

PONTÍFICIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Unidade Betim
Bacharelado em Sistemas de Informação

Hitalo Filipe Silva de Araujo

**PROJETO E DESENVOLVIMENTO DE UM ARCABOUÇO DE AGENTE DE
CONVERSAÇÃO APLICADO AO CURSO DE SISTEMAS DE INFORMAÇÃO DA
PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

Betim
2016

Hitalo Filipe Silva de Araujo

**PROJETO E DESENVOLVIMENTO DE UM ARCABOUÇO DE AGENTE DE
CONVERSAÇÃO APLICADO AO CURSO DE SISTEMAS DE INFORMAÇÃO DA
PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

Trabalho apresentado ao curso de Sistemas de Informação da Pontifícia Universidade Católica de Minas Gerais, unidade Betim, como requisito parcial para obtenção do título de bacharel em Sistemas de Informação.

Orientador: Prof. Me. Camillo Jorge Santos Oliveira

**Betim
2016**

Hitalo Filipe Silva de Araujo

**PROJETO E DESENVOLVIMENTO DE UM ARCABOUÇO DE AGENTE DE
CONVERSAÇÃO APLICADO AO CURSO DE SISTEMAS DE INFORMAÇÃO DA
PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**Trabalho apresentado ao curso de Sistemas
de Informação da Pontifícia Universidade
Católica de Minas Gerais, unidade Betim,
como requisito parcial para obtenção do
título de bacharel em Sistemas de
Informação.**

Prof. Me. Camillo Jorge Santos – PUC Minas (Orientador)

Prof. Aluísio Eustáquio da Silva – PUC Minas (Banca Examinadora)

Prof. Dr. Sandro Laudares – PUC Minas (Banca Examinadora)

Betim, 08 de junho de 2016.

"a questão de saber se as máquinas podem pensar (...) é tão relevante como a questão de saber se os submarinos podem nadar." (DIJKSTRA *apud* RUSSELL; NORVIG, 2013, p. 890)

RESUMO

O objetivo deste trabalho é projetar e desenvolver um agente de conversação aplicado ao curso de Sistemas de Informação da PUC Minas a fim de aproveitar informações armazenadas em algum meio digital e disponibilizá-las para consultas em linguagem natural. A metodologia envolveu o projeto e desenvolvimento dos módulos de interpretação da linguagem de marcação AIML, de interface com o *software SWI-Prolog* (linguagem *Prolog*) e de modelos probabilísticos de Recuperação de Informação. Envolveu também o levantamento de 51820 perguntas/respostas provenientes do sítio na Internet *ask.fm/pucminas* (conta oficial da PUC Minas) e de outras centenas de documentos do curso de Sistemas de Informação. Com a métrica MRR obteve-se uma taxa de acerto de 91,3% para o Modelo Probabilístico Clássico e 97,5% para o *Best Match 25*.

Palavras-chave: Agentes de Conversação. Inteligência Artificial. Recuperação da Informação. .NET. *Prolog*. AIML. BM25, Modelo Probabilístico.

LISTA DE ABREVIATURAS E SIGLAS

AC – Agente de Conversação

AFD – Autômato Finito Determinístico

AIML – Artificial Intelligence Markup Language

ALICE – Artificial Linguistic Internet Computer Entity

ASP – Active Server Pages

BM25 – Best Match 25

DTD – Definição de Tipo de Documento

ER – Expressão Regular

HTML – Hypertext Markup Language

IA – Inteligência Artificial

MPC – Modelo Probabilístico Clássico

MRR – Mean Reciprocal Rank

PUC Minas – Pontifícia Universidade Católica de Minas Gerais

RI – Recuperação de Informação

RR – Reciprocal Rank

SGML – Standard Generalized Markup Language

SI – Sistemas de Informação

XML – Extensible Markup Language

SUMÁRIO

1.	INTRODUÇÃO	8
1.1	Objetivo Geral.....	9
1.2	Objetivos Específicos	9
1.3	Justificativa	9
1.4	Estrutura da monografia.....	10
2	REFERENCIAL BIBLIOGRÁFICO.....	11
2.1	Agentes de Conversação notáveis	11
2.2	Inteligência Artificial	12
2.2.1	<i>Agentes em Inteligência Artificial</i>	13
2.2.2	<i>Alan Turing e o Prêmio Loebner</i>	14
2.3	AIML	15
2.4	Recuperação de Informação	15
2.4.1	<i>Modelo Probabilístico</i>	16
2.4.2	<i>BM25</i>	17
2.4.3	<i>Métrica MRR</i>	17
3	METODOLOGIA.....	19
3.1	Construção da base de documentos.....	19
3.2	Modelagem do AC proposto	20
3.3	Interpretador AIML	23
3.3.1	<i>Autômato finito determinístico</i>	24
3.3.2	<i>Base de arquivos AIML</i>	26
3.4	Modelo Probabilístico Clássico de RI.....	27
3.5	BM25.....	30
3.6	Integração do interpretador AIML com <i>SWI-Prolog</i>	32
3.7	Comparação entre as três abordagens	33
3.8	Casamento de Padrões e Tratamento de Texto	34
3.8.1	<i>Algoritmo de Sellers</i>	34
3.8.2	<i>Transformação Léxica</i>	36
3.8.3	<i>Eliminação de stopwords</i>	37
3.9	Experimentos	37
3.10	Sistema <i>Web</i> de agente de conversação	40
4	CONCLUSÃO.....	48
4.1	Contribuições	48
4.2	Trabalhos Futuros.....	49

REFERÊNCIAS	50
APÊNDICE A – Marcações processadas pelo Interpretador AIML	53
APÊNDICE B – Teste de desempenho com MPC e BM25	56
ANEXO A – AIML 1.0.....	59

1. INTRODUÇÃO

O elevado tempo de atendimento na resolução de problemas em *callcenters* de operadoras de celular, televisão a cabo e outros ramos pode gerar transtornos a clientes e a empresas que prestam este tipo de serviço. Não raro, clientes precisam aguardar longos períodos para que sejam atendidos em suas solicitações a estas entidades (SCHIESSL, 2007).

A grande demanda de questionamentos pode sobrecarregar um setor destinado para auxílio. Uma alternativa utilizada por muitos é a criação de um conjunto de perguntas frequentes com suas respectivas respostas. Essa pode ser uma boa abordagem para um conjunto pequeno, em casos onde a coleção de informações é muito grande percorrer vários documentos sem utilizar alguma ferramenta computacional é inviável (TEIXEIRA, 2005).

Uma forma de tornar mais ágil e automatizar este tipo de processo, sem diminuir a qualidade de atendimento, seria a construção de um sistema capaz de receber, interpretar e responder solicitações que não necessariamente exigem um alto grau de complexidade. Assim, dúvidas simples poderiam ser sanadas sem intervenção de uma pessoa, garantindo, em uma grande proporção de casos, uma interação mais rápida e com uso de menos recursos humanos em um sistema dimensionado para permitir o fluxo padrão de atendimento.

Um Agente de Conversação é um programa de computador criado com a finalidade de imitar a capacidade humana de comunicar. Ou seja, é um programa que dada uma frase tenta formular uma resposta adequada, tal como faria um humano. (BRANDÃO, 2012, p. 16).

Este trabalho apresenta uma proposta de projeto e desenvolvimento de um arcabouço de Agente de Conversação (AC) aplicado ao curso de Sistemas de Informação (SI) da Pontifícia Universidade Católica de Minas Gerais. Um AC é um programa ou conjunto de programas que imita a capacidade humana de comunicação. *Ella*, *Albert-One*, *Tinymod* e *Pixel* são alguns dos agentes citados em Teixeira (2005). *ELIZA* e *ALICE* (*Artificial Linguistic Internet Computer Entity*) são dois dos mais conhecidos (WALLACE, 2015); (WEIZENBAUM, 1966).

O AC proposto responde a questionamentos relacionados ao curso de SI e a processos da universidade, que podem ser respondidos via agente computacional ao invés de um agente humano.

Assim, poderia se dizer que a construção de um sistema computacional¹ fundamentado com teorias de IA (Inteligência Artificial) e de RI (Recuperação de Informação), capaz de atender ao princípio fundamental de um agente de conversação (a de tornar a conversação homem-máquina semelhante à conversação homem-homem) é uma forma de trazer informação relevante para uma pessoa.

1.1 Objetivo Geral

Projetar e desenvolver um Arcabouço de Agente de Conversação aplicado ao curso de Sistemas de Informação da PUC Minas Betim capaz de responder perguntas.

1.2 Objetivos Específicos

Os objetivos específicos desta monografia são:

- Obter, tratar e armazenar uma base de documentos relacionada à universidade;
- Criar uma aplicação *Web* de AC;
- Prover respostas para perguntas e permitir meios para desenvolvimento contínuo da base de documentos.

1.3 Justificativa

O tempo necessário para uma pessoa obter respostas, por mais simples que estas sejam, pode ser consideravelmente grande para que ela fique satisfeita, além disso, a disposição de recursos humanos para tarefas simples é financeiramente custosa. A disponibilização, em meio eletrônico ou físico, de uma coleção de perguntas e respostas também pode ser feita sem auxílio de ferramentas para manipulação.

A elaboração de um AC foi identificada como uma maneira de tratar uma grande quantidade de informação com a flexibilidade das consultas em linguagem natural. Assim, um usuário pode obter benefícios da RI e da IA com pouco ou

¹ Sistema computacional entenda-se por *hardware* e *software*

nenhum treinamento e uma instituição pode reduzir custos sem prejudicar seus usuários, nesse caso, os alunos.

Pode se dizer também que a automatização desse processo pode diminuir a mobilização de funcionários para essa função e o tempo necessário para respostas. Isso se deve aos fatos de que após ter uma base de dados significativa, o sistema precisa apenas de poucos incrementos nesta base e um servidor bem dimensionado pode responder a solicitações.

1.4 Estrutura da monografia

Nos demais capítulos desta monografia encontram-se todos os aspectos relacionados com a proposta do Projeto e Desenvolvimento de um Arcabouço de Agente de Conversação aplicado ao curso de Sistemas de Informação da Pontifícia Universidade Católica de Minas Gerais.

No Capítulo 2, tem-se um referencial bibliográfico, onde são abordados os assuntos: agentes de conversação notáveis, questões de Inteligência Artificial, agentes em Inteligência Artificial, *Alan Turing* e o prêmio *Loebner*, a linguagem de marcação AIML e, finalmente, Recuperação da Informação.

O Capítulo 3 apresenta a metodologia utilizada nas etapas do projeto e desenvolvimento. São apresentados: a base de documentos, a modelagem do AC proposto, interpretador AIML, o modelo probabilístico clássico de recuperação de informação, o modelo *Best Match* 25, a integração com o programa *SWI-Prolog*, a comparação entre as abordagens para busca de respostas, casamento de padrões e tratamento de texto, experimentos e, finalmente, o sistema *Web* de agente de conversação.

O Capítulo 4 conclui o trabalho apresentando as contribuições e propostas para trabalhos futuros.

2 REFERENCIAL BIBLIOGRÁFICO

Este capítulo traz o levantamento bibliográfico sobre os principais conceitos relacionados ao Projeto e Desenvolvimento de um Arcabouço de Agente de Conversação aplicado ao curso de Sistemas de Informação da Pontifícia Universidade Católica de Minas Gerais proposto.

A Seção 2.1 apresenta os ACs mais importantes. A Seção 2.2 apresenta informações e análises relacionadas à Inteligência Artificial. A Seção 2.3 descreve o padrão AIML 1.0. Finalmente, a Seção 2.4 apresenta modelos e técnicas de Recuperação de Informação utilizados.

2.1 Agentes de Conversação notáveis

Os agentes de conversação atuais fazem uso dos recursos mais avançados que a evolução da tecnologia pode oferecer, porém, isso não tira o brilho das primeiras aplicações que ainda hoje servem de referência para suas sucessoras.

