, analisando quais as características técnicas são aplicáveis para o desenvolvimento de um sistema tutor inteligente que interage com o aluno em língua natural.

1.2 Organização do Trabalho

O trabalho está organizado nos seguintes capítulos a saber, além desta introdução:

O capítulo 2 apresenta um breve histórico sobre os robôs de conversação contando sobre a criação do termo Inteligência Artificial, a questão levantada sobre Turing, bem como os primeiros *chatterbots* e as gerações de robôs de conversação apresentando as tecnologias utilizadas no desenvolvimento desses agentes e o papel das emoções.

Passa-se então ao capítulo 3 que tratará sobre o papel do tutor no ensino a distância, a relação deste com o aluno, e os modelos de aluno, com o fim de analisar qual o modelo de aluno que o robô de conversação é melhor aplicável, bem como analisar quais aspectos do tutor devem ser passados para o *chatterbot* para que este se torne uma ferramenta inovadora no ensino.

No quarto capítulo é mostrada a linguagem AIML (*Artificial Intelligence Markup Language*), a linguagem de marcação de inteligência artificial que pode ser utilizada para o desenvolvimento da base de conhecimento do *chatterbot*, e logo em seguida é mostrado, no capítulo 5, a arquitetura que se pretende utilizar no *chatterbot* a ser aplicado como tutor inteligente.

O sexto capítulo conclui o trabalho apresentado os aspectos positivos e negativos do *chatterbot* aplicado a educação a distância, analisando o seu papel no processo da construção do conhecimento do aluno, bem como a sua atuação como um sistema tutor inteligente, também é apresentado a proposta de trabalhos futuros.

2. CHATTERBOTS E INTELIGÊNCIA ARTIFICIAL

Foi em 1950 que o matemático Alan Turing implantou a dúvida sobre as máquinas realmente poderem pensar, e desde então isso vem sido posto a prova através do Teste de Turing, um teste que consiste em um participante que conversará com uma máquina e outro participante humano, e deverá indicar qual participante é a máquina e qual é o humano.

O Teste de Turing surgiu do jogo da imitação que consistia em três participantes, sendo um homem, uma mulher e um interrogador que ficaria separado dos outros dois, o homem e a mulher eram “etiquetados” como X e Y e o interrogador deveria então descobrir quem era o homem e quem era a mulher, enquanto os objetivos dos participantes era, um deles ajudar o interrogador e o outro atrapalhar.

Deste teste, surgiu a ideia de substituir um dos participantes por uma máquina, então o interrogador deveria descobrir qual dos participantes era a máquina e qual dos participantes era humano. A máquina que conseguisse enganar o interrogador teria então passado no teste de Turing.

Segundo RUSSEL e NORVIG (2004) um computador para ter a capacidade de conversar com um ser humano deveria ter as seguintes capacidades:

- Processamento de linguagem natural o que permitiria que o computador entendesse uma linguagem ao invés de apenas símbolos matemáticos;
- Representação de conhecimento para armazenar o que sabe ou ouve;
- Raciocínio Automatizado para usar as informações que armazenou e responder perguntas e tirar conclusões novas;
- Aprendizado de máquina para adaptar-se a novas circunstâncias, detectar e extrapolar padrões

Mas para fazer com que uma máquina seja capaz de uma comunicação que o faça indistinguível de humanos, ou mais próximo disso; é necessário também,

conhecer a forma como pensamos e agimos, e estudar o campo interdisciplinar da ciência cognitiva, que tenta fazer uma mescla entre a inteligência artificial e técnicas experimentais da psicologia.

Para ROTHERMEL (2007), há três gerações de robôs de conversação:

- A primeira geração deu origem ao *chatterbot* Eliza, que agia como uma psicanalista e fazia com que o usuário falasse sobre seus problemas, era baseado em regras gramaticais e não armazenava conversas anteriores.
- A segunda geração era baseada em regras de produção e redes neurais, deu origem ao robô JULIA de Michael Mauldin(1994).
- A terceira geração e mais recente é baseada em AIML[1] para construção da base de conhecimento, o projeto mais conhecido é o A.L.I.C.E. (WALLACE, 2001)

AIML é uma linguagem de marcação, semelhante a XML e necessita de um interpretador que lê uma entrada textual e resgata dentro de *tags*[2] o que responder ao usuário, utiliza um arquivo com extensão AIML como base de conhecimento do *chatterbot*. A AIML, sendo derivada do XML[3], é baseada em categorias, e cada uma destas representa um padrão de entrada (WALLACE, 2001).

Por outro lado temos uma forma mais complexa e não menos eficaz que pode, também, gerar resultados, essa forma começa do princípio da comunicação como ação, onde o ponto inicial é, por parte do sujeito falante, a intenção de falar, a partir daí, acontece a geração de uma expressão que seja adequada a um ouvinte específico, e por parte do sujeito ouvinte, a percepção da frase, que envolve a análise, eliminação da ambigüidade e incorporação da frase.

A análise mencionada no parágrafo anterior é dividida em três partes que são: a interpretação ou análise sintática, interpretação ou análise semântica e interpretação ou análise pragmática(RUSSEL et NORVIG, 2004).

Esse método faz uso de regras de gramática, funciona como se estivéssemos ensinando o computador a falar de uma forma gramaticalmente correta, imagina-se um pai ao ensinar a criança a falar, no entanto, ensinando

---

1 *Artificial Intelligence Markup Language*

2 *Tags* são marcações utilizadas em hipertextos com o intuito de atribuir certas propriedades ou formato ao texto.

3 eXtensible Markup Language – linguagem de marcação para necessidades especiais.

gramática. Para um ser humano em estágio inicial de desenvolvimento pode não ser a forma mais eficaz, mas para a máquina é.

2.1 Processamento de Linguagem Natural

Neste ponto vamos identificar alguns conceitos como linguagem formal que é um conjunto de cadeias, cada cadeia é uma concatenação de símbolos terminais, ora chamados palavras.

Linguagem natural são usadas por uma comunidade de falantes, como o português, inglês, chinês.

Uma gramática é um conjunto finito de regras que especificam uma linguagem, linguagens formais têm uma gramática oficial, já as linguagens naturais não têm, mas os linguistas estão sempre investigando e inserindo suas propriedades em uma gramática.

Também temos que todas as linguagens possuem um significado ou uma semântica, por exemplo, a linguagem aritmética onde X e Y são expressões, então X + Y é uma expressão válida e sua semântica é soma de X e Y.

Nas linguagens naturais também devemos compreender a pragmática de uma cadeia, ou seja, qual seu significado em um dado momento, por exemplo, os dois significados da seguinte frase: "O João tá morto!"

Dependendo da situação podemos dizer que o indivíduo de nome João faleceu ou então ele está apresentando fadiga devido ao seu desempenho laboral.

A maioria dos formalismos de regras gramaticais baseia-se na ideia de estrutura sintagmática (RUSSEL, S.; NORVIG, P., 2004), onde as cadeias baseiam-se em subcadeias chamadas sintagmas, Sintagma Nominal e Sintagma Verbal para formar a Sentença.

Essas categorias são chamadas de símbolos não - terminais, que são determinadas usando regras de reescrita, através de uma notação denominada de BNF (*Backus - Naur Form*) Forma de Backus-Naur.

$$S \rightarrow SN\ SV$$

A regra acima define que toda sentença S consiste de qualquer sintagma nominal SN seguido por qualquer sintagma verbal SV. Ou seja, na sentença: “O

Wumpus está morto!”, temos o sintagma nominal SN (O WUMPUS) e o sintagma verbal SV (ESTÁ MORTO!), formando então a sentença S.

Também de acordo com RUSSEL e NORVIG (2004) um episódio de comunicação é composto por sete processos:

1. **Intenção:** onde um falante F decide que existe uma proposição P que vale a pena informar a um ouvinte O;
2. **Geração:** o planejamento sobre como a proposição P vai ser expressa para que o ouvinte O seja capaz de entender;
3. **Síntese:** realização física da comunicação, seja pronunciando a proposição ou escrevendo;
4. **Percepção:** o ouvinte O percebe a realização física da proposição P;
5. **Análise:** dedução dos possíveis significados da proposição P pelo ouvinte O, esse processo é dividido em três sub-processos a saber;
   a. Análise Sintática: construção de uma árvore de análise para a cadeira de entrada;
   b. Interpretação Semântica: extração do significado;
   c. Interpretação Pragmática: que leva em conta que uma mesma palavra pode ter vários significados, tem por função escolher a interpretação mais adequada a situação.
6. **Eliminação da ambigüidade:** quando o ouvinte O deduz que o falante F pretendia transmitir a proposição P_i, onde no caso ideal $P_i = P$. Aqui o ouvinte interpreta o que o ouvinte provavelmente queria transmitir.
7. **Incorporação:** o ouvinte decide acreditar, ou não, em P_i.

A Figura 1 mostra os setes processos que compõem a comunicação, segundo RUSSEL e NORVIG.

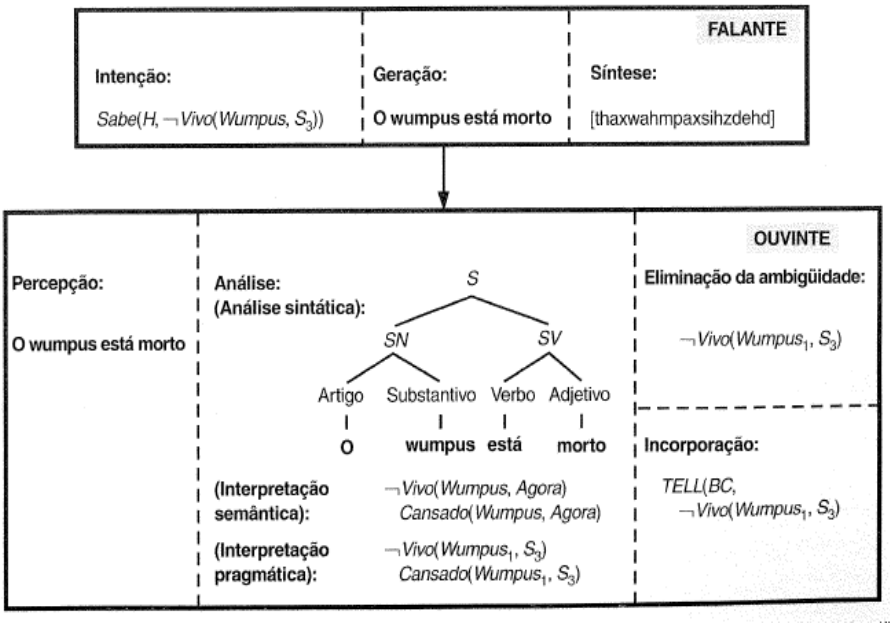

Figura 1 Sete etapas do processo de comunicação
Fonte: (RUSSEL, NORVIG. 2004)

2.2 Computação Afetiva

Conforme PICCARD (1995) as emoções influenciam na tomada de decisões, com base nisso, a computação afetiva consiste no reconhecimento de emoções humanas pelo computador, bem como a capacidade do computador de expressar tais emoções e tomar decisões baseado nelas.

Em um sistema tutor inteligente ou um *chatterbot*, as emoções poderiam ser expressas em forma textual de acordo com as palavras utilizadas dando um tom de hostil ou amigável a conversa, dando inclusive uma certa personalidade a máquina, uma máquina incapaz de perceber ou expressar emoções provavelmente não passaria no teste de Turing.

Outra forma de expressão das emoções são os avatares[4], que podem contribuir para que o *chatterbot* fique mais amigável e demonstre visualmente a sua personalidade.

[4] Em informática, avatar é a representação visual de um utilizador em realidade virtual. De acordo com a tecnologia, pode variar desde um sofisticado modelo 3D até uma simples imagem. São normalmente pequenos, de tamanhos variados mas deixando espaço livre para a função principal do site, programa ou jogo que se está a usar. (Wikipedia: http://pt.wikipedia.org/wiki/Avatar_(realidade_virtual))

Com base na importância das emoções nos robôs de conversação e a influência que se tem sobre o aprendizado, sintetiza-se a seguir três teoria sobre as emoções que podem ser aplicadas em modelos computacionais.