Em meados do ano de 1966, *Joseph Weizenbaum* criou ELIZA, a primeira construção de sistema que faz interação com um humano amigavelmente, (TEIXEIRA, 2005). ELIZA segue uma abordagem que leva em consideração perguntas baseadas na resposta anterior do usuário, formando um diálogo conexo e livre da necessidade de bases grandes de conhecimento (WEIZENBAUM, 1966).

Algum tempo depois surgiu ALICE, de *Richard S. Wallace*, variante do agente de *Weizenbaum* que trouxe consigo a linguagem de marcação AIML (WALLACE, 2005). Esse até então novo robô, agradou a comunidade da área e conquistou diversos prêmios internacionais. Ainda hoje, ALICE recebe contribuição da comunidade de entusiastas na Internet, fato que a proporcionou uma gigantesca base de conhecimento.

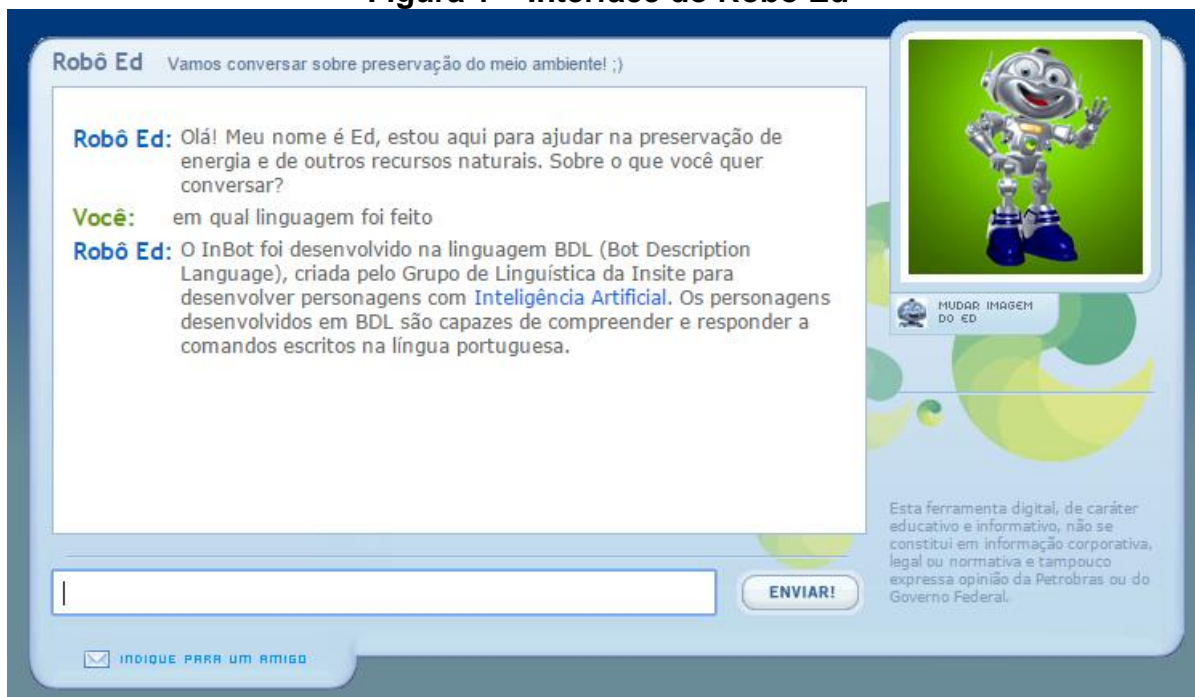
Existem muitos robôs de conversação, seja para uso no apoio a educação como agentes educacionais, para interação com clientes ou apenas para entretenimento. *Emile*, *Autotutor*, *AZ-ALICE* e *TARA* são alguns desses agentes citados em Brandão (2012).

No Brasil, pode-se dizer que um dos mais notáveis é o Robô Ed, agente criado para *Conpet*² e *Petrobrás*³ que dá dicas sobre temas ambientais, (CONPET,

² Empresa ligada a Petrobrás

2016). Como o Robô Ed, os ACs têm domínios de conhecimento específicos e limitados, eles frequentemente respondem com frases genéricas quando nenhuma resposta armazenada se encaixa com uma entrada do usuário. A Figura 1 apresenta a interface *Web* do robô Ed.

Figura 1 – Interface do Robô Ed



Fonte: CONPET, 2015

2.2 Inteligência Artificial

A Inteligência Artificial (IA) é o campo nascido em meados dos anos 50 que estuda as possibilidades de um computador perceber, raciocinar e agir (WINSTON *apud* RUSSELL; NORVIG, 2013). Russel e Norvig (2013) esclarecem que:

Atualmente, a IA abrange uma enorme variedade de subcampos, do geral (aprendizagem e percepção) até tarefas específicas, como jogos de xadrez, demonstração de teoremas matemáticos, criação de poesia, direção de um carro em estrada movimentada e diagnóstico de doenças. A IA é relevante para qualquer tarefa intelectual, é verdadeiramente um campo universal. (RUSSELL; NORVIG, 2013, p. 3).

Dois conceitos cercam esse ramo: IA forte e IA fraca. A IA forte é a definição de que os computadores poderiam ser como mentes humanas, nas suas ações, modo de pensar e de criar, o que para muitos teóricos seria algo impossível. Já a IA fraca é o conceito de que as máquinas não precisam ser totalmente como pessoas,

³ Empresa brasileira do ramo petrolífero

mas devem agir como elas, assim como o que os agentes de conversação buscam a princípio. Quatro abordagens vêm sendo seguidas nesse sentido: pensando como um humano, pensando racionalmente, agindo como seres humanos e agindo racionalmente (RUSSELL; NORVIG, 2013).

2.2.1 Agentes em Inteligência Artificial

Em IA e em outras áreas da computação, um agente é aquilo que age e percebe seu ambiente por meio de sensores e atuadores. Russel e Norvig (2013) afirmam que:

“Um agente de *software* recebe sequências de teclas digitadas, conteúdo de arquivos e pacotes de rede como entradas sensórias e atua sobre o ambiente exibindo algo na tela, escrevendo em arquivos e enviando pacotes de rede.” (Russel e Norvig, 2013, p. 31).

O Quadro 1 apresenta o ambiente de tarefa de um AC. Um ambiente de tarefa é onde um agente é inserido para que possa efetuar as ações.

Quadro 1 - Ambiente de tarefa

Tipo de Agente	Agente de Conversação
Medida de desempenho	Responder perguntas feitas por alunos
Ambiente	Alunos de Sistemas de Informação – PUC
Atuadores	Exibir respostas e escrever em arquivos
Sensores	Teclado, mouse e arquivos.

Fonte: Elaborado pelo autor com informações extraídas de Russel e Norvig, 2013, p. 37.

O ambiente de tarefa de um AC pode ser parcialmente observável, multiagente, determinístico, episódico e estático. Segundo Russel e Norvig (2013):

- Parcialmente observável por não ter acesso completo ao ambiente em que está inserido;
- Multiagente por não atuar sozinho;
- Determinístico porque o próximo estado do ambiente é determinado pelo estado atual e pela ação do agente;
- Episódico por receber uma percepção e logo em seguida executar uma ação;

- Estático pelo fato de que o ambiente não muda enquanto o agente está procurando a ação necessária.

2.2.2 Alan Turing e o Prêmio Loebner

Alan Mathison Turing foi um cientista britânico famoso por ser um dos pioneiros a acreditar que os computadores, ou até então as máquinas em geral, poderiam um dia se tornar inteligentes como um ser humano. Longe de ficar apenas no campo da imaginação, *Turing* partiu para a criação de um modelo teórico matemático, fato interessante devido que esse modelo seria para uma máquina inexistente na época, o computador, que recebeu o nome de Máquina de *Turing*. Contribuiu de forma significativa na segunda guerra mundial desenvolvendo um equipamento para decifragem de mensagens alemãs (RUSSELL; NORVIG, 2013).

On computable numbers, with application to the Entscheidungs de 1936, trabalho de *Turing*, foi considerado o início da era digital (TURING, 1936). Também introduziu o conceito de IA no mundo científico. Fora do campo computacional, o cientista iniciou o campo da morfogênese na biologia.

A maior competição mundial voltada para IA se baseia no Teste de *Turing*, teste esse que definia características básicas para que um computador fosse considerado inteligente. Nele, o computador seria aprovado se um interrogador humano não conseguisse descobrir se as respostas vêm de uma pessoa ou máquina (RUSSELL; NORVIG, 2013), com comunicação por meio de texto. *Russell* e *Norvig* citam as características mínimas de uma máquina para ser aprovada nesse teste: capacidade de processamento de linguagem natural, representação de conhecimento, raciocínio automatizado e aprendizado. Para um sistema completo os autores citam também a visão computacional e a robótica como características complementares.

O Premio *Loebner* de competição anual realizada desde 1991, premia projetos que obtém desempenho mais próximo do ideal. *Rose*, de *Bruce Wilcox*, foi o último robô de conversação a ganhar o prêmio, em 2015 (LOEBNER, 2015).

O projeto e desenvolvimento deste arcabouço não visa necessariamente um algoritmo para atender ao teste de *Turing*, mas que atende ao conceito base criado pelo cientista, ou seja, comunicação próxima à comunicação humana.

2.3 AIML

A linguagem de marcação XML (*Extensible Markup Language*, Linguagem de Marcação Extensível) é uma linguagem baseada na metalinguagem SGML (*Standard Generalized Markup Language*) que permite a criação de marcações ilimitadas (DEITEL, 2003).

AIML (*Artificial Intelligence Markup Language*) é uma linguagem de marcação voltada para IA que usa a linguagem XML para armazenar e estruturar categorias. Uma categoria é uma unidade de conhecimento em um documento AIML (TEIXEIRA, 2003). ALICE (WALLACE, 2003), *BonoBOT* (SGANDERLA, 2003), *Doroty* (LEONHARDT, 2005) e *Tuxbot* (TEIXEIRA, 2003) são ACs que utilizam AIML.

A Figura 2 apresenta uma categoria AIML onde uma entrada “O que é RI?” leva a uma resposta “RI – Recuperação de Informação”.

Figura 2 - Categoria AIML

```
<aiml>
  <category>
    <pattern>O que é RI?</pattern>
    <template>RI - Recuperação de Informação</template>
  </category>
</aiml>
```

Fonte: Elaborado pelo autor

De acordo com Teixeira (2003) as principais marcações AIML são:

- **<aiml>**: Identifica o início e o fim de um documento AIML;
- **<category>**: Identifica uma unidade de conhecimento em um documento AIML;
- **<pattern>**: Identifica o padrão digitado pelo usuário que o sistema usa para procurar respostas;
- **<template>**: Contém a resposta da categoria e outras possíveis marcações.

2.4 Recuperação de Informação

Segundo Baeza-Yates e Ribeiro-Neto (2013), a Recuperação de Informação (RI) é uma área da Ciência da Computação que provém aos usuários acesso fácil às

informações e estuda a representação, o armazenamento, a organização e acesso a documentos, páginas *Web* e outros registros.

Baeza-Yates e Ribeiro-Neto (2013) afirmam também que a modelagem de RI pode se dividir em duas tarefas principais, que seriam a concepção de um arcabouço lógico e a definição de uma função de ordenamento para computar o grau de similaridade entre documentos e as consultas do usuário. Os modelos vetorial, booleano e probabilístico formam o conjunto de modelos clássicos de RI utilizados em arcabouços lógicos.

Um sistema de RI é uma quádrupla $(D, Q, F, R(q_i, d_j))$, onde D é o conjunto de visões lógicas da coleção; Q o conjunto das representações das consultas do usuário; F um arcabouço para modelagem do sistema e $R(q_i, d_j)$ uma função de ordenação que associa um número real à representação de uma consulta $q_i \in Q$ a cada documento $d_j \in D$ (BAEZA-YATES; RIBEIRO-NETO, 2013).