2.2.1 Modelo OCC – Ortony, Clore and Collins

O modelo OCC (Ortony, Clore and Collins), desenvolvido com o objetivo de se implementar em um computador, se baseia na teoria cognitivista das emoções.

Para os autores as emoções são reações com valência e condições desencadeantes, no caso, não existem eventos que causam emoções, mas, emoções podem ocorrer como um resultado de como os eventos são percebidos e interpretados. (FROZZA, 2009).

As emoções são divididas em três categorias de acordo com o estímulo: consequência de eventos, ações de agentes e aspectos de objetos, tendo identificadas 22 emoções que são organizadas em pares; cada par, formado por uma emoção com valência positiva e a sua contrária com valência negativa.

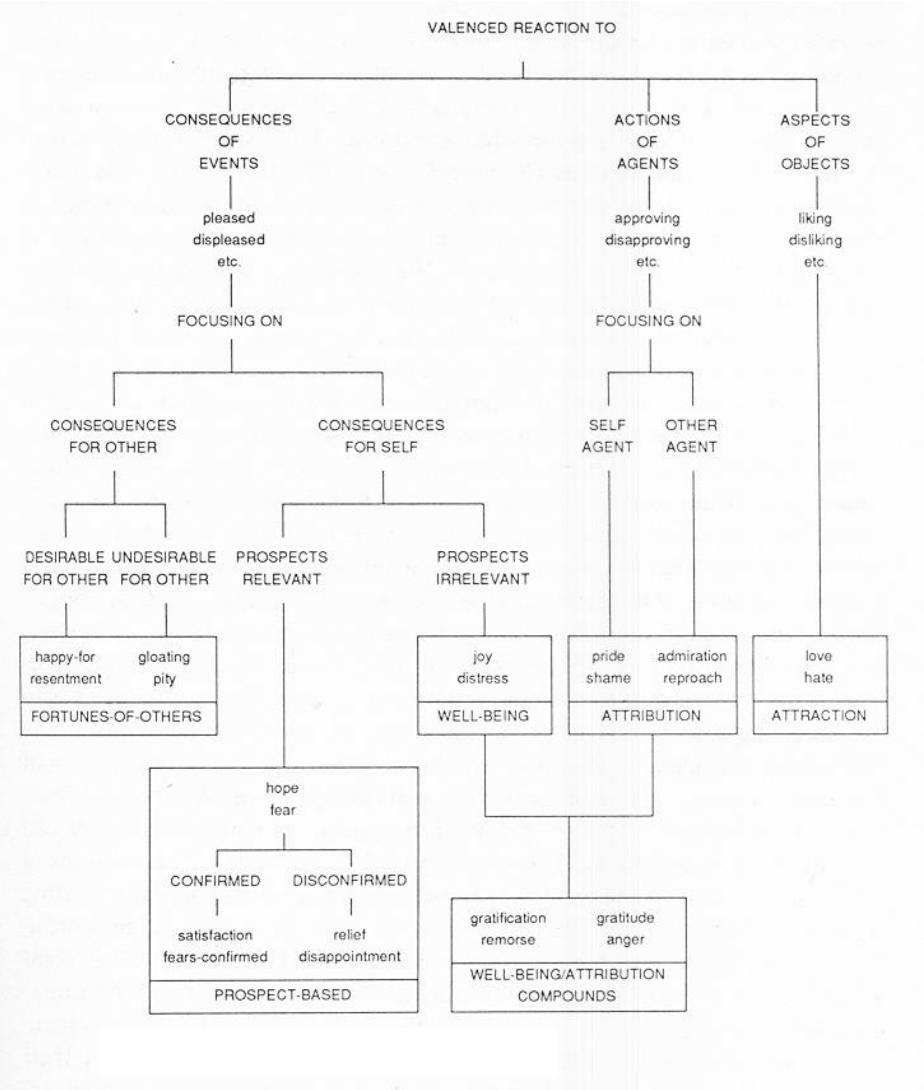

Figura 2 Estrutura das Emoções de acordo com o Modelo OCC
Fonte: (ORTONY, CLORE, COLLINS. 1999, p. 19)

2.2.2 Teoria de Roseman sobre as Emoções

A teoria de Roseman é uma abordagem básica da Teoria da Avaliação das Emoções, seu modelo é derivado de cinco dimensões cognitivas que determinam quando uma emoção chega e qual ela é.

Por sua estrutura simples que pode ser traduzida rapidamente em regras que definem exatamente quais avaliações provocam quais emoções, esse modelo foi recebido positivamente nos círculos de IA (RUEBENSTRUNK, 1998).

A combinação das cinco dimensões do modelo original de Roseman equivaliam a 13 emoções, o autor modificou seu modelo diversas vezes, o modelo original foi usado na implementação de um modelo computacional.

2.2.3 A Teoria de Frijda sobre as Emoções

Na teoria de Frijda as emoções surgem dos problemas para a realização de metas, que são chamados de “preocupações”.

> *The center of Frijda's theory is the term concern. A concern is the disposition of a system to prefer certain states of the environment and of the own organism over the absence of such conditions. Concerns produce goals and preferences for a system. If the system has problems to realize these concerns, emotions develop.*
>
> *The strength of such an emotion is determined essentially by the strength of the relevant concern(s).* (RUEBENSTRUNK 1998)

Segundo o autor, a força de uma emoção é determinada pela força da preocupação, dessa forma, havendo uma solução para uma preocupação, o alcance de uma meta, se têm uma emoção positiva, havendo um problema na realização de uma preocupação, se tem uma emoção negativa.

Qualquer situação, positiva ou negativa, ocorrida com uma preocupação é chamada, por Frijda, de tendência de ação.

A tendência de ação é ligada com o estado emocional, servindo como um dispositivo para o que é chamado de realização da preocupação.

O modelo computacional ACRES é baseado nesta teoria.

3. O TUTOR

Neste capítulo é apresentada a definição de tutoria, tipos de tutoria, bem como apresentado o perfil do tutor e do aluno, com o fim de fundamentar um modelo de tutor para ser aplicado em um modelo computacional de robô de conversação servindo assim, como um tutor inteligente.

Para PAN (2005) a função do tutor na educação a distância é uma necessidade, visto que ele tem o papel de prestar assistência ao aluno, sendo tanto o tutor como o professor, responsáveis pelo bom ensino, e colaboradores do processo de ensino – aprendizagem. O autor salienta que o tutor é uma ponte móvel entre o aluno, o curso e o professor.

A educação a distância acontece com alunos e professores separados fisicamente, interagindo através das tecnologias de informação e comunicação, e neste paradigma que entra o tutor como uma presença indispensável, onde em muitos modelos é o responsável pelo contato presencial com os alunos (VEDOVE, 2007), e interagindo com estes nos bate-papos, fóruns e recursos dos ambientes virtuais de aprendizagem, dando apoio ao professor.

É visível o papel do tutor, segundo os autores, como um mediador entre o relacionamento professor – conteúdo – aluno.

Quanto ao perfil do tutor, ARGÜIS ET AL. (2002), nos remete a três qualidade essenciais: o ser, o saber e o saber fazer.

> Encontramos escritas as qualidades que um tutor precisa ter. Fizemos uma sondagem sobre alunos de ensino médio e observamos características comuns entre os autores Baihy, Artigot, González Simancas, Río Rovira, Benavent, etc. Roman e Pastor fazem três distinções:
>
> - As qualidades humanas (o ser do tutor). A empatia, a maturidade intelectual – volitiva e afetiva, a sociabilidade, a responsabilidade e a capacidade de aceitação.

> - As qualidades científicas (o saber). Conhecimento da maneira de ser do aluno, conhecimento dos elementos pedagógicos para conhecer e ajudar o aluno.
> - As qualidades técnicas (o saber fazer). Trabalhar com eficácia e em equipe, participando de projetos e programas definidos em comum acordo para a formação do aluno. (ARGÜIS ET AL., 2002. pg. 17)

Pode-se fazer a seguinte relação entre as qualidades do tutor e um sistema de *chatterbot*:

- O ser do tutor, se refere a questões afetivas, o que pode ser implementado com os recursos de computação afetiva.
- O saber, se refere ao conhecimento do tutor na área que se propõe a auxiliar, no caso de um *chatterbot* se refere a base de conhecimento do mesmo.
- O saber fazer, se refere as qualidades do tutor de interagir com o aluno, conteúdo e professor, bem como participar de projetos, e participar da formação do aluno (ARGÜIS ET AL, 2002), no entanto um *chatterbot* não tem a capacidade de participar de livre e espontânea vontade de um projeto, ou trabalhar em equipe, logo, essa qualidade pode se traduzir na percepção do sistema tutor inteligente de captar o andamento do aluno e tomar decisões participando do processo de ensino – aprendizagem.

2.3 Funções do Tutor:

A característica do tutor é a de mediador, entre professor / aluno / conteúdo, tanto presencialmente ou à distância, e diminuir a sensação de que o aluno está sozinho.

Desta forma, MUNHOZ (2003) destaca que o tutor pode sugerir ao aluno modificar a forma com que está estudando o material, criar situações diferenciadas para que o aluno possa compreender melhor o que está sendo apresentado de acordo com a personalidade deste. O tutor também deve suprir a ausência do professor e possibilitar o diálogo.

> Outra característica desejável é o seu envolvimento no entorno social dos estudantes [...]. O conhecimento deste entorno social facilita sobremaneira o trabalho de tutoria.
>
> Assim posto, o sistema de tutoria deve ser dotado de uma flexibilidade que permita ao programa de cada unidade didática ser adaptado a cada um dos

> alunos de acordo com a sua personalidade e utilizando os conhecimentos anteriores adquiridos em sua prática profissional e história de vida. Assim, o tutor consciente das dificuldades de cada estudante, poderá determinar novos rumos para a aprendizagem estabelecidos em contato com os professores especialistas. (MUNHOZ, 2003)

O trabalho principal do tutor não é o de transmitir conteúdo ou construir o material, isso compete ao professor, o tutor, deve mediar o processo de aprendizagem, e ser capaz de incentivar o aluno, fazê-lo ser autônomo na construção do conhecimento.

Munhoz (2003) destaca cinco funções dos tutores:

- Motivar e despertar o interesse dos alunos no desenvolvimento das práticas;
- Orientar o aluno nas dificuldades, sendo intermediário entre este e o professor;
- Indicar novos materiais, artigos, textos, endereços da internet, sobre o conteúdo, possibilitando que os alunos concluam as tarefas;
- Avaliar o progresso do aluno;
- Atuar junto de outros tutores e professores, ampliando a visão da avaliação individual dos estudantes e identificar falhas dos materiais e tarefas.

2.4 O Aluno de EAD

Conforme TAROUCO ET AL (2003), o sucesso do EAD está no posicionamento do aluno com relação ao processo educacional. O aluno de EAD é mais ativo que o aluno tradicional, interagindo e colaborando para o seu próprio desenvolvimento educacional e de seus colegas, através das tecnologias e mediado pelo professor e tutor.

A autora ainda cita o privilégio do aluno na relação com a tecnologia (TAROUCO, 2003, pg. 9 APUD MORAN, 2000), visto a sua intimidade e facilidade para lidar com os recursos tecnológicos, produção de materiais e trabalho em grupo.

Com isso a autora mostra dois perfis de alunos, o aluno tradicional e o aluno aprendiz, sendo o último um agente no seu processo de ensino aprendizagem,

construindo o seu conhecimento através da exploração de recursos, criatividade, interação, socialização e senso crítico.

Tabela 1: Aluno Tradicional x Aluno Aprendiz

Aluno Tradicional	**Aluno Aprendiz**
- Recebem passivamente as informações do professor a partir do livro – texto.	- Explora as possibilidades.
- Procura a “resposta certa”, segundo o método ensinado pelo professor.	- Inventa soluções alternativas.
- Participação individual, sem estabelecer relação de trocas entre os colegas e o professor.	- Colabora e coopera com o professor e com os colegas.
- Apresenta respostas prontas e memorizadas. (“decoreba”)	- Revisa seus pensamentos e apresenta melhor solução encontrada.
- Lê e responde a ficha de leitura cobrada pelo professor.	- Lê, critica, recria e reelabora textos.
- Avaliação: decora regras e/ou fórmulas. - Prepara-se somente para memorizar informações. - Repete o que o professor diz.	- Avaliação: busca novas respostas. - Procura reconstruir o que aprendeu. - Reconhece suas dificuldades e/ou falhas e procura superá-las; - Interage com o professor; às vezes superando-o.

Adaptado de TAROUCO, 2003

Um sistema tutor inteligente ideal deve atender as expectativas de um aluno aprendiz, devendo auxiliá-lo e assimilar informações do aluno, o que resultaria num acréscimo a sua base de conhecimento e como conseqüência uma melhora em seu próprio desempenho, mas para isso o processo de Incorporação desse novo conhecimento deve ser eficaz.

Outra características que um sistema tutor inteligente deve possuir é a de captar e analisar emoções, expressar emoções de acordo com o contexto da conversação e descobrir o perfil do aluno para guiá-lo no processo de aprendizagem.

4. A.I.M.L – ARTIFICIAL INTELLIGENCE MARKUP LANGUAGE

AIML é uma linguagem de marcação utilizada para o desenvolvimento de robôs de conversação, baseado na técnica de casamento de padrões.

Derivado de XML (*eXtensable Markup Language*), a AIML foi desenvolvida pela *Alicebot Free Software Comunnity* e Richard S. Wallace (WALLACE, 2001) entre 1995 e 2000.

O AIML descreve uma classe de objetos conhecido como objetos AIML, esses objetos são interpretados por um programa de computador que descreve o comportamento do robô de conversação.

As unidades básicas da linguagem são: tópico (*topic*) e categoria (*category*).

Categoria é a unidade básica que descreve um padrão de entrada e um padrão de resposta, conhecido respectivamente como *pattern* e *template*.

O padrão de linguagem é simples consistindo de palavras, números, espaços e curingas (“_” e “*”), esse padrão não é variável, devendo conter apenas um espaço entre as palavras, e os curingas servindo como palavras.

A versão atual da linguagem AIML é 1.0.1 e suporta vários curingas em um padrão de entrada.

```
<aiml version="1.0">
<category>
    <pattern>OI</pattern>
    <template>
           <random>
                 <li>Oi como e o seu nome?</li>
                 <li>Oi, meu nome e Robo.</li>
                 <li>Oi</li>
           </random>
    </template>
</category>
</aiml>
```

Figura 3: Padrão de uso de AIML para criação da base de dados de um *chatterbot*.

Na Figura 3, vê-se um código de uso básico da linguagem AIML, inicialmente é descrito a versão da linguagem, e então é descrita uma categoria com um padrão de entrada e um padrão de saída com várias saídas aleatórias possíveis.

Nota-se o uso de *tags*, na linguagem AIML, tal como em HTML[5], também, no exemplo, há uma *tag* chamada *pattern* que contém o padrão de entrada, e a *tag template* que contém o padrão de resposta do robô de conversação. No exemplo da Figura 3, dentro do padrão de saída, há a *tag random*, sua utilização informa ao interpretador que ele deve escolher uma resposta aleatória dentre as possíveis respostas que são separadas com a *tag li*.

É visível, no exemplo mostrado, a semelhança entre a linguagem AIML e outras linguagens de marcação, como HTML, XML, da qual é derivada, e outras.

Uma maneira de simular uma memória na conversação é utilizar a *tag that*, que faz uma referência a declaração anterior do *chatterbot*. (LEONHARDT, 2003) Ela pode ser utilizada, também, no caso de se querer passar a idéia de resposta à uma pergunta do robô de conversação.

Pode-se destacar o uso da *tag that* através do exemplo da Figura 4.

[5] HTML – HyperText Markup Language – Linguagem de Marcação de Hipertexto

```
<category>
    <pattern>REDE</pattern>
    <template>
          Achei que eu tinha problemas com lingüística, mas
você...<br>
          Sabes que a palavra rede pode dizer respeito a
     muitas coisas diferentes!<br>
          Mas sobre que tipo de redes você gostaria de saber?
    </template>
</category>
<category>
     <pattern>DE COMPUTADOR</pattern>
     <that>Mas sobre que tipo de redes você gostaria de
saber</that>
     <template><srai>REDE DE COMPUTADOR</srai></template>
</category>
```

Figura 4: Exemplo de código AIML demonstrando uso da *tag that*
Fonte: (LEONHARDT, 2003)

> "Neste caso a categoria "DE COMPUTADOR" é ativada somente quando o aluno digita esta frase respondendo a pergunta do *chatterbot*: "Mas sobre que tipo de redes você gostaria de saber". Este recurso enriquece a conversação, simulando uma memória da conversa apesar de ainda limitada à última sentença." (LEONHARDT, 2003).

Quando do uso da *tag that* em uma categoria, a dita categoria só será ativada se o texto da conversa anterior contiver o conteúdo referenciado na *tag that*. Esse é um recurso opcional para manter o contexto da conversação, e pode ser utilizado no caso do *chatterbot* pedir ou perguntar algo, por exemplo.

Há também a possibilidade de utilizar a *tag topic* para dividir o conhecimento do robô em tópicos, sendo assim um tópico poderá conter várias categorias sobre um determinado assunto.

Isso é bastante usado quando um mesmo padrão de entrada pode ser compartilhado por mais de um tópico, expressando que os padrões de entrada equivalentes podem ser utilizados em contextos diferentes.

A *tag topic* é inserida antes da *tag category*, podendo ser incluídas dentro de um *topic* várias *tags category*. Após isso, deve-se definir qual tópico está sendo utilizado através da *tag set*, como exemplificado na Figura 5.

A Figura 5 também apresenta a *tag think*, que em AIML serve para que o interpretador processe alguma informação sem retornar nada ao usuário do *chatterbot*, essa *tag* é bastante utilizada quando se faz necessário dimensionar

variáveis para uso posterior.

Utilizar tópicos é uma maneira de diferenciar o tema da conversação e para casos onde é necessário utilizar o mesmo padrão de entrada e retornar saídas diferentes.

Há também uma *tag* que pode ser utilizada para retornar o tópico corrente, a *tag get*, conforme utilizada na Figura 5, retorna o tópico atual.

4.1 Recursão com AIML

Com AIML é possível efetuar recursão com a utilização da *tag srai*. Conforme WALLACE (2009), não há acordo entre o significado da sigla, mas pode ter vários devido às diversas formas de uso do operador.

> “AIML implements recursion with the <srai> operator. No agreement exists about the meaning of the acronym. The"A.I." stands for artificial intelligence, but"S.R." may mean"stimulus-response,""syntactic rewrite,""symbolic reduction,""simple recursion," or"synonym resolution." The disagreement over the acronym reflects the variety of applications for <srai> in AIML.” (WALLACE, 2009).

```
<topic name="CHATTERBOT">
      <category>
            <pattern>O QUE E</pattern>
            <template>E um robo de conversacao, baseado em tecnicas de
                  inteligencia artificial.
            </template>
      </category>
</topic>

<topic name="IA">
      <category>
            <pattern>O QUE E</pattern>
            <template>E uma disciplina da ciencia da computacao
                  utilizada para simular a inteligencia humana em
                  computadores.
            </template>
      </category>
</topic>

<category>
      <pattern>VAMOS FALAR SOBRE CHATTERBOT</pattern>
      <template>
            Ta bom.
            <think><set name="topic">chatterbot</set></think>
      </template>
</category>

<category>
      <pattern>VAMOS FALAR SOBRE INTELIGENCIA ARTIFICIAL</pattern>
      <template>
            Ta bom.
            <think><set name="topic">ia</set></think>
      </template>
</category>

<category>
      <pattern>SOBRE O QUE ESTAMOS FALANDO MESMO</pattern>
      <template><get name="topic"/></template>
</category>
```

Figura 5: Demonstração de uso de tópicos de conversação com AIML

Exemplificando os usos da *tag srai*, conforme WALLACE, temos redução simbólica (*symbolic reduction*), dividir e conquistar (*divide and conquer*), sinônimos (*synonyms*), correções gramaticais ou ortográficas (*spelling or Grammar corrections*), palavras-chave (*keywords*), condicionais (*conditionals*).

Também, conforme o autor há certo perigo quanto ao uso da recursão, pois permite ao *botmaster* criar recursão (*loops*) infinita.

4.1.1 Redução Simbólica – *Symbolic Reduction*

Redução Simbólica é o nome dado ao processo de reduzir as diversas formas de se comunicar uma sentença em uma sentença mais simples. Pode-se exemplificar com as maneiras de se fazer a seguinte pergunta:

_Você sabe o que é um robô de conversação?
_O que é um robô de conversação?
_Me diz o que é um robô de conversação?

Pode-se afirmar que as três perguntas têm a mesma resposta, então ao invés de ser digitado um padrão de saída, ou *template*, para cada padrão de entrada descrito acima, pode-se utilizar a *tag srai* para direcionar para a resposta padrão.

```
<category>
            <pattern>O QUE E CHATTERBOT</pattern>
            <template>E um robo de conversacao, baseado em tecnicas de
                  inteligencia artificial.
            </template>
</category>
<category>
      <pattern>O QUE E UM ROBO DE CONVERSACAO</pattern>
      <template><srai>O QUE E CHATTERBOT</srai></template>
</category>
<category>
      <pattern>VOCE SABE ALGO SOBRE CHATTERBOT</pattern>
      <template><srai>O QUE E CHATTERBOT</srai></template>
</category>
```

Figura 6 *Tag srai* sendo utilizada para o propósito de redução simbólica

4.1.2 Dividir e Conquistar – *Divide and Conquer*

Ao contrário da técnica de redução simbólica, na técnica de dividir e conquistar há um padrão de entrada que pode possuir várias saídas possíveis, então o padrão de entrada é dividido em sentenças, cada uma com um padrão de saída que é concatenado e mostrado ao usuário.

Também é possível utilizar curingas para essa técnica, assim as diversas formas de se dizer um enunciado, podem ser previstas e tratadas.

Ao usar curingas pode-se pegar o valor do curinga através da *tag star* ou

quando se está utilizando recursão, pode-se juntar a recursão e o valor do curinga com a *tag sr*.

```
<category>
            <pattern>MEU NOME E *</pattern>
            <template>Oi <star/></template>
</category>
```

Figura 7: Capturando o valor de um curinga

```
<category>
            <pattern>O QUE E CHATTERBOT</pattern>
            <template>E um robo de conversacao, baseado em tecnicas de
                        inteligencia artificial.
            </template>
</category>

<category>
      <pattern>ENTAO E UM ROBO *</pattern>
      <template><srai>ROBO</srai><sr/></template>
</category>

<category>
      <pattern>ROBO</pattern>
      <template>Sim, mas um robo de conversacao. Nao um robo
            comum.
      </template>
</category>

<category>
      <pattern>BASEADO EM IA</pattern>
      <template>Utiliza tecnicas de inteligencia artificial para fazer
            seu trabalho.
      </template>
</category>

<category>
      <pattern>MAS UTILIZA QUE TECNICAS</pattern>
      <template>Utiliza tecnicas de Inteligencia Artificial.</template>
</category>
```

Figura 8: Exemplo de Recursão com a Técnica Dividir e Conquistar

Na Figura 8, pode-se visualizar que a *tag srai*, na segunda categoria irá redirecionar para o padrão de entrada ROBO e para outro padrão de entrada de acordo com o que foi digitado no curinga, concatenando as duas saídas em uma só. A *tag sr*, que é responsável por redirecionar ao que foi digitado pelo usuário no curinga, segundo WALLACE (2009) a *tag sr* é similar ao seguinte: <sr/> = <srai><star/></srai>.

Pode-se exemplificar a conversação da Figura 8 da seguinte maneira:

Usuário: _Entao é um robo, baseado em IA?
Chatterbot: _Sim, mas um robô de conversação, não um robô comum. Utiliza técnicas de Inteligência Artificial para fazer seu trabalho.