Tuxbot é um exemplo de AC que faz uso de modelos de RI para busca de respostas. *Tuxbot* tem o objetivo de esclarecer dúvidas relacionadas ao sistema operacional *Linux* (TEIXEIRA, 2005).

2.4.1 Modelo Probabilístico

O modelo probabilístico é um modelo de RI que propõe recuperar documentos relevantes ao usuário por intermédio da probabilidade. O modelo foi desenvolvido por *Robertson e Jones* (1976).

Dada uma consulta de usuário q e um documento d_j da coleção, o modelo probabilístico tenta estimar a probabilidade do usuário achar o documento d_j interessante (isto é, relevante). O modelo supõe que essa probabilidade de relevância depende apenas das representações da consulta e do documento, ou seja, das informações disponíveis ao sistema. (BAEZA-YATES; RIBEIRO-NETO, 2013, p. 48).

A Equação 1 apresenta a equação de similaridade para obter a relevância de documentos quando não se sabe previamente o conjunto de documentos relevantes:

$$sim(d, BUSCA) = \sum_{i=1}^t \log_2 \left(\frac{N + 0,5}{n_i + 0,5} \right) \quad (1)$$

Onde t é o número de termos da consulta presentes no documento i ; N é igual à quantidade de documentos que a base possui e n_i é o número de documentos em que o termo t aparece.

2.4.2 BM25

O modelo BM25 (*Best Match 25*) é resultado de experimentos sobre o modelo probabilístico clássico (MPC) de *Robertson e Jones* (1976) utilizando três princípios: frequência inversa de documentos, frequência dos termos e normalização pelo tamanho dos documentos (BAEZA-YATES; RIBEIRO-NETO, 2013). A Equação 2 é a equação de similaridade do BM25.

$$sim_{BM25}(d_j, BUSCA) = \sum_{k_1[q, d_j]} B_{i,j} \times \log\left(\frac{N - n_i + 0,5}{n_i + 0,5}\right) \quad (2)$$

onde

$$B_{i,j} = \frac{(K_1 + 1)f_{i,j}}{K_1 \left[(1 - b) + b \frac{len(d_j)}{avg_doclen} \right] + f_{i,j}} \quad (3)$$

sendo:

- K_1 e b : Constantes empíricas;
- N : Número de documentos da base;
- n_i : Número de documentos em que o termo t aparece;
- $f_{i,j}$: Frequência do termo k_1 no documento d_j ;
- $len(d_j)$: Tamanho em caracteres do documento d_j ;
- avg_doclen : Tamanho médio dos documentos da base.

Baeza-Yates e Ribeiro-Neto (2013) indicam que as constantes K_1 e b podem receber os valores 1 e 0,75, respectivamente, para coleções genéricas.

2.4.3 Métrica MRR

Em sistemas de buscas e respostas que tem por objetivo recuperar uma resposta em vez de produzir uma lista de documentos é desejável uma métrica que favoreça resultados onde a primeira resposta correta esteja no topo do *ranking* (BAEZA-YATES; RIBEIRO-NETO, 2013). Uma métrica que apresenta boas

características para esses sistemas é a MRR (*Mean Reciprocal Rank*). O RR (*Reciprocal Rank*) é definido na Equação 4:

$$\begin{cases} \frac{1}{S_{\text{correto}}(R_i)}, & \text{se } S_{\text{correto}}(R_i) < S_h \\ 0, & \text{caso contrário} \end{cases} \quad (4)$$

A função $S_{\text{correto}}(R_i)$ retorna a posição da primeira resposta correta. Quando essa posição é menor que a posição limite S_h o RR é igual a razão de 1 pela posição da resposta, caso o contrário o RR é igual a zero. O MRR é a média de RR para uma coleção de perguntas Q , como apresentado na Equação 5.

$$MRR(Q) = \frac{1}{N_q} \sum_{i=1}^{N_q} \frac{1}{S_{\text{correto}}(R_i)} \quad (5)$$

Ainda na Equação 5, N_q é o número de consultas do conjunto sobre o qual a métrica é aplicada.

3 METODOLOGIA

Nos Capítulos anteriores, observa-se a importância dos AC, IA e RI para o projeto e desenvolvimento de um arcabouço de agente de conversação. Também foram descritas algumas ferramentas, tais como, AIML, modelo probabilístico de RI clássico e uma variação que seria o BM25, além da métrica MRR, utilizadas para projetar e desenvolver o AC proposto.

Este capítulo dedica-se a descrição da metodologia utilizada no Projeto e Desenvolvimento de um Arcabouço de Agente de Conversação aplicado ao curso de Sistemas de Informação da Pontifícia Universidade Católica de Minas Gerais.

O restante do capítulo encontra-se dividido da seguinte maneira: a Seção 3.1 descreve a construção da base de documentos, na Seção 3.2 a modelagem do AC proposto é apresentada, a Seção 3.3 apresenta o interpretador utilizado para interpretação de marcações AIML; a Seção 3.4 descreve o MPC; a Seção 3.5 descreve o BM25; na Seção 3.6 é apresentado como o interpretador AIML faz comunicação com o *SWI-Prolog*; na Seção 3.7 é apresentada a comparação entre as três abordagens para busca de respostas; a Seção 3.8 descreve como o casamento e o tratamento de texto é feito; e, finalmente, a Seção 3.9 faz a descrição dos experimentos para validar o AC proposto utilizando a métrica MRR.

3.1 Construção da base de documentos

A construção de uma base de conhecimento grande é uma tarefa árdua se feita manualmente. Por isso, foi desenvolvido um programa na linguagem C# (*C Sharp*) para extração de informações da conta oficial da PUC Minas no sítio <http://ask.fm/pucminas>. Até o dia 23 de fevereiro de 2016 a conta da PUC Minas possuía 51820 perguntas/respostas. Neste sítio, pessoas (entre elas alunos, ex-alunos e vestibulandos) fazem perguntas que são respondidas por funcionários da universidade.

Pode-se considerar que os documentos extraídos são relevantes e confiáveis, primeiro porque provêm de necessidades reais, segundo porque as respostas são escritas por profissionais capacitados.

As 51820 perguntas/respostas estavam armazenadas em 2073 páginas HTML. Estas páginas foram copiadas e salvas em documentos separados. Cada

documento foi processado para que as perguntas/respostas fossem extraídas e copiadas para documentos de extensão *.txt*.

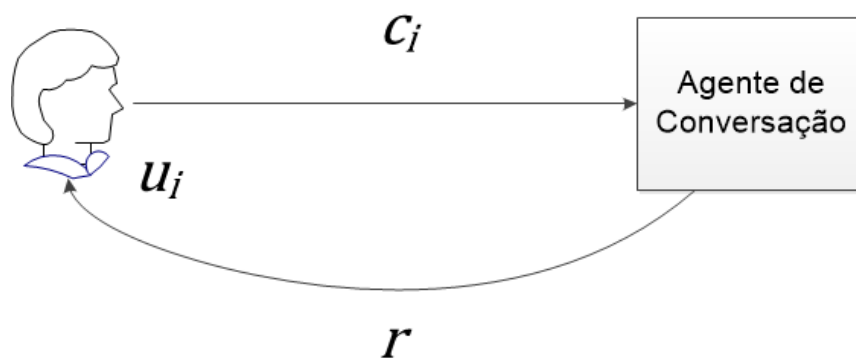
Cada documento possui uma pergunta e uma resposta. Cada pergunta é delimitada pelas marcações “<think>” e “</think>”, dessa forma é possível identificar o que é pergunta e o que é resposta em documento⁴.

A coleção de documentos também contém informações de cursos diferentes do curso de Sistemas de Informação, fato devido a dificuldade em separar essas informações, informações extraídas manualmente de documentos das dezenas de matérias do curso e documentos com informações geradas pelo autor.

3.2 Modelagem do AC proposto

A Figura 3 apresenta o modelo de definição de funcionamento ou a definição de um AC que possui como componentes um conjunto de usuários U e o AC. Dado um conjunto de consultas C , um conjunto de respostas R , um usuário $u_i \in U$, para $i \in N^*$, faz uma consulta $c_i \in C$, para $i \in N^*$ e o AC retorna uma resposta $r_i \in R$, para $i \in N^*$.

Figura 3 - Definição de um AC



Fonte: Elaborado pelo autor

A Figura 4 apresenta o modelo de definição de funcionamento ou a definição do AC que é proposto. Identificou-se que diferentes abordagens para obtenção de respostas seria uma forma de obter melhores resultados. O AC pode responder o usuário de três formas.

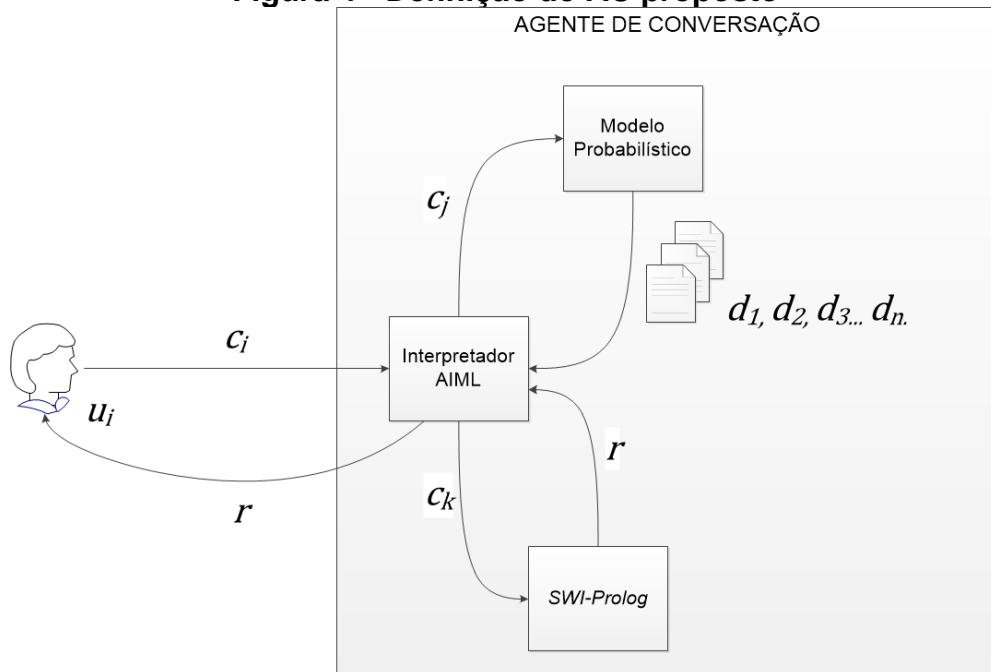
⁴ Apesar de também serem encontradas nos documentos do Modelo Probabilístico, estas marcações são utilizadas apenas no interpretador AIML para fins de controle.

A primeira forma de obter uma resposta é utilizando informações apenas dos documentos AIML, ou seja, do interpretador AIML. Dado um conjunto de consultas C , um conjunto de respostas R , um usuário $u_i \in U$, para $i \in N^*$, faz uma consulta $c_i \in C$, para $i \in N^*$ e o AC retorna uma resposta $r_i \in R$, para $i \in N^*$.

Na segunda forma de obter uma resposta, o AC utiliza o Modelo Probabilístico. Dado um conjunto de consultas C , um conjunto de respostas R , uma coleção de documentos D , um usuário $u_i \in U$, para $i \in N^*$, faz uma consulta $c_i \in C$, para $i \in N^*$, o interpretador AIML faz uma consulta $c_j \in C$, para $j \in N^*$ e o Modelo Probabilístico retorna um conjunto de $d_i \in D$, para $i \in N^*$ em ordem decrescente de relevância e o AC retorna uma resposta $r_i \in R$, para $i \in N^*$.