4.1.3 Sinônimos – Synonyms

A versão 1.0.1 da linguagem AIML não aceita mais que um padrão de entrada em uma categoria (WALLACE, 2009), quanto a vários padrões de saída ou *templates* podem ser utilizados com a *tag random* e a *tag li como* já exemplificado na Figura 3. No entanto, é possível utilizar a recursividade de *srai* para que vários padrões de entrada sejam redirecionados para um único padrão de saída.

```
<category>
            <pattern>CELULAR</pattern>
            <template>Telefone celular (portugues brasileiro) ou
                  telemovel (portugues europeu) e um aparelho de
                  comunicacao por ondas electromagneticas que permite a
                  transmissao bidireccional de voz e dados utilizaveis
                  em uma area geografica que se encontra dividida em
                  celulas (de onde provem a nomenclatura celular), cada
                  uma delas servida por um transmissor/receptor. A
                  invencao do telefone celular ocorreu em 1947 pelo
                  laboratorio Bell, nos EUA.
            </template>
</category>

<category>
            <pattern>TELEMOVEL</pattern>
            <template><srai>CELULAR</srai></template>
</category>

<category>
            <pattern>TELEFONE CELULAR</pattern>
            <template><srai>CELULAR</srai></template>
</category>

<category>
            <pattern>TELEFONE MOVEL</pattern>
            <template><srai>CELULAR</srai></template>
</category>
```

Figura 9: Exemplo da tag srai utilizada para sinônimos

4.1.4 Correções Gramaticais ou Ortográficas – *Spelling or Grammar Corrections*

Aqui é demonstrado o uso da *tag srai* de forma que o *chatterbot* corrija o usuário, supondo que ele esteja digitando algo errado, então ele o corrige, e logo, redireciona para o padrão correto.

```
<category>
            <pattern>DO YOU SPIKE ENGLISH</pattern>
            <template>Voce quer dizer: SPEAK.
                  <srai>DO YOU SPEAK ENGLISH</srai>
            </template>
</category>

<category>
            <pattern>DO YOU SPEAK ENGLISH</pattern>
            <template>YES, I DO.</template>
</category>
```

Figura 10: Exemplo de uso da tag srai para correção ortográfica

Conforme Wallace (2009), o robô age como um tutor de linguagem.

4.1.5 Palavras Chave – *Keywords*

É a forma de escrever um padrão de saída que será ativado quando forem digitadas certas palavras chave.

A Figura 11 mostra *ipsis literis*, o exemplo citado por WALLACE (2009), sobre o uso da *tag srai* para redirecionar palavras chave a um mesmo padrão de saída, onde inicialmente, no exemplo, há a palavra chave com uma saída genérica e então nas categorias seguintes a palavra chave antecedida por um curinga, logo após, precedida por um curinga e então a palavra chave entre curingas, e em todas as categorias, o resultado é que ela redireciona para o mesmo padrão de saída.

```
<category>
      <pattern>MOTHER</pattern>
      <template> Tell me more about your family. </template>
</category>

<category>
      <pattern>_ MOTHER</pattern>
      <template><srai>MOTHER</srai></template>
</category>

<category>
      <pattern>MOTHER _</pattern>
      <template><srai>MOTHER</srai></template>
</category>

<category>
      <pattern>_ MOTHER *</pattern>
      <template><srai>MOTHER</srai></template>
</category>
```

Figura 11: Uso da tag srai para o redirecionamento através de palavras chave.
Fonte: (WALLACE, 2009)

4.1.6 Condicionais – *Conditionals*

O uso da *tag srai* juntamente com o uso de variáveis permite que o *botmaster* trabalhe com a idéia de condições. Também existe uma *tag* mais apropriada para isto, mas há a possibilidade de utilizar a recursão de forma que trabalhe como condicional.

```
<category>
      <pattern>QUEM E ELE</pattern>
      <template><srai>ELE E <get name="he"/></srai></template>
</category>

<category>
      <pattern>VAMOS FALAR DO *</pattern>
      <template>
            Ta bom.
            <think><set name="he"><star/></set></think>
      </template>
</category>

<category>
      <pattern>ELE E TURING</pattern>
      <template>Turing foi um grande matematico.</template>
</category>

<category>
      <pattern>ELE E DESCONHECIDO</pattern>
      <template>Nao sei quem e.</template>
</category>
```

Figura 12: Usando condições com *srai*.

4.2 Variáveis em AIML

É possível utilizar variáveis para armazenar valores na linguagem AIML, para isso deve-se usar a *tag set* para instanciar uma variável e a *tag get* para retornar o valor da variável.

Tanto uma como a outra deve receber um atributo *name* que se refere ao nome da variável que se deseja criar (*tag set*) ou mostrar (*tag get*), e entre a *tag* o valor que se deseja passar, no caso, através do curinga, com a *tag star*.

Se uma sentença possui mais de um curinga, deve-se chamá-la de acordo com o atributo *índex* da *tag star*, sendo assim: <*star/*> receberá o valor do primeiro curinga, <*star índex="2"/*> irá receber o valor do segundo curinga e assim sucessivamente, podendo também ser utilizado <*star índex="1"*> para o primeiro curinga.

Esse recurso aliado a criatividade do *botmaster* permite que o robô de conversação armazene certos valores que poderão ser utilizados posteriormente, armazenados em um banco de dados e simular a memória do *chatterbot*, bem como auxiliar no processo de manter o diálogo e a intenção da conversação, ou gerar reações emotivas.

```
<category>
      <pattern>USANDO UMA VARIAVEL * </pattern>
      <template>
            <think><set name="var1"><star/></set></think>
            A variavel e <get name="var1"/>.
      </template>
</category>

<category>
      <pattern>USANDO A VARIAVEL * E A VARIAVEL * </pattern>
      <template>
            <think><set name="var1"><star/></set></think>
            <think><set name="var2"><star index="2"/></set></think>
            A variavel 1 e <get name="var1"/> e a variavel 2 e <get
name="var2"/>.
      </template>
</category>
```

Figura 13: Uso de variáveis em AIML

4.3 Estrutura Condicional

Foi visto, ainda neste capítulo, a possibilidade de utilizar a *tag srai* para simular condições, no entanto, a linguagem AIML possui uma *tag* específica para tal fim, onde é informado uma variável, um valor de critério e então um retorno, semelhante às estruturas condicionais das diversas linguagens existentes.

```
<category>
      <pattern>OI</pattern>
      <template>
            <condition name="session" value="desenvolvimento">
                  Desculpe. Nós já não nos saudamos antes?
            </condition>
            <condition name="session" value="fechamento">
                  Desculpe, mas já estava de saída.
            </condition>
      </template>
</category>
```

Figura 14: Uso de condicional em AIML com a *tag condition*.

Na Figura 14 é demonstrado um exemplo da *tag condition*, onde é efetuado um teste na variável *session*, caso seu valor seja “desenvolvimento”, o *chatterbot* retorna uma saída, caso o valor da variável seja “fechamento”, a saída é outra.

A *tag condition* é utilizada dentro da *tag template*, também é possível colocar condições dentro de condições, simulando um operador lógico *AND*.

```
<category>
      <pattern>OI</pattern>
      <template>
            <condition name="session" value="desenvolvimento">
                  <condition name="emotion" value="calmo">
                        Desculpe. Nos já não nos saudamos antes?
                  </condition>
                  <condition name="emotion" value="agressivo">
                        De novo, como você é repetitivo.
                  </condition>
            </condition>
      …
      </template>
</category>
```

Figura 15: Condições dentro de Condições simulando um operador AND.

Na Figura 15 há condições dentro de condições, sendo assim, dois critérios

devem ser considerados para que determinada saída seja mostrada. Na ilustração, o primeiro critério se refere à variável *session*, e o segundo se refere à variável *emotion*, de acordo com o valor dessas duas variáveis, uma determinada saída é apresentada.

Outras formas de uso da *tag condition* são apresentadas a seguir (WALLACE, 2001):

- *List-condition tag*, neste caso, é aberta uma *tag condition* e os critérios são distribuídos dentro de *tags li*, na Figura 16 é apresentado um exemplo, onde a *tag li* que não contem atributo e valor de critério é considerado como valor padrão, *default*, no caso de nenhuma das outras condições terem sido atendidas.

```
<category>
      <pattern>OI</pattern>
      <template>
            <condition>
                  <li name=”session” value=”desenvolvimento”>
                  Desculpe. Nós já não nos saudamos antes?
                  </li>
                  <li name="session" value="fechamento">
                  Desculpe, mas já estava de saída.
                  </li>
                  <li>
                  Oi, tudo bem?
                  </li>
            </condition>
      </template>
</category>
```

Figura 16: List-Condition Tag.

- *Single Name List-Conditions Tag*, nesta forma, é aberta uma *tag condition* informando a variável que se quer testar, e, nas *tag li* é apresentado valores de critérios para cada caso, semelhante às estruturas seletivas *switch* das linguagens de programação tradicionais.

```
<category>
      <pattern>OI</pattern>
      <template>
            <condition name=”session”>
                  <li value=”desenvolvimento”>
                  Desculpe. Nós já não nos saudamos antes?
                  </li>
                  <li value="fechamento">
                  Desculpe, mas já estava de saída.
                  </li>
            </condition>
      </template>
</category>
```

Figura 17: *Single Name List-Condition Tag*

4.4 Outros Recursos

A linguagem AIML possui diversas *tags*, tendo já sido abordado detalhes sobre as principais delas, que tornam AIML semelhante às linguagens de programação tradicionais, *tags* para recursão, condicionais, definição de variáveis, entrada e saída de dados.

Cabe ainda mostrar outros recursos que estão disponíveis através de algumas *tags* especiais que podem ajudar que o *botmaster* proponha maior interação na linguagem.

4.4.1 Tag System

Dentre essas, existe em AIML a *tag system*, que permite que o interpretador execute comandos do *Shell*[6], assim é possível executar através do interpretador comando de sistema que não são previstos e permitir interatividade com banco de dados, e outros sistemas, inclusive gerência de rede através do *chatterbot*(LEONHARDT, 2005).

```
<category>
      <pattern>QUE DIA E HOJE</pattern>
      <template>Hoje e <system>date</system></template>
</category>
```

Figura 18: Exemplo da Tag System executando comando do shell do sistema.

[6] *Shell*, *prompt* de comando, *Shell script*, comandos do sistema operacional.

No exemplo da Figura 18, a *tag system* executa o comando *date* do *prompt* de comando do sistema operacional, cabe ainda ressaltar que alguns tratamentos devem ser efetuados para o retorno correto do comando, visto que a *tag system* produz uma saída em linha. Sendo assim, se o comando a ser executado tiver mais de uma linha de saída, pode gerar alguns problemas de acordo com o interpretador que está sendo executado, sendo assim, o ideal é utilizar funções em uma linguagem específica e então chamá-las através do *prompt* de comando.

```
1  #!/usr/bin/env python
2
3  import os
4  import re
5  import time
6  import sys
7
8
9  def pingPython():
10     ip = "192.168.1.1"
11     pingaling = os.popen("ping -q -c2 "+ip,"r")
12     print "Testando ",ip, sys.stdout.flush()
13
14 pingPython()
```

Figura 19: Função em linguagem Python para retornar o estado de uma máquina na rede

```
<category>
      <pattern>A MAQUINA 192.168.1.1 ESTA NA REDE</pattern>
      <template>
            <system>python ping.py</system>
      </template>
</category>
```

Figura 20: Instrução em AIML para executar um arquivo em linguagem Python.

4.4.2 Substituições

Em AIML é possível utilizar fazer pequenas substituições poupando que seja necessário a criação de diversas categorias para o mesmo fim, essas substituições podem ser usadas para substituir gírias por palavras formais em sentenças, para que o *chatterbot* entenda o enunciado de acordo com a sua base de conhecimento,

trocar abreviações por palavras por extenso, que é o caso do “internetês[7]”, por exemplo: substituir “vc” por “você”.

As substituições possuem, geralmente, um arquivo específico em XML, contendo todas as possibilidades previstas pelo *botmaster* para que o robô de conversação consiga entender os enunciados das sentenças, esse arquivo possui uma divisão das substituições de acordo com o fim a que se destinam, sendo assim, as divisões são:

- *Input*: toda e qualquer entrada de dados por um usuário, que pode ser substituída por outra. Conforme NETO (2004), nesta parte, os acentos e abreviaturas são retirados.
- *Gender*: são substituições relativas ao gênero, troca de um gênero para o outro, quando usada a *tag gender.*
- *Person:* substituições da primeira para a segunda pessoa, e vice-versa, quando do uso da *tag person*.
- *Person2:* substituições da primeira para a segunda pessoa, e vice-versa. Quando do uso da *tag person2.*

```
<substitute find=” comigo “ replace=” com você “ />

<substitute find=” ele “ replace=” ela “ />

<substitute find=” vc “ replace=” você “ />

<substitute find=”ã“ replace=”a“ />
```

Figura 21: Exemplo de tag de substituição.

Um arquivo de substituições também pode ser na extensão INF[8] ao invés de XML, alguns exemplos de arquivos de substituições são apresentados no Anexo I.

4.4.3 *TAGS* para Substituições

Algumas *tags* são utilizadas para realizar as substituições detalhadas na

[7] Internetês - é um neologismo (de: *Internet* + sufixo *ês*) que designa a linguagem utilizada no meio virtual, em que "as palavras foram abreviadas até o ponto de se transformarem em uma única expressão, duas ou no máximo cinco letras". WIKIPEDIA (Disponível em http://pt.wikipedia.org/wiki/Internetês acessado em 10/05/2011)

[8] INF - Arquivo de informação, geralmente usado para ajustar parâmetros.

última seção, há quatro tipos de substituições, que são: *input*, *gender*, *person* e *person2.*

A primeira é utilizada em qualquer entrada de dados feita pelo usuário do robô de conversação, já as outras três são utilizadas de acordo com as *tags* de mesmo nome.

Sendo assim, a *tag gender* serve para fazer uma troca de gênero em uma conversação, de acordo com o que for especificado dentro do arquivo de substituições.

```
<category>
      <pattern>* VAI *</pattern>
      <template>
            Eu ouvi dizer que
            <gender><star/></gender>
            tambem vai <star index="2" />
      </template>
</category>
```

Figura 22: Uso da *tag gender* para troca do gênero na conversação.

No exemplo da Figura 22, se o usuário informar algo ao *bot* como: “Ele vai sair hoje.”, o retorno do robô de conversação seria “Eu ouvi dizer que ela também vai sair hoje.”. A *tag gender* trocará o gênero do que foi digitado no primeiro curinga, de acordo com o que é especificado no arquivo de substituições.

Já as *tags person* e *person2* trocam o sujeito da sentença, a primeira troca o sujeito da primeira para a terceira pessoa e vice-versa e da primeira para a segunda pessoa e vice-versa, respectivamente.

```
<category>
      <pattern>* VOU *</pattern>
      <template>
            <person><star/></person> vai <star index="2" />
      </template>
</category>

<category>
      <pattern>* GOSTO *</pattern>
      <template>
            Entao <person2><star/></person2> gostas
            <star index="2" />?
      </template>
</category>
```

Figura 23: Exemplos da tag person e person2.

Na Figura 23, são apresentados exemplos básicos das *tags person* e *person2*, onde se imagina que um usuário, na primeira categoria, queira informar: “Eu vou fazer uma ação.”, ao que o *chatterbot* responde: “Ele vai fazer uma ação.”, substituindo o primeiro curinga, trocando da primeira pessoa para a terceira pessoa e mantendo o segundo curinga igual. Já na segunda categoria da Figura 23, a *tag person2* troca da primeira pessoa para a segunda pessoa, caso o usuário queira informar: “Eu gosto de chimarrão.”, então o *chatterbot* responde: “Então tu gostas de chimarrão.”

Cabe lembrar que as trocas serão feitas de acordo com o conteúdo do arquivo de substituições, sendo assim, o *botmaster* poderá informar no arquivo de substituições inclusive as trocas de conjugações verbais, deixando o arquivo mais completo, os exemplos citados foram limitados a trocas dos pronomes apenas.

4.4.4 Outras *tags*

Nesta seção são apresentadas algumas outras *tags* da linguagem AIML que podem ser úteis para o acréscimo de recursos ao *chatterbot*.

A *tag gossip*, conforme WALLACE(2001), dá ao robô de conversação uma característica de aprendizado, sendo assim, o conteúdo do elemento *gossip* é processado pelo interpretador e geralmente armazenado em um arquivo separado.

Não se pode dizer que ela torne o aprendizado do robô de conversação não assistido, pois, este elemento não cria novas categorias.

A *tag learn* serve para carregar um arquivo de categorias para o *chatterbot*, sua sintaxe se dá da seguinte forma:

```
<learn filename=”arquivo.aiml” />
```

Figura 24: Tag learn para carregar arquivos AIML.

A linguagem possui elementos para capitalizar palavras como *uppercase, lowercase, sentence* e *formal*.

Uppercase torna todas as letras da sentença que estiverem dentro elemento em maiúsculas, por sua vez *lowercase* é o oposto, tornando as letras minúsculas.

Sentence deixa a primeira letra da primeira palavra em maiúscula e *formal* deixa a primeira letra de cada palavra em maiúscula, bastante usado para nomes próprios.

4.5 Interpretadores AIML

Na seção anterior foram apresentadas, detalhadamente, as *tags* ou elementos da linguagem AIML, visto que a linguagem é baseada em XML, são necessários programas específicos para o funcionamento do robô de conversação, esses programas são os interpretadores, que são desenvolvidos em muitas linguagens de computador.

Cada interpretador contém os recursos necessários para executar corretamente um robô de conversação baseado em AIML, e para tanto deve ser capaz de reconhecer os elementos da linguagem. Acompanhado a isso, cada interpretador possui funções específicas para armazenar o histórico da conversação de um robô em um arquivo de *log*[9], utilizar recursos para armazenagem temporária da conversação, simulando assim, uma espécie rudimentar de aprendizado, funções para o funcionamento do *chatterbot* na *web*, programas de mensagem instantânea entre outras funções.

Apenas para listar alguns dos interpretadores, mencionados no sítio do projeto ALICEBOT (WALLACE, 2001):

- *Program D*: interpretador em linguagem de programação JAVA, podendo atuar na *web*, programas de mensagem instantânea, *prompt* de comando e com funções para registro de conversações e outras;
- *Program E*: em linguagem PHP, o *program E* utiliza banco de dados, armazenando a base de conhecimento AIML do robô de conversação para o banco de dados, não efetuando a leitura diretamente do arquivo AIML. Também disponibiliza interfaces de conversação em *Flash*, HTML e XML-R.
- *Program P*: também conhecido como PASCALice, interpretador desenvolvido em *Delphi*.
- *Program Y*: ou *PyAIML*, desenvolvido em linguagem *Python*.

[9] Registro do histórico das conversações do robô de conversação.

- *Program Z*: baseado em *Common Lisp* e desenvolvido pela *PandoraBots*, essa implementação é a forma mais rápida de desenvolver um robô de conversação, visto que possui serviço de hospedagem de *chatterbots* e ferramentas para o desenvolvimento destes, o sítio está sempre sendo atualizado, tendo até uma versão em português e já possuindo uma base de conhecimento inicial com opções para treinamento do *chatterbot*.

Quando se trata de interpretadores AIML, inúmeras são as opções disponíveis para uso, nesta seção foram destacadas algumas das mais comuns, sendo que na maioria, as implementações são *opensource*, podendo ser acrescentados novos recursos.

5. O CHATTERBOT

Neste capítulo é feito uma síntese do que foi mostrado nos capítulos anteriores. Essa síntese será aplicada ao desenvolvimento de um *chatterbot* que será utilizado para a difusão da educação a distância no âmbito da Universidade Federal do Rio Grande do Sul.

Entre os tópicos que serão abordados ao longo do capítulo, têm-se a proposta de arquitetura, bem como um estudo da Teoria da Análise da Conversação com o intuito de aplicação no *chatterbot* para que este mantenha a coerência da conversação, também será tratado o aspecto afetivo do robô e atualização da base de conhecimento.

5.1 Arquitetura do Robô de Conversação

A proposta de arquitetura do *chatterbot* pode ser dividida em dois módulos principais e independentes e a interface com o usuário, todos eles interligados por uma camada de controle, que irá verificar as entradas e saídas e fazer as devidas regras de negócio da aplicação.

Logicamente, a interface do usuário é uma página HTML[10] que irá permitir a interação do usuário através de campos de entrada e rótulos para a saída da resposta do robô, também elementos em *flash*[11] e *javascript*[12] para possibilitar uma interação mais amigável e demonstração do caráter afetivo do *chatterbot*.

O módulo de conversação é responsável por receber a entrada do usuário, em linguagem natural, interpretá-la, e achar a devida resposta do *chatterbot* na base

[10] *Hipertext Markup Language* – linguagem de marcação de hipertexto, utilizada para construção de páginas *web*.

[11] *Adobe Flash* (antes: *Macromedia Flash*), ou simplesmente *Flash*, é um *software* primariamente de gráfico vetorial - apesar de suportar imagens bitmap e vídeos - utilizado geralmente para a criação de animações interativas que funcionam embutidas num navegador *web*. Fonte: Wikipedia

[12] É atualmente a principal linguagem para programação *client-side* em navegadores *web*. Fonte: Wikipedia

de dados AIML, sendo assim a base de dados AIML utilizada apenas para consulta aos padrões de entrada e saída. Uma segunda base de dados armazenará a entrada do usuário e a resposta recebida, bem como as intenções de conversa e outras variáveis relevantes para marcar o possível estado emocional do usuário e do robô de conversação. Essa segunda base de dados servirá como registro das conversações e interações, e possibilitará que possa ser feito a atualização da base de dados AIML.

O módulo afetivo tem por função perceber as possíveis emoções do usuário do *chatterbot* e reagir de determinada forma para cada interação, tornando o robô com um aspecto mais humano, este módulo tem três camadas que serão detalhadas mais adiante, uma é responsável por perceber as emoções, a segunda funciona como um motor de inferência na base de dados emocional do *chatterbot,* e a terceira, leva a reação resultante do módulo afetivo para a camada de controle, que se encarregará de levar a saída para o usuário.

Logo, o módulo emotivo irá interagir, por intermédio da camada de controle, com as outras duas bases de dados, pois deverão ser armazenados o estado emocional do usuário e do robô no registro de *logs,* e a base de conhecimento AIML deverá possuir reações para cada aspecto afetivo do robô, sendo assim o resultado afetivo da camada de controle deverá passar para o interpretador AIML que fará as devidas inferências na base de dados AIML.