A terceira forma para obtenção de resposta é fazer uma consulta ao *SWI-Prolog*⁵. Dado um conjunto de consultas C , um conjunto de respostas R , um usuário $u_i \in U$, para $i \in N^*$, faz uma consulta $c_i \in C$, para $i \in N^*$, o interpretador AIML faz uma consulta $c_k \in C$, para $k \in N^*$ ao *SWI-Prolog* e o AC retorna uma resposta $r_i \in R$, para $i \in N^*$.

Figura 4 - Definição do AC proposto



Fonte: Elaborado pelo autor

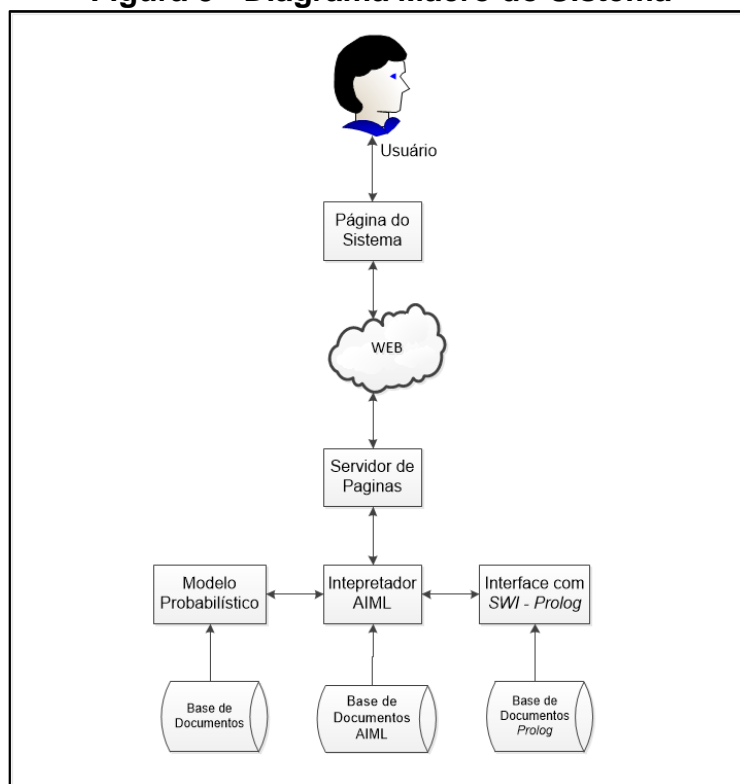
⁵ *SWI-Prolog* versão 7.2.3 por Jan Wielemaker (jan@swi-prolog.org). Copyright (c) 1990-2015 University of Amsterdam, Amsterdam.

Como primeiro passo foi construído um módulo de interpretação de linguagem de marcação AIML. O módulo assume a função de agente de conversação. Com a conclusão do interpretador AIML, teve início a programação do módulo que faz interface com o programa *SWI-Prolog*, que viabilizou a consulta a bases de dados construídas na linguagem *Prolog*. Visando as possibilidades que a linguagem *Prolog* pode fornecer, este módulo tem por objetivo ser o meio de comunicação entre o interpretador AIML e o *SWI-Prolog*. O terceiro componente construído é o que faz uso de modelos clássicos de RI para busca de documentos relevantes.

Os componentes do arcabouço assim como mostrado no diagrama da Figura 5, são:

- Interface com o Usuário;
- Servidor de páginas ASP .NET⁶;
- Interpretador AIML;
- Módulo de interface com *SWI-Prolog*;
- Modelo de RI (MPC e BM25);
- Bases de documentos.

Figura 5 - Diagrama Macro do Sistema



Fonte: Elaborado pelo autor

⁶ © 2016 Microsoft Corporation.

Ainda na Figura 5, o usuário faz interação com o sistema por intermédio da interface identificada no diagrama como “Página do Sistema”, as páginas são geradas pelo servidor de páginas ASP (*Active Server Pages*) que por sua vez envia a entrada de texto para o núcleo do sistema. O núcleo do sistema é formado pelo interpretador AIML que carrega e processa os documentos AIML e faz solicitações ao modelo probabilístico e a interface com o programa *SWI-Prolog*. Ainda na Figura 5, as setas bidirecionais indicam onde o fluxo de dados ocorre em dois sentidos e as unidirecionais onde é em sentido único.

3.3 Interpretador AIML

O interpretador AIML é a parte fundamental desse sistema, tem o papel central no funcionamento do AC proposto e por isso é o único módulo que deve estar sempre habilitado e funcionando. É o interpretador AIML faz comunicação com os outros componentes principais do programa. Caso algum problema ocorra durante o funcionamento ou quando é inicializado, o sistema interrompe ou não inicia o funcionamento do AC.

O interpretador AIML é o componente do AC responsável por procurar respostas nos arquivos do padrão AIML e por fazer solicitações aos módulos probabilísticos e de interface com *SWI-Prolog*. É composto por:

- Autômato Finito Determinístico (AFD);
- Coleção de marcações;
- Mecanismo de expressões regulares que identifica ações necessárias e solicita consultas nas bases de dados.

O interpretador AIML é capaz de processar as marcações definidas e listadas no Apêndice A. O conjunto original de marcações possui 49 definições, mas tem se como pressuposto que nem todas são necessárias para que a geração de arquivos em AIML seja eficaz o suficiente para o AC projetado e desenvolvido. Além das marcações do sistema, é previsto que marcações não aceitas sejam encontradas nos arquivos. Nesse caso, as marcações e o conteúdo destas são excluídos.

Identificou-se a necessidade de se criar algumas marcações especiais, **<mprob>** para solicitação de documentos relevantes ao modelo probabilístico e **<prolog>** para comunicação com o *SWI-Prolog*. Para que o interpretador AIML faça

solicitações ao modelo probabilístico e ao *SWI-Prolog* é necessário que estas solicitações sejam permitidas na inicialização do AC.

O sistema só pode enviar uma resposta por vez para o usuário. Os modelos probabilísticos MPC e BM25 podem gerar uma lista com vários documentos. Quando o interpretador AIML recebe uma lista de documentos, a relevância do primeiro documento é comparada com a do segundo, se a relevância do segundo for menor, o conteúdo do primeiro é mostrado. Quando o primeiro documento tem a mesma relevância que o segundo, o programa exibe um deles aleatoriamente. Esta comparação acontece com toda a coleção de documentos que estão no topo da lista.

3.3.1 Autômato finito determinístico

Um AFD é uma quintupla, sendo Q um conjunto de estados finito, Σ o alfabeto de entradas possíveis, $\delta: Q \times \Sigma \rightarrow Q$ uma função de transição de estados, $q_0 \in Q$ o estado inicial e $F \subseteq Q$ o conjunto de estados finais, Vieira (2004).

Sendo $Q = \{E_0, E_1, E_2, E_3, E_4\}$, $q_0 = E_0$, $F = \{E_0\}$ e Σ igual ao conjunto de símbolos disponíveis, as funções de transição do autômato do agente seriam:

$$\delta(E_0, <) = E_1,$$

$$\delta(E_0, \Sigma - \{<\}) = E_0,$$

$$\delta(E_1, >) = E_2,$$

$$\delta(E_1, /) = E_3,$$

$$\delta(E_1, \Sigma - \{>, /\}) = E_1,$$

$$\delta(E_2, <) = E_1,$$

$$\delta(E_2, \Sigma - \{<\}) = E_2,$$

$$\delta(E_3, >) = E_0,$$

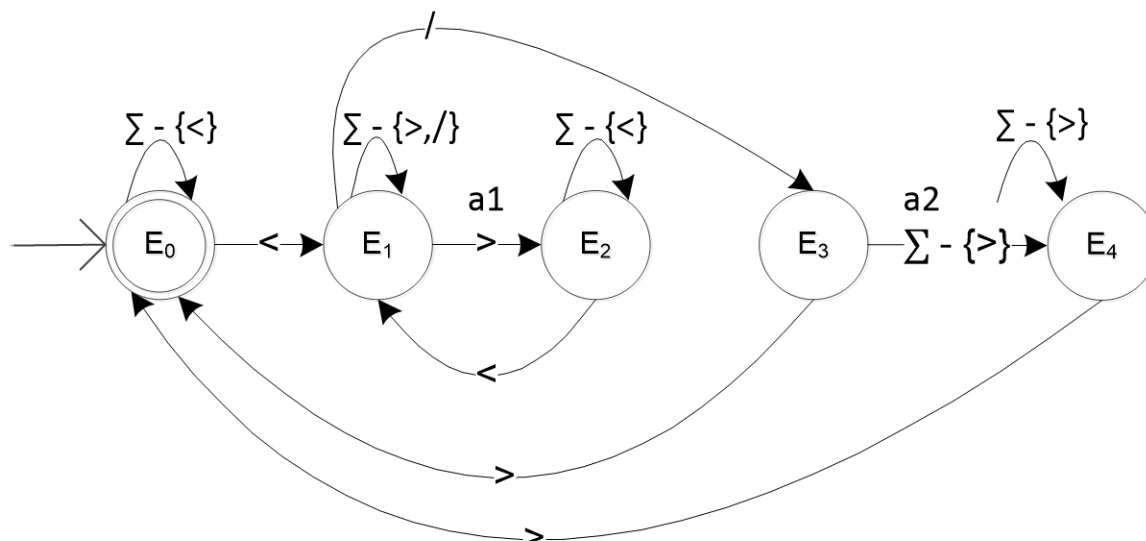
$$\delta(E_3, \Sigma) = E_4,$$

$$\delta(E_4, >) = E_0,$$

$$\delta(E_4, \Sigma - \{>\}) = E_4,$$

A Figura 6 apresenta o AFD de onde foi possível obter as funções de transição.

Figura 6 - Autômato Finito Determinístico



Fonte: Elaborado pelo autor

O Quadro 2 apresenta as transições feitas a partir do AFD da Figura 6. O número da coluna de próxima transição identifica para qual linha o programa deverá continuar o processamento. A1 e A2 são operações (ações semânticas) que o programa deverá atender. A1 indica internamente que uma marcação foi aberta e A2 que alguma foi fechada.

Quadro 2 - Autômato Finito

Linha	Estado Atual	Símbolo Lido	Próximo Estado	Próxima Transição	Ação Semântica
0	E ₀	<	E ₁	2	Nenhuma
1	E ₀	Σ	E ₀	0	Nenhuma
2	E ₁	>	E ₂	5	A1
3	E ₁	/	E ₃	7	Nenhuma
4	E ₁	Σ	E ₁	2	Nenhuma
5	E ₂	<	E ₁	2	Nenhuma
6	E ₂	Σ	E ₂	5	Nenhuma
7	E ₃	>	E ₀	0	Nenhuma
8	E ₃	Σ	E ₄	9	A2
9	E ₄	>	E ₀	0	Nenhuma
10	E ₄	Σ	E ₄	9	Nenhuma

Fonte: Baseado no modelo proposto para construção de compiladores de Setzer e Homem de Melo (1983)

O interpretador AIML utiliza o AFD em conjunto com expressões regulares para que seja possível determinar a ordem correta de processamento de marcações. Uma expressão regular (ER) é uma forma sequencial de definição de uma linguagem regular, Morais (2004). Como exemplo de ER tem-se $(A + B^*)$. Esta ER denota as sequências AB, ABB e AB BB e quaisquer outras iniciadas com A e seguidas de um ou mais B.

Pode se notar que os símbolos de entrada ilustrados na Figura 6 são diferentes dos símbolos do Quadro 2. Como o processamento utilizando os passos do Quadro 2 é sequencial, é explícita a precedência de caracteres. Por exemplo, para fazer a transição do estado E_3 para o estado E_4 é necessária a entrada de qualquer símbolo diferente de ">". Como a estrutura ilustrada pelo Quadro 2 já testou se o caractere é igual a ">", a entrada fará o casamento qualquer que seja o símbolo.

Como existe uma verificação pelo AFD. Entradas com erros não aparecem, então nenhum tratamento especial é preciso ser feito a fim de que sejam aceitas ou recusadas.

3.3.2 Base de arquivos AIML

Parte da base de dados do AC é formada pelos arquivos AIML. A Figura 7 ilustra a organização definida para esta base de dados. A estrutura foi dessa forma planejada para facilitar a organização de categorias, temas, assuntos e tópicos. Para o interpretador AIML esta organização não faz diferença, os arquivos são identificados e o conteúdo carregado como se pertencesse a um documento apenas.

Figura 7 - Diretório AIML

 Exemplos	16/03/2016 15:22	Pasta de arquivos	
 Gerais	06/05/2016 02:57	Pasta de arquivos	
 Materias	06/05/2016 02:31	Pasta de arquivos	
 aiml.xml	26/03/2016 03:07	Arquivo XML	86 KB
 aiml2.xml	01/05/2016 23:07	Arquivo XML	7 KB
 aiml4.xml	28/12/2015 00:09	Arquivo XML	149 KB
 aiml5.xml	01/01/2016 03:18	Arquivo XML	1 KB
 aiml22.xml	01/05/2016 23:09	Arquivo XML	1 KB
 aimlASK.xml	06/01/2016 00:31	Arquivo XML	2 KB
 aimlTesteDoSistema.xml	23/12/2015 00:13	Arquivo XML	1 KB
 personalidade.xml	06/05/2016 02:32	Arquivo XML	8 KB

Fonte: Elaborado pelo autor

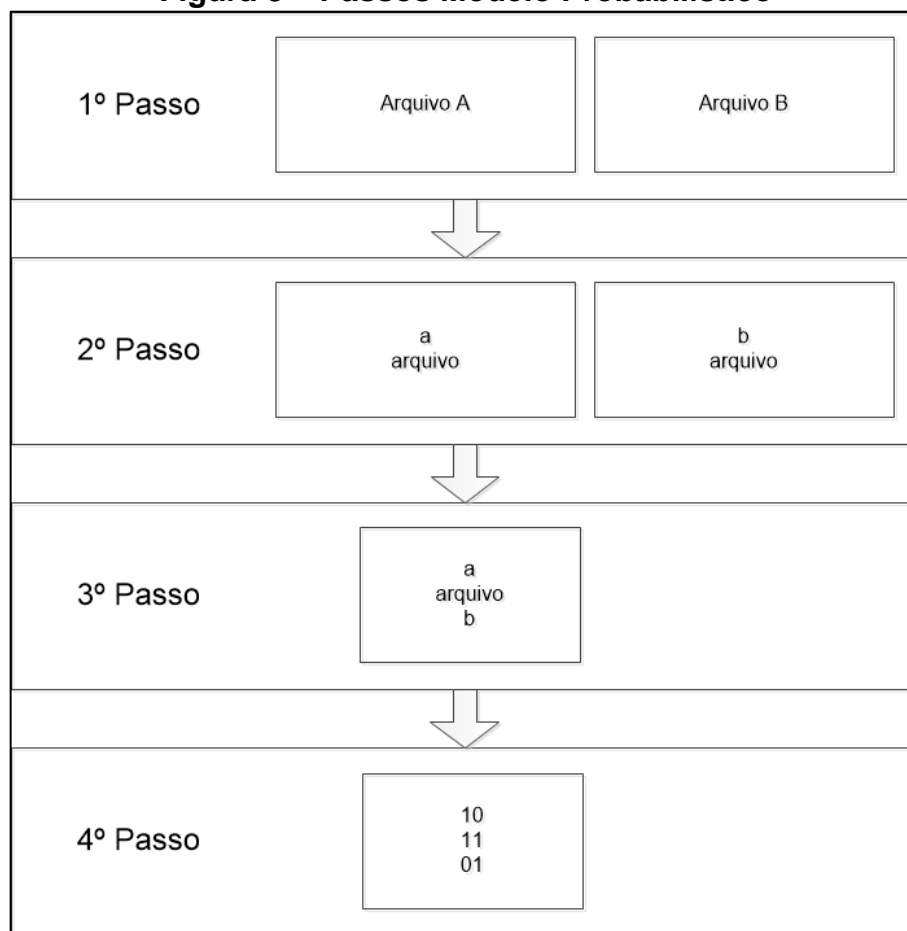
Como se observa na Figura 7, os documentos estão com extensão XML, isso porque arquivos AIML nada mais são do que arquivos XML que seguem o padrão de marcações AIML. Cada documento XML da base deve estar de acordo com algum arquivo de definição de tipo documento (extensão .DTD). Um arquivo de definição de documento define as regras para construção de documentos XML. Esse arquivo de definição pode estar em qualquer diretório, desde que a associação definida no XML esteja com o caminho correto.

Os arquivos AIML são armazenados em uma árvore para que a pesquisa de padrões possa ser feita.

3.4 Modelo Probabilístico Clássico de RI

O MPC é a parte do sistema responsável pelo tratamento e processamento de texto, pela construção de estruturas e por cálculos de acordo com a teoria do modelo probabilístico desenvolvido por *Robertson e Jones* (1976).

Os arquivos utilizados na recuperação utilizando o MPC não necessitaram de uma estruturação como no padrão AIML, dessa forma a alimentação da base de dados foi mais rápida. Entretanto, as respostas obtidas com o MPC nem sempre foram relevantes.

Figura 8 – Passos Modelo Probabilístico

Fonte: Elaborado pelo autor

A Figura 8 ilustra os passos utilizados para processar os documentos utilizados no MPC, nela um documento com o texto “Arquivo A” e outro com o texto “Arquivo B” passam por etapas, ou passos, que tornam a recuperação possível. O primeiro passo é a obtenção e seleção dos documentos da base; o segundo é formado pela normalização do texto e pela criação dos documentos de visão lógica (coleção de termos sem repetição); o terceiro passo é o processamento das visões lógicas, que gera um único documento de termos únicos, que seria o vocabulário da coleção; o quarto e último passo é a criação dos vetores binários para cada documento, cada vetor binário indica a presença (1) ou a ausência (0) dos termos do vocabulário em cada documento.

Um vetor é uma estrutura de dados composta de dados de mesmo tipo alocados sequencialmente (SILVA, 2012).










Deve-se atentar que para o funcionamento correto do conjunto formado por Interpretador e MPC, as consultas devem ser habilitadas na inicialização do AC. É importante dizer também que este componente tem uma coleção de vetores

binários que devem ser carregados antes de qualquer consulta. Esse carregamento é feito quando o AC é inicializado.

A Figura 9 mostra os documentos e diretórios do MPC. O diretório do MPC é organizado da seguinte forma:

- **“DOCUMENTOS”**: guarda todos os documentos que serão processados. Para fins de organização, estes documentos podem estar contidos em subpastas;
- **“COLECAO VETORES BINARIOS”**: é o diretório onde os vetores binários dos documentos são escritos;
- **“Vetores binarios.txt”**: é o documento em que todos os vetores binários são agrupados;
- **“VISA LOGICA”**: armazena os documentos de visão lógica de cada documento;
- **“Vocabulario.txt”**: contém o vocabulário de toda a coleção.

Figura 9 - Diretório do Modelo Probabilístico

 COLECAO BM25	06/04/2016 10:44	Pasta de arquivos
 COLECAO VETORES BINARIOS	14/03/2016 02:23	Pasta de arquivos
 DOCUMENTOS	11/03/2016 00:53	Pasta de arquivos
 VISA LOGICA	11/03/2016 01:12	Pasta de arquivos
 Tamanho dos documentos.txt	21/03/2016 02:51	Documento de Texto
 Tamanho medio dos documentos.txt	21/03/2016 02:51	Documento de Texto
 Vetores binarios.txt	21/03/2016 23:34	Documento de Texto
 Vetores BM25.txt	06/04/2016 11:31	Documento de Texto
 Vocabulario.txt	11/03/2016 01:24	Documento de Texto

Fonte: Elaborado pelo autor

A coleção de documentos pode ser muito grande, diante disso é possível presumir que o tempo necessário para processamento e gravação em memória secundária vai ser diretamente proporcional. Para tratar, verificar e escrever os 51830 documentos que formam a base inicial de documentos foram necessárias 20 horas. O tempo para trabalhar todos os documentos não precisa ser contínuo, o AC permite que o processamento seja interrompido e retomado posteriormente. Esta interrupção é possível porque cada vetor binário é escrito na memória secundária logo após ser terminado.

Quando uma consulta é enviada para o MPC, uma lista de documentos ordenados pela relevância de conteúdo é feita aplicando-se a equação de similaridade do modelo probabilístico com os valores obtidos dos vetores binários.

A Figura 10 apresenta como a consulta a um dos modelos probabilísticos (MPC ou BM25) pode ser feita. No interpretador AIML, quando o padrão da categoria apresentada faz casamento com a entrada do usuário, é enviado o conteúdo entre “<mprob>” e “</mprob>” para o modelo probabilístico.

Figura 10 - Marcações para consulta em modelo probabilístico

```
<category>
  <pattern>*</pattern>
  <template>
    <mprob>
      <star/>
    </mprob>
    <condition name="infomp" value="false">
      <srai>relevante</srai>
    </condition>
  </template>
</category>
```

Fonte: Elaborado pelo autor

A marcação condicional (<condition>) da Figura 10 solicita a verificação que indica se algum documento relevante foi obtido e caso nenhum seja recuperado a marcação de busca recursiva (<srai>) busca alguma outra resposta.

Ainda na Figura 10, observa-se que a marcação <mprob> não indica se o modelo probabilístico para consulta é o MPC ou o BM25. Esta escolha de modelo é feita durante a inicialização do AC.

3.5 BM25

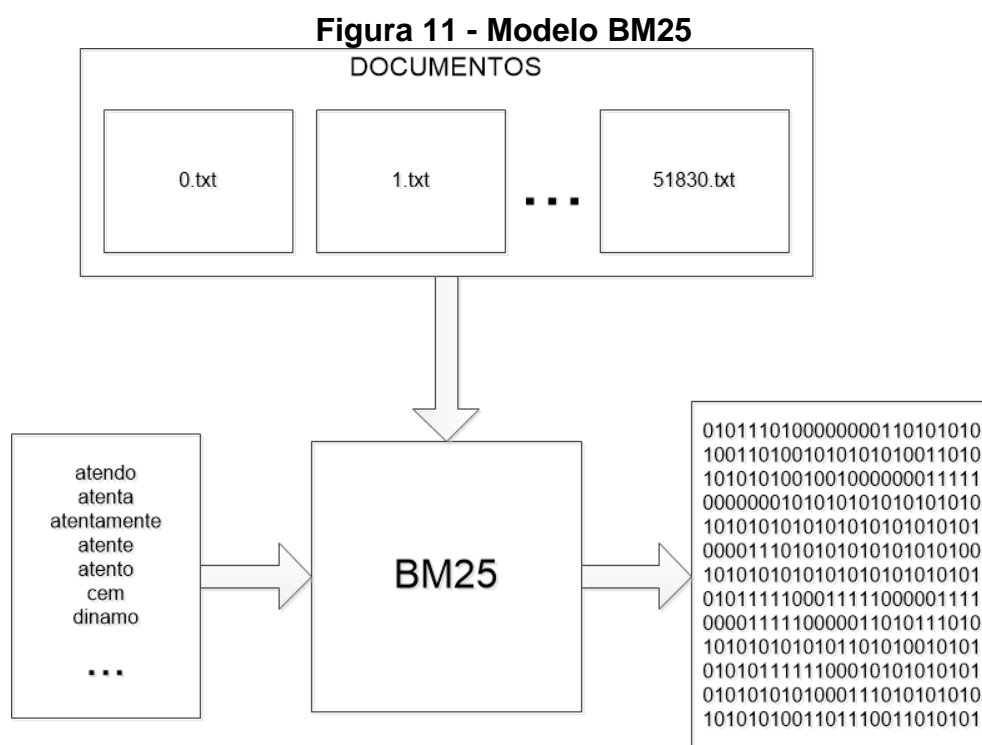
O modelo BM25 utiliza três princípios: frequência inversa de documentos, frequência dos termos e normalização pelo tamanho dos documentos (BAEZA-YATES; RIBEIRO-NETO, 2013). Os documentos do BM25 são organizados da seguinte forma:

- “**DOCUMENTOS**”: armazena todos os documentos que serão processados;
- “**VISAO LOGICA**”: armazena os documentos de visão lógica de cada documento;
- “**Vocabulario.txt**”: contém o vocabulário de toda a coleção.