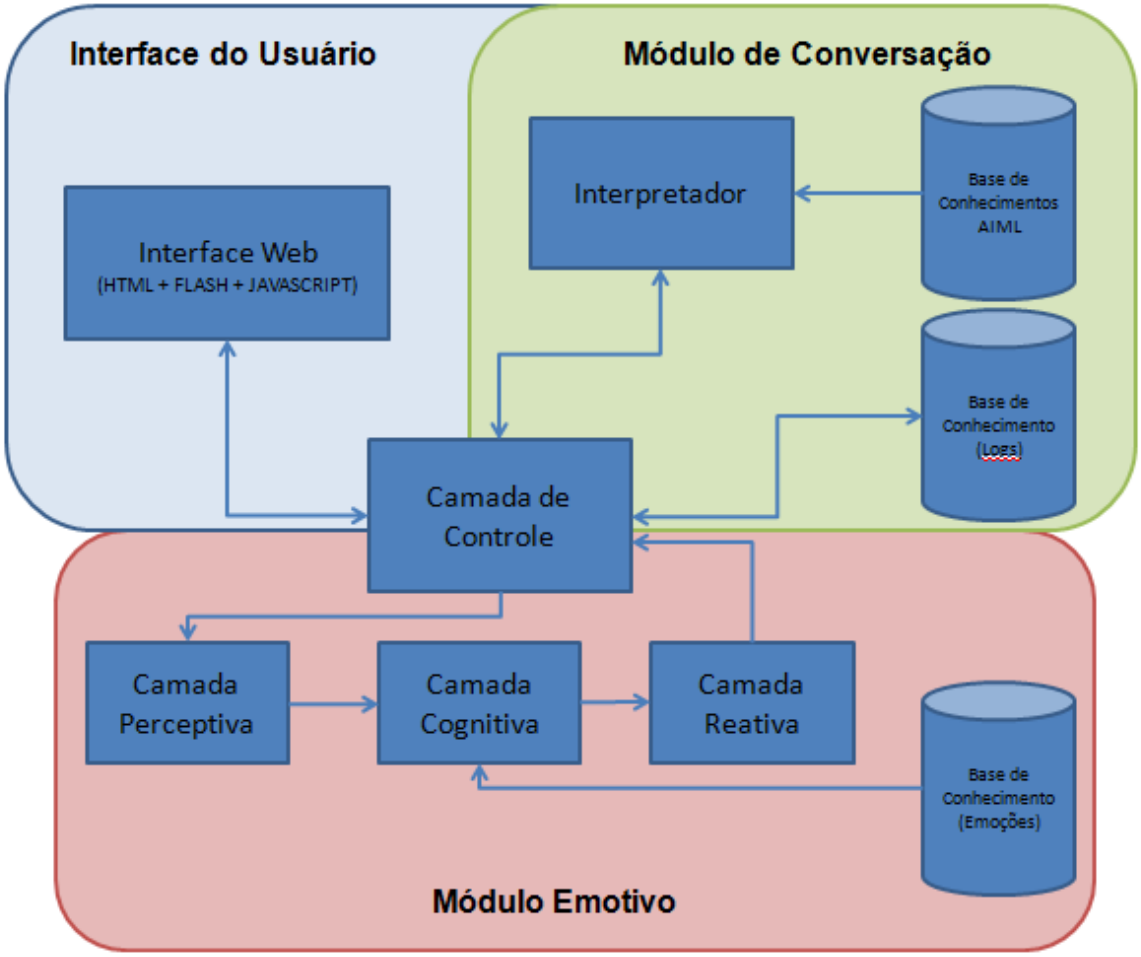

Figura 25: Proposta de Arquitetura do Robô de Conversação.

A proposta da arquitetura do Módulo Emotivo é baseada na arquitetura utilizada por Dóris 3D (FROZZA, 2009), no entanto algumas adaptações devem ser feitas, em virtude de o tutor inteligente Dóris 3D não interagir com o usuário através de linguagem natural, mas através de perguntas de múltipla escolha.

Outro ponto interessante da arquitetura proposta é que esta foi estruturada para que ficasse dentro de um padrão de desenvolvimento *web* em três camadas, conhecido como MVC (*Model View Controller*), neste padrão de desenvolvimento, o aplicativo é dividido em três partes a saber, o modelo, responsável pela camada de acesso aos dados, que no caso do robô de conversação são as bases de conhecimento afetivas, conversacionais e registro de *logs*, a camada de visão que é responsável pela interface do usuário, e a camada de controle que é responsável por organizar os dados de uma camada para a outra.

O interpretador que será utilizado para inferência na base de dados AIML é o interpretador *pyAIML*, que utiliza a linguagem *Python*.

5.2 Conversação com Intenções

Um dos problemas que pode ocorrer em sistemas de conversação, como *chatterbot*, é a limitação deste, de manter o contexto da conversação, sem desviar do assunto ou tornar a repetir certas frases.

Para contornar esse problema, NEVES(2005) apresentou o conceito de iAIML, uma adaptação da linguagem que serviria para verificar as intenções de conversação e o andamento desta, baseada na Teoria da Análise da Conversação e utilizando os elementos básicos da última versão da linguagem AIML.

> Nossa análise revelou três problemas recorrentes nesses *chatterbots*: (1) eles não levam em conta a estrutura global de uma conversação (...) e aceitam por exemplo, que saudações típicas da abertura apareçam no desenvolvimento ou no fechamento; (2) esses sistemas tratam sentenças repetidas dos usuários com base na sua estrutura sintático – morfológica – por exemplo, os sistemas não consideram “oi” e “olá” como uma repetição; e (3) muitas sentenças são tratadas como desconhecidas, quando na verdade são turnos adjacentes ao turno anterior (...) (NEVES, 2005, cap. 2)

As novas formas de uso de linguagem AIML permitiriam corrigir os diversos problemas apontados através do estudo do autor sem a necessidade de atualizações ou novas aplicações.

5.2.1 Teoria da Análise da Conversação

Uma conversação é uma interação verbal entre um ou mais interlocutores (Dittmann *apud* Marcuschi, 1991, p. 15), havendo aceitação ao tema pelos mesmos e um mínimo de conhecimento sobre o assunto a ser tratado, não sendo necessária a interação face a face.

Numa conversação toma-se como unidade o turno. Em dado momento, um falante tem o turno de fala, então um ouvinte toma o turno para si e se torna o falante atual, e há uma troca de turnos de um falante para o outro, sendo que esta

troca nem sempre é tão precisa, podendo haver interrupções ou sobreposição de falas na conversação, no caso de um *chatterbot*, o turno pode ser preciso, baseado em uma regra do tipo “fala um por vez”, visto que se fala aqui de um sistema computacional textual.

No que tange a organização local da conversação, Schegloff apud Marcuschi(1991) introduziu o termo “pares adjacentes”, que são as seqüências de turnos de uma conversação, como exemplos de pares adjacentes o autor cita (Marcuschi, 1991, p.35):

- Pergunta – resposta;
- Ordem – execução;
- Convite – aceitação/recusa;
- Cumprimento – cumprimento;
- Xingamento – defesa/revide;
- Acusação – defesa/justificativa;
- Pedido de desculpa – perdão;

Apesar da necessidade de se conhecer os pares conversacionais, não é a única importância para a análise da conversação, visto que também deve-se levar em conta a localização do turno no contexto geral, vamos tomar como exemplo o par conversacional mais comum, que é o de Pergunta – Resposta.

Sabe-se que quando se faz uma pergunta, imediatamente, espera-se uma resposta, no entanto esta resposta pode vir na forma de uma pergunta ou então como resposta do tipo sim-não, logo, deve-se analisar o resultado de acordo com o tópico onde a conversa está inserida ou o aspecto emocional do respondente.

Falante A: Você gostou da janta?

Falante B: O que você acha? (com rosto de satisfação)

Logo, analisa-se a resposta do Falante B com base no seu aspecto emocional, e presume-se que ele ficou satisfeito devido a alguma expressão corporal ou outro detalhe de relevância, como tom de voz ou gesto.

Marcuschi (1991) apresentou que além dos organizadores locais da conversação (tomada de turno, pares adjacentes), existem recursos para a organização global da

conversação. Nesse caso, o autor, diz que é comum que uma conversação tenha no mínimo três seções distintas, uma abertura, o desenvolvimento e um fechamento, sendo a abertura o contato inicial onde os falantes se conhecem, se cumprimentam, logo após se tem o desenvolvimento, onde ocorre a conversa em um ou mais tópicos de interesse mútuo, podendo haver a troca de tópicos, e então, o fechamento, que é a saída, neste caso os falantes se despedem e a conversação pode acabar momentaneamente.

Quanto à organização do tópico, Marcuschi (1991, p. 77) diz que a conversação começa com o tópico que motivou o encontro, se o encontro foi inesperado pode iniciar com a surpresa e passar para outro tópico logo em seguida.

Conforme o autor, uma conversação fluente é onde os tópicos são alternados com naturalidade, através do uso de marcadores de troca de tópicos, como: “isto me lembra aquela do”, “sim, mas mudando de assunto”, ou “mas voltando ao assunto”.

Os marcadores apresentados mostram as possibilidades de troca do tópico e quebra do tópico. No primeiro caso, troca de tópico, há o término de um tópico e introdução de um novo e no outro, quebra de tópico, há a interrupção do tópico atual podendo retornar ao anterior, é na continuidade e alternância dos tópicos que se mantém a coerência da conversação.

A organização das quebras de tópico pode ocorrer na forma de subseqüências encaixadas, quanto um tópico é introduzido podendo dar retorno ao tópico original para término, e subseqüências alternadas quando ocorre uma quebra para um novo tópico e pode haver nova quebra para o tópico original, no entanto sem haver o término dos tópicos (STECH apud MARCUSCHI, 1991, p. 82).

Sabendo essas distinções podem-se definir as subseqüências encaixadas de acordo com a relação com o tópico original, sendo:

- Subordinadas: onde o novo tópico faz parte ou tem relação com o tópico original.

Ex.: “Já que estamos falando sobre a linguagem de programação Python, deixe-me explicar um pouco sobre as técnicas de programação para você entender o que quero dizer.”

- Associativa: onde o novo tópico tem uma associação acidental com o tópico, mas não tem relevância para o desenvolvimento da conversa;

Ex.: “Isso me lembra da vez que tive no restaurante X, olha só o que aconteceu...”

- Formulativa: conforme Marcuschi (1991), esse tipo de quebra está num nível metalingüístico, onde o tópico novo é introduzido para tratar de como tratar do tópico original.

Ex.: Vamos discutir quais as melhores formas de tratar deste assunto.

5.2.2 iAIML – AIML com Intenção

A forma utilizada aqui para o tratamento de intenção, com base na Teoria da Análise da Conversação, foi apresentada por NEVES (2005), e utiliza os elementos da linguagem AIML na versão atual 1.0.1, sem a necessidade de instalação de nenhum programa adicional.

O mecanismo apresentado por NEVES utiliza três variáveis para tratar a intenção, a variável *session*, *user_intention* e *bot_intention*.

A primeira variável, *session*, indica o andamento global da conversação, com base em Marcuschi (1991). Logo, a variável pode ter três valores, a saber, que são: abertura, desenvolvimento e fechamento. As variáveis *user_intention* e *bot_intention* têm, respectivamente, a função de informar a intenção corrente do usuário e a intenção do robô.

Conforme NEVES (2005), a variável *session* tem a função principal de posicionar o andamento global da conversação para que não ocorram situações como turnos de abertura na seção de desenvolvimento, ou seja, durante a conversação em determinado tópico, o usuário retornar a abertura, saudando o robô de conversação, caso isso ocorra, a intenção do robô será a de questionar o andamento da conversa.

Todo o controle das intenções ocorrerá através da *tag set* para definir o valor das três variáveis e tópicos da conversação, e das *tags condition* e *srai* que irão testar as condições das três variáveis e a segunda para redirecionar ao resultado esperado, além claro das *tags* básicas da linguagem AIML para analisar e recuperar os padrões de conversação.

```
<set name=”session”>abertura</set>
<set name=”user_intention”>saudar</set>
<set name=”bot_intention”>perguntar como esta se sentindo</set>
```

Figura 26: Variáveis para tratamento de intenção
Fonte: (NEVES, 2005)

NEVES (2005) apresentou uma regra que serviria para tratar turnos de abertura durante a sessão de desenvolvimento, essa regra deve ser acrescentada em todas as categorias que se referem ao turno de abertura.

O funcionamento é basicamente da seguinte maneira, cada vez que um turno de abertura é chamado pelo usuário, é realizado um teste para verificar a sessão atual da conversa, caso a sessão atual seja a de desenvolvimento, então o *bot* questiona o andamento da conversa. No exemplo apresentado pelo autor, é realizado um redirecionamento para uma categoria específica que trata da redundância.

```
<condition name=”session” value=”desenvolvimento”>
    <srai>questioner andamento global</srai>
</condition>
```

Figura 27: Exemplo de regra para controle do andamento do diálogo
Fonte: (NEVES, 2005)

A função da variável *user_intention* é testar se a intenção do usuário não está sendo repetida na sentença atual, e a função de *bot_intention* é tratar sentenças desconhecidas. Conforme NEVES (2005), quando o *chatterbot* não conhece o padrão de entrada, pode ser considerado que seja um seguimento ao turno anterior, ou resposta de uma pergunta feita pelo robô de conversação, sendo assim utilizada a variável *bot_intention* para testar a intenção e redirecionar para uma saída adequada.

```
<condition name=”user_intention” value=”saudar”>
    <srai>questioner repetição</srai>
</condition>
```

Figura 28: Controle de repetições do usuário
Fonte: (NEVES, 2005)

```
<category>
      <pattern>*</pattern>
      <template>
            <condition name=”bot_intention” value=”perguntar se o
            usuario tem Projetos nos editais”>
                  Eu fui criado em um projeto do Edital 15 da
                  SEAD/UFRGS.
            </condition>
      </template>
</category>
```

Figura 29: Controle de sentenças desconhecidas com a variável *bot_intention*
Fonte: (NEVES, 2005)

As categorias da base de dados AIML do *chatterbot* deverão conter as definições de intenção, regras de controle de repetição e tratamento de sentenças desconhecidas.

```
<category>
      <pattern>OI</pattern>
      <template>
            <!-- controle do andamento global -->
            <condition name=”session” value=”abertura”>
                  <!-- definição das variáveis  -->
                  <set name=”user_intention”>saudar</set>
                  <set name=”bot_intention”>perguntar como esta se
                  sentindo</set>
                  <set name=”session”>abertura</set>
                  Oi como você está?
            </condition>
            <condition name=”session” value=”desenvolvimento”>
                   Oi, mas nós já nos saudamos antes.
            </condition>
            <condition name=”session” value=”fechamento”>
                  Desculpa, mas nós já não nos saudamos antes?
            </condition>
            <!—controle de repetições -->
            <condition name=”user_intention” value=”saudar”>
                  Desculpe, mas nós já não nos saudamos antes?
            </condition>
      </template>
</category>
```

Figura 30: Exemplo de Categoria com a intenção de Saudar
Fonte: (NEVES, 2005)

5.3 Desenvolvimento das Emoções

Para o tratamento das emoções no *chatterbot*, o resultado esperado, é que o sistema tenha a capacidade de entender o aspecto emocional do aluno durante a conversação. Com base na análise da conversa, perceber se o aluno está sendo agressivo, calmo, se está entendendo as informações do *chatterbot*.

O trabalho da análise da conversa, quanto ao aspecto afetivo, será realizado pela Camada Perceptiva, no entanto mecanismos devem ser integrados à base AIML na forma de variáveis com o intuito de definir a integração entre a base de conversação e a base afetiva, tanto para auxiliar na compreensão da emoção que o aluno quer passar, como para especificar a emoção que o *chatterbot* irá passar.

A Camada Cognitiva será capaz de realizar a inferência das variáveis captadas pela camada perceptiva na base de dados afetiva, e verificar qual a emoção foi percebida, e qual emoção deverá ser passada para o aluno, conforme o

modelo de computação afetiva que for escolhido para implementação no robô de conversação.

Dependendo da escolha do modelo de computação afetiva as variáveis podem conter além do aspecto emocional, variáveis de intensidade das emoções.

A Camada Reativa receberá o resultado obtido pela Camada Cognitiva e irá inferir diretamente na base AIML através do interpretador e também, nos recursos de interface que permitirão que o *chatterbot* demonstre a emoção através de expressões além da conversação.

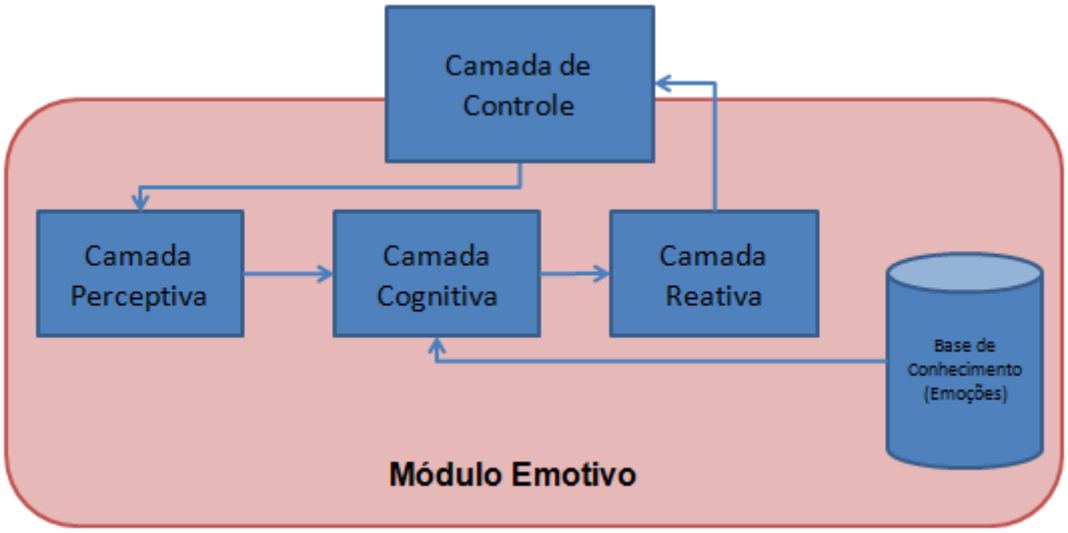

Figura 31: Módulo Emotivo do Chatterbot
Fonte: (FROZZA, 2009)

A interação entre o Módulo Emotivo, nome dado ao recurso do *chatterbot* para o tratamento de emoções, com os outros módulos (conversação e interface do usuário) será através da Camada de Controle.

O uso de três camadas para tratar as emoções é baseado no trabalho apresentado por FROZZA (2009), no entanto a Camada Cognitiva, responsável pela inferência na base de dados afetiva, bem como a modelagem dessa base de dados dependerá do modelo de computação afetiva que for utilizado.

E devido a complexidade e amplitude do tema, o mesmo será tratado em trabalho futuro.

Visto que se tem a idéia de que os módulos do *chatterbot* sejam independentes, inclusive o módulo Emotivo, pode-se trabalhar com a idéia de que este módulo seja agregado a outros sistemas, bem como também é possível trabalhar com a idéia de que as emoções do robô de conversação estejam de

acordo com a sua identidade ou personalidade, podendo assim haver um treinamento desta na base de dados afetiva, o que pode incidir que cada robô de conversação seja diferente do outro neste aspecto.

5.4 Aprendizado do Robô de Conversação

Com base no estudo realizado (WALLACE, 2009), o aprendizado dos *chatterbots* baseados em AIML, geralmente pode ocorrer de maneira supervisionada, sendo assim, é realizada uma análise dos *logs* de conversação para a realização de uma realimentação na base de dados de acordo com as entradas de conversação desconhecidas.

Como o robô de conversação proposto utiliza regras para conversação com intenção (NEVES, 2005), é possível fazer um tratamento para sentenças desconhecidas utilizando as variáveis de intenção descritas e palavras-chave inseridas nos padrões de entrada juntamente com os curingas. Isso pode ajudar a fornecer autonomia para o robô de conversação no sentido de reconhecer sentenças desconhecidas, mas ainda não constitui aprendizado.

Outra possibilidade dos *chatterbots* baseados em AIML, é a utilização de uma *tag* chamada *gossip*, no entanto ela pode ser incompatível com determinados interpretadores, como é o caso do interpretador em linguagem *Python*, que foi o escolhido para o robô de conversação proposto.

A *tag gossip* juntamente com o uso da *tag person2* e de variáveis pré-definidas no robô de conversação faz uma inserção na base de dados de conversação, não agindo diretamente sobre a base AIML, mas o texto inserido pelo aluno no padrão de entrada é tratado e inserido em um arquivo texto na forma de uma padrão de saída, assim quando algum outro usuário do *chatterbot* chamar uma *gossip* (que literalmente significa intriga), o robô escolhe uma das diversas saídas inseridas e retorna para o usuário.

Essa é uma forma do *chatterbot* ter uma espécie de aprendizado não supervisionado, porém ineficaz, visto que todos os padrões de entrada são convertidos e guardados em uma mesma categoria.

```
<category>
      <pattern>EU *</pattern>
      <template>
            <think>
                  <gossip>
                        <get name=”name”/> disse que <get
                        name=”gender”/><person2/>
                  </gossip>
            </think>
      </template>
</category>
```

Figura 32: Uso da tag gossip para inserção de padrões de entrada.

Conforme a Figura 32, pode se ter o seguinte caso, um usuário com a variável *gender* e *name* já definidas, entra com um padrão de entrada explicando alguma coisa a seu respeito, e com ajuda das variáveis e da *tag person2* responsável por trocar o conteúdo do curinga da primeira para a segunda pessoa, realiza o tratamento da frase que é guardada em um arquivo de *gossip* (intriga).

Como a *tag gossip* não é reconhecida pelo interpretador *PyAIML*, pode-se trabalhar com a idéia de desenvolver essa funcionalidade em trabalho futuro.

Já o aprendizado do módulo Emotivo pode ser realizado quando da definição do modelo computacional a ser utilizado e mecanismos para treinar a personalidade do *chatterbot*.

6. CONSIDERAÇÕES FINAIS

Aprofundou-se aqui o conhecimento sobre a linguagem AIML e como ela pode ser utilizada, juntamente com a teoria da Análise da Conversação (MARCUSCHI, 1991), para o tratamento de intenções permitindo que seja realizada uma conversa coerente entre usuário e máquina.

O recurso do *chatterbot* pode ser aplicado como FAQs[13], atendimento ao cliente, divulgação de produtos e também na forma de um tutor inteligente em uma disciplina, isso aliado a computação afetiva gera ótimos resultados para o aprendizado do aluno EAD, visto que o perfil deste mostra dinamicidade.

A idéia não é substituir o tutor EAD por um robô de conversação, mas sim implementar uma ferramenta extra para auxiliar o aluno em tempo integral, uma ferramenta que tenha a maioria das qualidades do tutor, servindo para ajudar na construção do conhecimento, motivar o aluno a procurar novas soluções e colaborar com o professor e o tutor no que diz respeito ao andamento do aluno na disciplina.

Como ponto negativo pode-se constatar a questão do aprendizado do *chatterbot* visto que o mesmo necessita de uma reavaliação da base de dados de conversação para que este aprendizado possa ocorrer, não sendo um processo autônomo.

Como ponto positivo pode-se citar o resultado do experimento realizado por NEVES (2005), no uso da iAIML comparado a AIML comum. Neste experimento, 87 usuários após diálogo com dois robôs de conversação responderam a um questionário contendo três perguntas de múltipla escolha.

[13] FAQ – *Frequently Asqued Questions*

Tabela 2: Resultados obtidos com os *chatterbots* avaliados

	AIML Padrão		**iAIML**	
	Média	Desvio	Média	Desvio
A qualidade do diálogo	1,55	0,10	1,77	0,05
O andamento da conversa	1,93	0,05	2,12	0,08
A coerência das réplicas	1,82	0,18	2,42	0,30

Adaptado de NEVES, 2005.

Para a implementação do *chatterbot* proposto foi submetido um projeto no Edital 15 SEAD/UFRGS (2011). O mesmo já foi aprovado e está em fase de desenvolvimento dentro da arquitetura apresentada, trabalho este que está sendo realizado com apoio de bolsista de graduação.

Os robôs de conversação abrem diversas possibilidades de trabalhos futuros, destacam-se a implementação do módulo Emotivo para o *chatterbot*, bem como a independência deste módulo, e a questão do aprendizado da base de dados AIML através da *tag gossip* que não é suportada pelo interpretador, podendo ser desenvolvida posteriormente.

REFERÊNCIAS BIBLIOGRÁFICAS

ARGÜIS, Ricardo ET AL. **TUTORIA, COM A PALAVRA, O ALUNO.** Coleção Inovação Pedagógica. Vol. 6. Ed. Artmed. Ano 2002.

ASHMORE, Calvin. **Ortony, Clore, and Collins: The Cognitive Structure of Emotions.** Dezembro de 2008. Disponível em: [http://ww.icosilune.com/2008/12/ortony-clore-and-collins-the-cognitive-structure-of-emotions/comment-page-1/]. Acesso em 17/01/2011.

Café TI. **Tutorial de AIML - Parte I.** Outubro de 2007. Disponível em: [http://cafe-ti.blog.br/26~tutorial-de-aiml-parte-1.html]. Acesso em: 09/06/2010.