- **“COLECAO BM25”**: é o diretório onde os vetores dos documentos são armazenados;
- **“Tamanho dos documentos.txt”**: armazena a quantidade de caracteres de cada documento. A quantidade de caracteres é o tamanho de cada um deles;
- **“Tamanho medio dos documentos.txt”**: tamanho médio dos documentos calculado com base no número de caracteres;
- **“Vetores BM25.txt”**: contém o conjunto de vetores do BM25 agrupados.

Os diretórios **“DOCUMENTOS”** e **“VISA0 LOGICA”** e o documento **“Vocabulario.txt”** são compartilhados com o MPC.

A Figura 11 ilustra o processo de geração dos vetores do BM25. Primeiro os documentos e o vocabulário são carregados na memória principal do computador. Se um termo do vocabulário estiver presente em um documento, o passo seguinte é a contagem das ocorrências desse termo, se não estiver presente nenhum processamento é feito. A construção dos vetores do BM25 também pode ser interrompida assim como na construção dos vetores binários do MPC.



Fonte: Elaborado pelo autor

Quando uma consulta é enviada para o BM25, uma lista de documentos ordenados em ordem decrescente de relevância de conteúdo é feita aplicando-se a equação de similaridade do BM25.

A equação de similaridade do modelo BM25 tem como parâmetro a frequência que um termo aparece em um documento. Tem se como decisão de projeto limitar essa frequência entre o intervalo de zero a quinze. Assim, apenas quatro *bits* são necessários para armazenar cada frequência de cada termo e nenhum caractere de controle precisa ser colocado no arquivo que armazena esses dados. Qualquer frequência que ultrapasse esse limite é considerada como quinze.

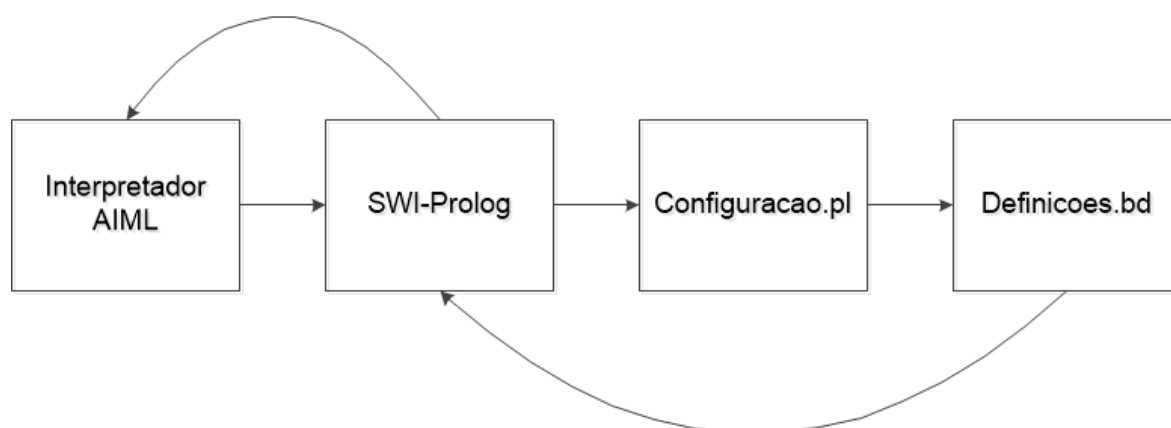
Para à decisão de considerar esse intervalo analisaram-se documentos utilizados. Foi observado que menos memória principal do computador seria necessária e que nenhum prejuízo para a computação de similaridade iria ocorrer. Processar os arquivos previamente para obter a frequência máxima possível também seria uma opção, entretanto isso geraria uma carga extra no sistema.

3.6 Integração do interpretador AIML com *SWI-Prolog*

A linguagem *Prolog* foi utilizada para criação de definições. Esta linguagem é adequada a problemas onde é necessário representar algum tipo de conhecimento (BARBOSA; CUNHA, 2006).

A Figura 12 apresenta como o sistema se comunica com o *SWI-Prolog*. Primeiro o interpretador AIML faz uma solicitação ao programa, o programa então carrega um arquivo de configurações (que contém informações sobre o documento de cláusulas em *Prolog*) e o documento com as cláusulas. Após esses passos o *SWI-Prolog* faz as operações necessárias e devolve a resposta para o AC.

Figura 12 – Interpretador AIML e *SWI-Prolog*

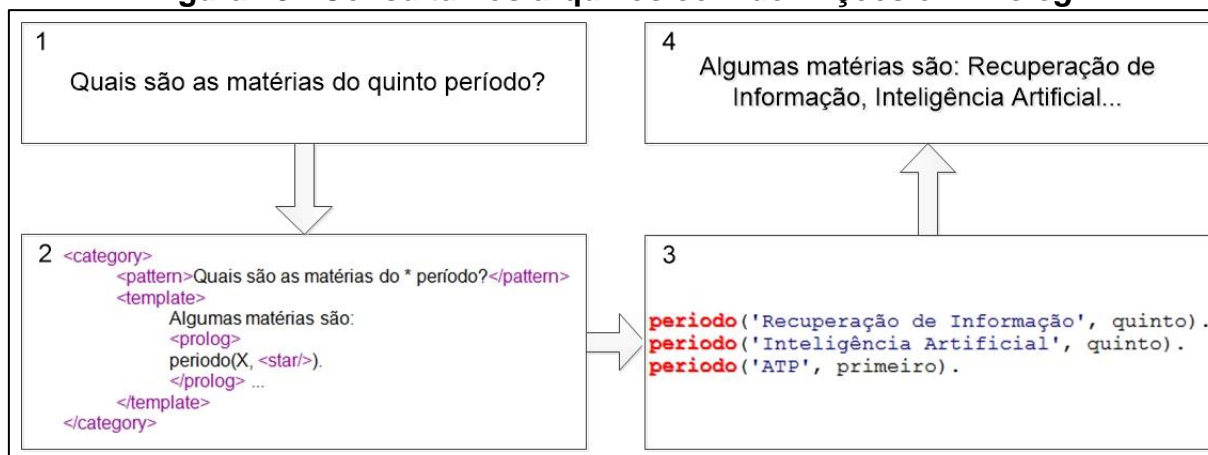


Fonte: Elaborado pelo autor

A Figura 13 mostra uma consulta utilizando *Prolog*. O usuário escreve uma pergunta (1), o sistema busca o casamento para essa entrada e encontra um padrão

que solicita consultar os arquivos *Prolog* (2), as cláusulas são encontradas no documento *Prolog* (3), e por fim, a resposta é enviada para o usuário (4).

Figura 13 - Consulta nos arquivos com definições em *Prolog*



Fonte: Elaborado pelo autor

O programa *SWI-Prolog* compatível com o sistema é o de versão 7.2.2 de 32 *bits*. Caso o programa não esteja instalado ou uma versão incompatível esteja no computador, o sistema ignora todas as chamadas destinadas a cláusulas *Prolog*. Na utilização do *SWI-Prolog* foi encontrado um problema, não é possível fazer consultas em paralelo, ou seja, apenas uma solicitação ao programa externo pode ser feita de cada vez. Dessa forma, uma fila de solicitações é formada e cada uma é respondida por vez.

Para possibilitar a comunicação entre o *SWI-Prolog* e a aplicação desenvolvida em *C#*, utilizou-se a biblioteca *SwiPLCs*⁷.

3.7 Comparação entre as três abordagens

As três abordagens utilizadas para recuperação de respostas possuem pontos positivos e negativos. O Quadro 3 apresenta as vantagens e desvantagens de cada uma das três.

⁷ Uwe Lesta, SBS-Softwares

Quadro 3 - Comparação entre abordagens

Abordagem	Vantagens	Desvantagens
AIML	Flexível para construção de padrões.	Pouco flexível para casamento de texto.
	Segue os padrões XML.	Sintaxe rígida e por isso é necessário dedicar mais tempo na geração de documentos.
	Linguagem utilizada por uma grande comunidade e por isso tem uma documentação bem desenvolvida e muitos exemplos de aplicações.	
Modelos Probabilísticos	Facilidade de construção da base de dados.	Alto custo computacional.
		Respostas não relevantes em algumas consultas.
Prolog	Flexibilidade de uma linguagem de programação.	Necessidade de conhecimento sobre a linguagem.
	Linguagem bem conhecida.	Apenas uma consulta é feita por vez.

Fonte: Elaborado pelo autor

3.8 Casamento de Padrões e Tratamento de Texto

Para aumentar a probabilidade de casamento de texto algumas técnicas podem ser utilizadas, como o algoritmo de programação dinâmica de *Sellers*, transformação léxica, substituição de termos e eliminação de *stopwords*.

3.8.1 Algoritmo de *Sellers*

O algoritmo de programação dinâmica de *Sellers*⁸ é usado para casamento aproximado de texto (QUINTÃO 2006). Não raro, a correspondência de uma

⁸ Algoritmo que compara e tenta fazer casamento entre duas sequências de caracteres em tempo polinomial atribuído a *Sellers* (1974).

sequência pequena de texto dentro de uma sequência maior pode ser encontrada, isso não caracteriza um erro, mas para a comparação feita no algoritmo do AC poderia causar problemas (a comparação no sistema é feita termo a termo). Por exemplo, a palavra "ia" poderia ser encontrada na palavra "tecnologia". Para que esse problema não ocorra, o algoritmo verifica o tamanho das duas palavras a serem comparadas e tenta encontrar a maior palavra dentro da menor, assim dois termos serão correspondentes apenas se a diferença entre os dois respeitar o limite de erros permitidos.

A Figura 14 mostra a possibilidade de encontrar a palavra "tema" (termo menor) dentro da palavra "sistema" (termo maior). Nessa comparação o termo foi encontrado com nenhum erro.

Figura 14 - Comparando termo menor com termo maior

tema	→	sistemas	
sistem <u>a</u> s			Erros = 0

Fonte: Elaborado pelo autor

A Figura 15 mostra a comparação entre a palavra "sistemas" (termo maior) e a palavra "tema" (termo menor). Nessa comparação quatro erros foram encontrados. Se o limite de erros permitidos fosse menor que quatro, o casamento entre termos iria ocorrer, caso contrário não.

Figura 15 - Comparando termo maior com termo menor

sistemas	→	tema	
sistem <u>a</u> s			Erros = 4

Fonte: Elaborado pelo autor

O número máximo de erros definido no AC foi de vinte por cento para palavras com cinco ou mais caracteres e de um erro para palavras com até quatro caracteres. A porcentagem está relacionada à quantidade de caracteres de um termo.

Quando se define um erro, uma sequência de caracteres A pode fazer casamento com uma sequência B com um caractere a mais, um caractere a menos, ou com um caractere trocado.

3.8.2 Transformação Léxica

Para tratar os textos digitados pelo usuário, e assim facilitar o processo de casamento de padrões, pontuações e acentos são excluídos, letras em caixa alta são substituídas pela correspondência em caixa baixa e caracteres especiais são eliminados.