FROZZA, R. SILVA, A.K. LUX, B. CRUZ, M.E.J.K., BORIN M. **Dóris 3D: Agente Pedadógico Baseado em Emoções.** 2009. UNISC.

KRAUS, H., FERNANDES, A. **Desenvolvimento de um Chatterbot para Área Imobiliária Integrando Raciocínio Baseado em Casos.** 2007. Disponível em: [http://revistaseletronicas.pucrs.br/fo/ojs/index.php/hifen/article/view/3857/2928]. Acesso em: 09/06/2010.

LEONHARDT, M. D., CASTRO, D. D., DUTRA, R. L. S., TAROUCO, L. M. R. **ELEKTRA: Um Chatterbot para Uso em Ambiente Educacional.** Artigo Publicado na Revista Renote. V.1 n.º 2, Setembro de 2003. Disponível em: [http://seer.ufrgs.br/renote/article/view/14336]

LEONHARDT, M. D., **Doroty: Um Chatterbot para Treinamento de Profissionais Atuantes no Gerenciamento de Redes de Computadores.** Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciências da Computação. Maio de 2005.

MARCUSCHI, Luiz Antônio. **Análise da Conversação.** 1991. Editora Ática. 2ª Edição.

MAULDIN, M. **Chatterbots, Tinymuds, And The Turing Text: Entering The Loebner Prize Competition.** 1994. Disponível em: [http://robot-club.com/lti/pub/aaai94.html]. Acesso em 09/06/2010.

MUNHOZ, A.S. **A Educação a Distância em busca do tutor ideal.** Artigo publicado na revista digital Colabora. Vol. 2. N.º 5. Setembro/2003. Disponível em: [http://www.ricesu.com.br/colabora/n5/index1.htm]. Acesso em 06/02/2011.

NETO, A. et al. **Chatterbot em AIML para o Curso de Ciência da Computação.** Artigo Publicado nos Anais do I Workcomp-Sul. Maio/2004. Florianópolis - SC Disponível em: [http://inf.unisul.br/~ines/workcomp/cd/pdfs/2312.pdf] Acesso em: 04/05/2010.

NEVES, A. M. M., BARROS, F. A. **iAIML: Um Mecanismo para Tratamento de Intenção em Chatterbots.** Artigo publicado nos anais do V ENIA (Encontro Nacional de Inteligência Artificial) – Sociedade Brasileira de Computação (SBC). Julho/2005 Disponível em [http://www.unisinos.br/congresso/sbc2005/]

ORTONY, A., CLORE, G. L., COLLINS, A. **The Cognitive Structure of Emotions.** 1999. Cambridge University Press

PAN, M.C.O. ET AL. **O tutor na formação de professores a distância: o que pensam os alunos?** Trabalho apresentado no III Seminário Internacional: As Redes de Conhecimentos e a Tecnologia. Junho/2005, UERJ/RJ. Disponível em: [http://www.lab-eduimagem.pro.br/frames/seminarios/pdf/marclopa.pdf]. Acesso em 02/02/2011.

PICARD, R.W. **Affective Computing.** 1995. Disponível em: [http://affect.media.mit.edu/pdfs/95.picard.pdf]. Acesso em 17/11/2010.

ROTHERMEL, A. Maria: Um chatterbot desenvolvido para os estudantes da disciplina "**Métodos e Técnicas de Pesquisa em Administração**". 2007. Disponível em: [http://www.aedb.br/seget/artigos07/1429_artigos2007eget.pdf]. Acesso em 04/05/2010.

RUEBENSTRUNK, Gerd. **Emotional Computers – Computer Models of Emotions and Their Meaning for emotion – psychological research.** Novembro de 1998. Disponível em: [http://www.ruebenstrunk.de/emeocomp/content.HTM], acesso em 17/01/2011.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial - Tradução da Segunda Edição.** Editora Elsevier. 2004.

STERNBERG, R. **PSICOLOGIA COGNITIVA.** Ed. Artmed. 4ª Ed. 2008. Porto Alegre.

TAROUCO, L.M.R., MORO, E.L.S., ESTABEL, L.B. **O PROFESSOR E OS ALUNOS COMO PROTAGONISTAS NA EDUCAÇÃO ABERTA E A DISTÂNCIA MEDIADA POR COMPUTADOR.** 2003. Revista Educar n. 21, Curitiba, Editora UFPR. Disponível em: [http://www.lume.ufrgs.br/handle/10183/19649] Acesso em: 07/03/2011

TURING, A. **COMPUTING MACHINERY AND INTELLIGENCE.** 1950. Disponível em: [http://www.loebner.net/Prizef/TuringArticle.html] Acesso em: 15/06/2010.

VEDOVE, J.C.D., CAMARGO, R. T. M. **A INFLUÊNCIA DA EMPATIA NA RELAÇÃO TUTOR – ALUNO.** Artigo publicado na revista InterSaberes, N.º 6, Ano 3, Jul/Dez 2008. Disponível em: [HTTP://intersaberes.grupouninter.com.br/6/índex.php] Acesso em: 02/02/2011.

WALLACE, R. **ALICEBOT.** 2003. Disponível em: [http://alicebot.blogspot.com/] Acesso em 16/06/2010.

WALLACE, R. **Artificial Intelligence Markup Language (AIML) Version 1.0.1.** 2001. Disponível em: [http://www.alicebot.org/TR/2001/WD-aiml/] Acesso em: 16/06/2010.

WALLACE, R. **A.L.I.C.E. Artificial Intelligence Foundation, Inc.** 2009. Chapter 13 The Anatomy of A.L.I.C.E. Disponível em: [http://www.alicebot.org/anatomy.html] Acesso em 16/06/2010.

ANEXO I – ARQUIVO DE SUBSTITUIÇÕES PARA PYAIML

```
[gender]
# masculino -> feminino
ele = ela
para ele = para ela
seu = sua
ele mesmo = ela mesma
# feminino -> masculino
ela = ele
para ela = para ele
sua = seu
ela mesma = ele mesmo


[person]
# 1st->3rd (masculine)
eu = ele
mim = ele
meu = dele
eu mesmo = ele mesmo
# 3rd->1st (masculine)
ele = eu
dele = meu
seu = meu
ele mesmo = eu mesmo
# 3rd->1st (feminine)
ela = eu
dela = meu
sua = minha
ela mesma = eu mesmo


[person2]
# 1st -> 2nd
eu = tu
mim = voce
meu = teu
meu = seu
eu mesmo = tu mesmo
# 2nd -> 1st
tu = eu
voce = mim
seu = meu
tu mesmo = eu mesmo

# TODO= this list is far from complete
[normal]
vc = voce
tu = VOCE
wanna = want to
gonna = going to

I'm = I am
I'd = I would
I'll = I will
I've = I have
you'd = you would
you're = you are
you've = you have
you'll = you will
he's = he is
he'd = he would
he'll = he will
she's = she is
she'd = she would
she'll = she will
we're = we are
```

we'd = we would
we'll = we will
we've = we have
they're = they are
they'd = they would
they'll = they will
they've = they have
y'all = you all
can't = can not
cannot = can not
couldn't = could not
wouldn't = would not
shouldn't = should not
isn't = is not
ain't = is not
don't = do not
aren't = are not
won't = will not
weren't = were not
wasn't = was not
didn't = did not
hasn't = has not
hadn't = had not
haven't = have not

where's = where is
where'd = where did
where'll = where will
who's = who is
who'd = who did
who'll = who will
what's = what is
what'd = what did
what'll = what will
when's = when is
when'd = when did
when'll = when will
why's = why is
why'd = why did
why'll = why will

it's = it is
it'd = it would
it'll = it will

ANEXO II – PROJETO DE IMPLEMENTAÇÃO – EDITAL 15 SEAD/UFRGS

Título do Projeto Robô de Conversação Aplicado a EAD como Tutor Inteligente

Linha Linha B

Apoio Solicitado Bolsista de Graduação

Equipamento Solicitado Kit 6 - Computador HP 6005 Pro AMD Phenom II X4 B93 6GB RAM DDR3 HD 500GB DVD-RW Placa de VÃ-deo NVidia 9500GT ou Superior ou ATI Radeon HD 2600XT ou Superior Ã• udio Integrado Windows 7 x64 bits

Justificativa para aquisição de Equipamento O desenvolvimento de um sistema de inteligência artificial como um chatterbot ou robô de conversação necessita de um equipamento capaz de suportar tecnologias de programação para web, como Java, PHP e Python que são algumas das linguagens utilizadas nos interpretadores da linguagem do chatterbot. Também, o alto processamento gráfico se justifica para o uso de aplicações gráficas para o desenvolvimento da interface que deve conter aspectos afetivos.

Membro da Equipe:

Bibiana de Lima Carapeços - Designer

Willian Esperandio - Programador

1 - Tipo de Projeto

Produção de Recursos Tecnológicos em EAD

Realização de Pesquisas em EAD

2 - Descrição

O projeto se trata do desenvolvimento de um chatterbot, ou robô de conversação que terá o objetivo de informar pessoas com pouco ou nenhum conhecimento sobre EAD, as principais atividades dessa modalidade de ensino na instituição, de uma forma amigável. Um chatterbot é um programa de computador que tem o objetivo de informar os usuários em linguagem natural, sendo assim, o programa deve manter uma conversação coerente, responder perguntas, informar, ensinar e interagir com o usuário. Alguns chatterbots tem ainda a capacidade de captar aspectos emocionais do usuário e interagir de uma forma afetiva, demonstrando emoções.

3 - Metodologia

A nível técnico, pretende-se utilizar um servidor web e configurar um interpretador AIML, linguagem de fácil aprendizado, para construção da base de conhecimento de um robô de conversação, baseado na técnica de casamento de padrões, ou seja, um usuário informa um padrão de entrada e o interpretador procura aquele padrão de entrada retornando uma resposta.

No aspecto afetivo, pretende-se utilizar uma interface em flash, que será desenvolvida pelo designer, na forma de um avatar, e também fazer um estudo sobre o modelo OCC (Ortony Clore and Collins) de afetividade computacional, para que seja possível fazer o chatterbot tratar dos aspectos afetivos dentro desse

modelo, que reconhece 22 emoções para interação com um usuário.

E referente a base de conhecimentos do chatterbot, devido ao caráter informativo do mesmo, utilizaremos informações sobre o EAD na UFRGS, cursos, monitoria EAD, capacitações EAD entre outras informações sobre o assunto.

4 - Resultados esperados

Espera-se ao final do projeto um robô de conversação que será disponível na web para informar usuários, sobre as ações de EAD, cursos, capacitações, monitoria e demais informações de EAD da UFRGS. Essa tecnologia deve chamar a atenção do usuário para que este se torne um possível aluno de EAD, também deve responder a aspectos afetivos e armazenar em log as conversações para que a base de dados possa sofrer alterações com o tempo tornando o robô de conversação cada vez melhor.

A nível acadêmico espera-se também, a elaboração de trabalho a ser apresentado no salão de ensino ou similar, bem como publicação em revista na área da educação.

5 - Proposta de Avaliação do Projeto

A proposta de avaliação do projeto será através do volume de informação contida na base de dados AIML, o que possibilitará saber se o robô de conversação terá um bom desempenho conversacional. E estudo dos logs de conversações anteriores para futuras modificações na base de conhecimento.

Quanto a aspectos afetivos espera-se implementar o modelo OCC (Ortony Clore and Collins) de computação afetiva para que o chatterbot possa interagir reconhecendo 22 tipos de emoções.

6 - Plano de trabalho do Bolsista

Num momento inicial o bolsista deverá aprender a linguagem AIML, bem como recursos do interpretador, tem-se a idéia de utilizar um interpretador em Python, para então o bolsista começar nos três primeiros meses do projeto trabalhar na alimentação da base de conhecimentos gerais e específicos do chatterbot.

Num segundo momento, espera-se a implementação de recursos de afetividade e interação com banco de dados.
Os três últimos meses do projeto serão dedicados a validação do projeto, testes e melhorias na qualidade das respostas e afetividade.