Um tratamento pode ser utilizado para padronizar plurais, flexões de verbos e gêneros. Para esse tratamento, foram utilizados arquivos disponibilizados na *Web* pelo projeto *Árvore Sintá(c)tica*. O Quadro 4 mostra como esse tratamento pode ser feito.

Quadro 4 - Substituição de termos

Palavra	Substituir por
a	o
bem comportadas	bem comportado
candidatasse	candidatar
derivado	derivar
estereofonicas	estereofonico
fotografa	fotografar
urbanizacoes	urbanizacao

Fonte: Elaborado pelo autor

Projeto Floresta Sintá(c)tica é um projeto formado por especialistas em língua portuguesa, onde textos são analisados de diferentes formas e disponibilizados para uso livre.

3.8.3 Eliminação de stopwords

Alguns termos tem pouco valor semântico, como artigos, preposições e pronomes, esses termos são conhecidos como *stopwords* (BAEZA-YATES; RIBEIRO-NETO, 2013). A eliminação desses termos diminui a quantidade de memória utilizada, o tempo de processamento em operações e ainda pode evitar que documentos não relevantes sejam recuperados. Uma das formas de eliminar esses termos é excluir todas as palavras com poucos caracteres, outra é armazenar uma coleção de palavras pouco relevantes e usá-las para identificar *stopwords*.

No diretório “*ArquivosGerais*” encontram-se dois documentos, “*stopwords.txt*” e “*!stopwords.txt*”, o primeiro armazena uma coleção de *stopwords* e o segundo uma coleção de não *stopwords*. O sistema considera que palavras pequenas, de quatro caracteres, não são significativas. A coleção de termos “*!stopwords.txt*” é usada para exceções a essa regra, ou seja, para termos pequenos que são relevantes.

3.9 Experimentos

Com o propósito de conseguir informações qualitativas e quantitativas, experimentos foram relevantes. A Tabela 1 apresenta os resultados obtidos por intermédio da métrica MRR.

Nesta aplicação da métrica foi considerada que a resposta correta seria aquela contida no documento de onde foi retirada a pergunta. Os passos para aplicação da métrica MRR foram os seguintes:

1. Abertura de um documento da base de documentos e identificação da pergunta e da resposta contidas nele;
2. Digitação da pergunta identificada (sequência exata);
3. Identificação da posição do documento na lista de documento relevantes criada pelo modelo probabilístico;
4. Aplicação da métrica.

Tabela 1 - Aplicação da métrica MRR

Consulta	Modelo Probabilístico			BM25		
	S _{correto}	RR	Resposta equivalente ou exata	S _{correto}	RR	Resposta equivalente ou exata
1	1	1,000	Sim	1	1,000	Sim
2	1	1,000	Sim	1	1,000	Sim
3	1	1,000	Sim	1	1,000	Sim
4	1	1,000	Sim	1	1,000	Sim
5	27	0,000	Sim	47678	0,000	Sim
6	1	1,000	Sim	1	1,000	Sim
7	3	0,333	Sim	12	0,000	Sim
8	4	0,250	Não	9	0,111	Sim
9	1	1,000	Sim	2	0,500	Não
10	1	1,000	Sim	6	0,167	Sim
11	1	1,000	Sim	1	1,000	Sim
12	1	1,000	Sim	1	1,000	Sim
13	1	1,000	Sim	1	1,000	Sim
14	2	0,500	Sim	20	0,000	Sim
15	1	1,000	Sim	1	1,000	Sim
16	1	1,000	Sim	1	1,000	Sim
17	2	0,500	Sim	1	1,000	Sim
18	1	1,000	Sim	1	1,000	Sim
19	30	0,000	Não	28	0,000	Sim
20	13	0,000	Sim	27	0,000	Sim
MRR	0,729			0,639		
S _h	10					
Desvio Padrão	0,390			0,454		

Fonte: Elaborado pelo autor

Em alguns casos a resposta retornada pelo AC veio de um documento relevante diferente daquele de onde a pergunta foi extraída, nesses casos, a resposta não foi considerada como correta. Respostas assim foram identificadas como equivalentes na coluna “Resposta equivalente ou exata” da Tabela 1. Entretanto, para o usuário essas respostas seriam corretas independentemente se vieram de um documento ou de outro. Por isso, a métrica também foi aplicada levando se em conta que a resposta correta seria aquela que respondia uma pergunta independentemente do documento origem. A Tabela 2 exhibe os valores obtidos com essa segunda abordagem.

Tabela 2 - Segunda abordagem na Aplicação da métrica MRR

Consulta	Modelo Probabilístico		BM25	
	S_{correto}	RR	S_{correto}	RR
1	1	1,000	1	1,000
2	1	1,000	1	1,000
3	1	1,000	1	1,000
4	1	1,000	1	1,000
5	1	1,000	1	1,000
6	1	1,000	1	1,000
7	1	1,000	1	1,000
8	4	0,250	1	1,000
9	1	1,000	2	0,500
10	1	1,000	1	1,000
11	1	1,000	1	1,000
12	1	1,000	1	1,000
13	1	1,000	1	1,000
14	1	1,000	1	1,000
15	1	1,000	1	1,000
16	1	1,000	1	1,000
17	1	1,000	1	1,000
18	1	1,000	1	1,000
19	30	0,000	1	1,000
20	1	1,000	1	1,000
MRR	0,913		0,975	
S_h	10			
Desvio Padrão	0,265		0,109	

Fonte: Elaborado pelo autor

Na Tabela 2 observa-se que os resultados obtidos dessa segunda forma foram melhores que os obtidos na primeira aplicação da métrica. O desvio padrão foi calculado na Tabela 1 e na Tabela 2 com base nas colunas RR.

Durantes os testes, observou-se que os modelos probabilísticos possuem características que podem levar a escolha de um ou de outro com base nos recursos computacionais disponíveis. O Quadro 5 apresenta o comparativo entre os dois modelos probabilísticos testados, os modelos MPC e BM25.

Quadro 5 - Comparação entre modelos probabilísticos

Parâmetro	MPC	BM25
Velocidade	Mais rápido ⁹	Mais lento ¹⁰
Memória	Uso elevado de memória principal.	Uso quatro vezes maior de memória principal.
Recuperação de documentos¹¹	Bons resultados.	Resultados melhores.

Fonte: Elaborado pelo autor

O BM25 apresenta resultados melhores que o MPC, entretanto ele exige mais recursos computacionais, tanto de memória primária quanto de processamento. A escolha do modelo a ser utilizado depende da prioridade dada, que seriam melhor qualidade na recuperação de documentos ou o menor uso de recursos computacionais.

3.10 Sistema *Web* de Agente de Conversação

Um AC que pode ser acessado de qualquer lugar e por intermédio de qualquer dispositivo pode ser mais desejável do que um restrito apenas a alguma máquina. Uma boa opção é a disponibilização de um AC em um sítio na *Web*, assim como feito com robô Ed (CONPET, 2016), agente *Fake Captain Kirk* (PANDORABOTS, 2015) e ALICE (WALLACE, 2005).

Tendo como incentivo os benefícios que as tecnologias *Web* podem fornecer a um AC, empregaram-se os módulos desenvolvidos em uma aplicação *Web*. Dessa forma, o sistema pode ser executado em um computador servidor e ser utilizado por meio de qualquer plataforma capaz de interpretar uma página comum da Internet.

No diagrama de casos de uso exibido na Figura 16 tem-se a apresentação dos três atores da aplicação *Web*, o administrador, o usuário e o AC. O administrador pode responder e fazer perguntas para o AC e para os usuários e ainda configurar parâmetros do sistema. Usuário pode responder e fazer perguntas para o AC e para o administrador. Já o AC pode responder e fazer perguntas para o

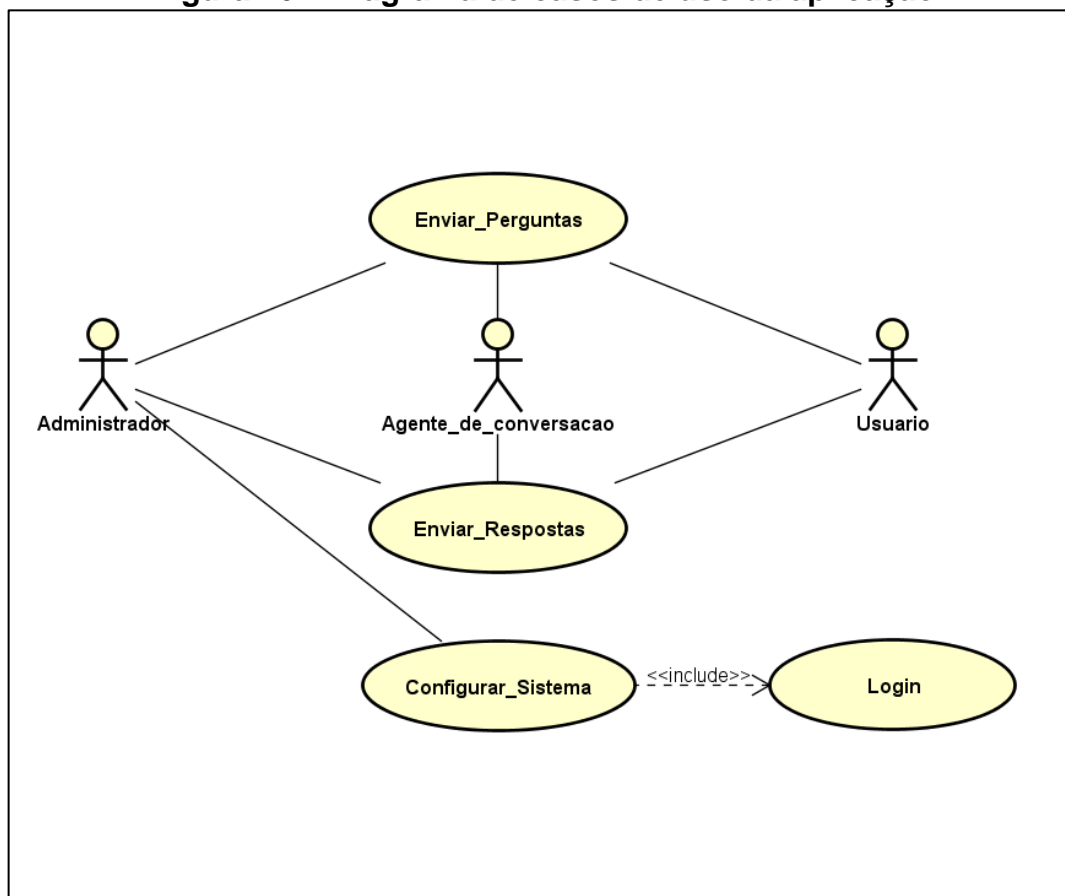
⁹ Média = 0,1058333 seg. por consulta com desvio padrão igual a 0,0349582

¹⁰ Média = 0,1101002 seg. por consulta com desvio padrão igual a 0,0296586

¹¹ Testes aplicados com a métrica *Mean Reciprocal Rank*

administrador e para o usuário. Um caso de uso relata a história do ponto de vista de um usuário final, de onde é possível determinar as funcionalidades e características de um software em relação à visão do usuário (PRESSMAN, 2011).

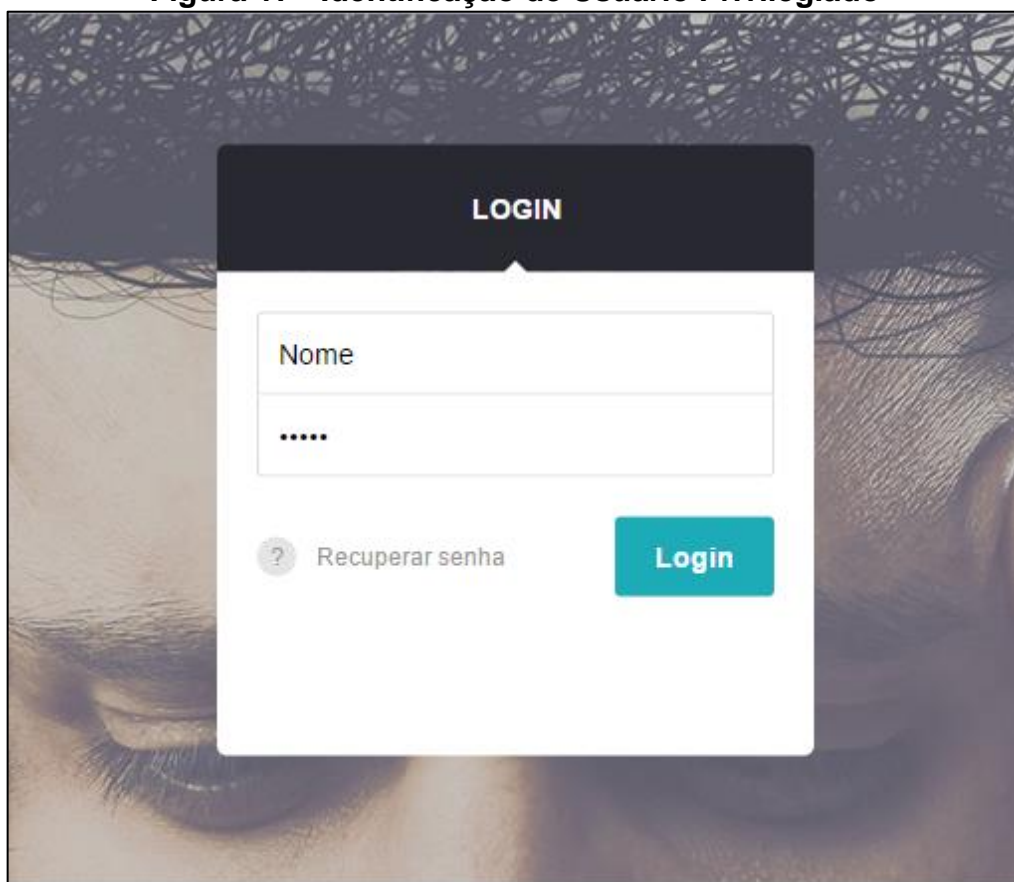
Figura 16 – Diagrama de casos de uso da aplicação



Fonte: Elaborado pelo autor

O administrador é responsável pela configuração do sistema, para isso é necessário que ele faça a autenticação no sítio (*Login*). Nessa configuração o administrador escolhe se algum modelo probabilístico será utilizado, se *Prolog* fará parte da base de documentos e qual o caminho raiz dos diretórios do sistema.

A página de autenticação é mostrada na Figura 17. Ao ser autenticado administrador, pode configurar o sistema. Caso um algum usuário tente usar a aplicação antes da configuração, ele é redirecionado para uma página que exibe uma mensagem de aviso.

Figura 17 - Identificação de Usuário Privilegiado

Fonte: Elaborado pelo autor

O administrador após fazer autenticação deve configurar alguns parâmetros do sistema. O primeiro deles é informar o caminho raiz de onde a aplicação deve procurar os diretórios. O segundo parâmetro refere-se à utilização dos arquivos em *Prolog*, para que as definições nessa linguagem sejam utilizadas o campo "*Habilitar Prolog*" deve estar marcado. O terceiro e último parâmetro é relativo à escolha do modelo probabilístico a ser consultado, o AC pode utilizar o MPC, o BM25 ou nenhum deles. Finalmente, o botão para inicialização do sistema deve ser acionado.

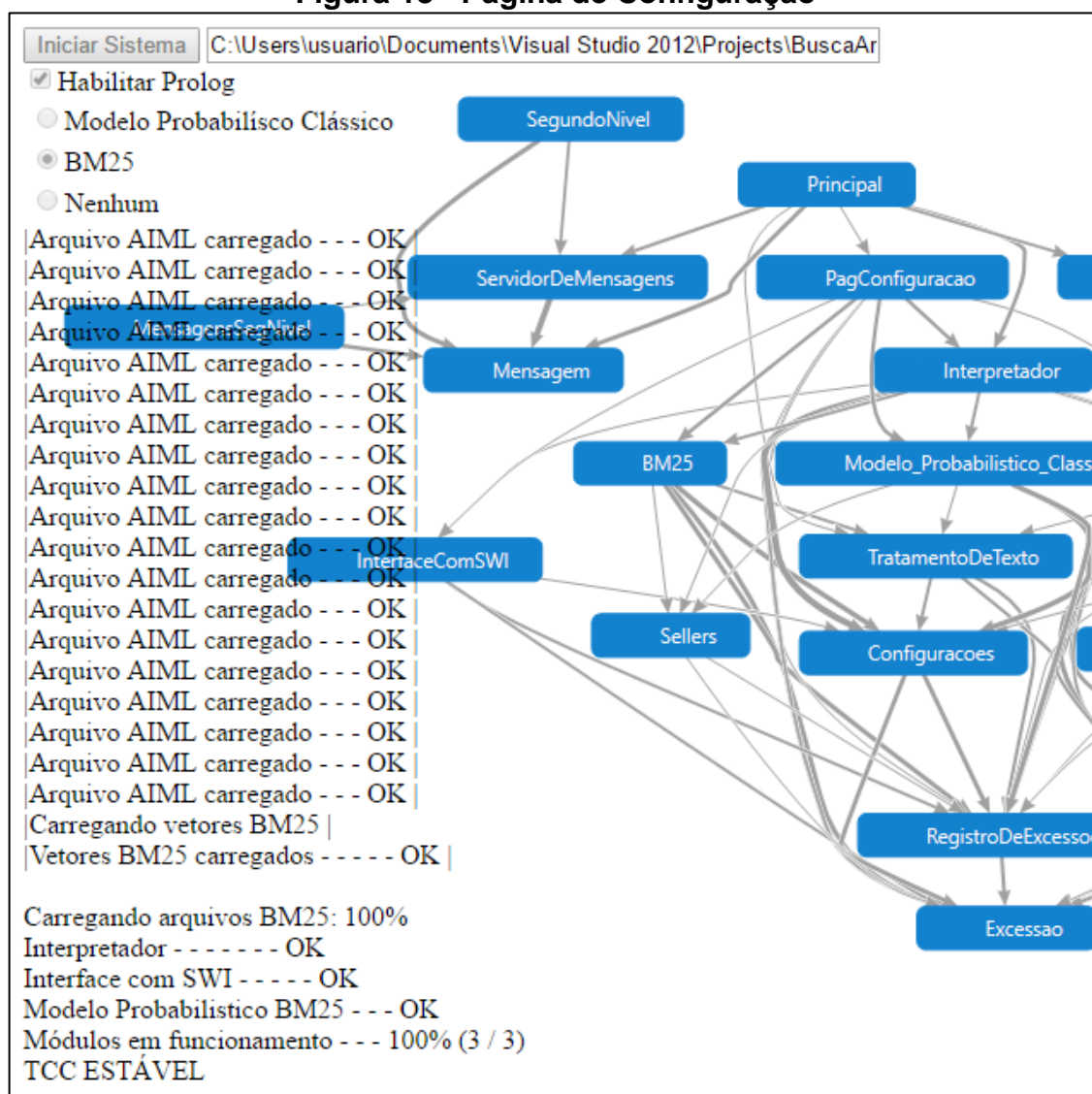
Apenas um modelo probabilístico pode ser escolhido por vez. Se o MPC e o BM25 funcionassem ao mesmo tempo não seria possível determinar de qual modelo a resposta seria mais adequada.

Com o acionamento do botão, uma sequência de informações do sistema é iniciada. Essa sequência apresenta informações de inicialização e exceções originadas ao longo da utilização da aplicação. As exceções que podem ser exibidas são relativas a erros de índices nos arquivos AIML e a diretórios e arquivos não encontrados.

Deve se atentar para as definições em AIML que solicitam consultas aos modelos probabilísticos e aos arquivos em *Prolog*. Os módulos podem estar desabilitados. Neste caso, indica-se o uso de marcações condicionais (**<if>**, **<else>** e **<condition>**) para contornar possíveis problemas.

A Figura 18 apresenta a página de configuração com as informações de uma inicialização onde *Prolog* e o BM25 foram habilitados. Nesse processo, todos os módulos iniciaram com sucesso, nenhum arquivo AIML estava fora dos padrões e nenhuma exceção foi identificada.

Figura 18 - Página de Configuração



Fonte: Elaborado pelo autor

Após configuração e inicialização do sistema, o AC pode ter as páginas de configuração e de segundo nível (moderação) acessadas por qualquer administrador e todas as outras páginas por qualquer categoria de usuário.

A Figura 19 apresenta a página principal da aplicação. Quando o sistema não estiver configurado ou quando a inicialização foi impossível, nenhuma página é exibida, com exceção das páginas de autenticação de configuração e de mensagens de aviso.

Figura 19 - Página Principal



Fonte: Elaborado pelo autor

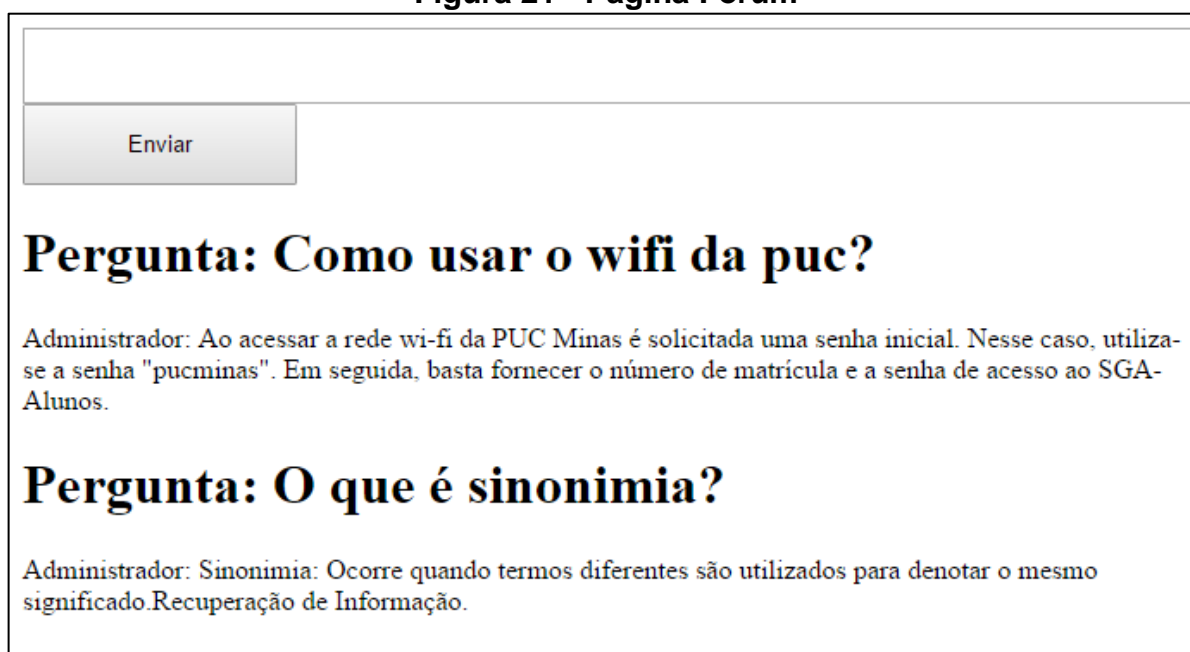
A Figura 20 apresenta a página de onde uma conversa individual pode ser mantida. Nesta página o usuário pode interagir com o AC e com algum administrador (quando uma marcação ativar o redirecionamento para a página de segundo nível/moderação). Na conversa individual todos os recursos do AC estão presentes, como o armazenamento de estados internos e de informações do usuário (nome, localidade, frases anteriores e outras).

Figura 20 – Conversa individual

Fonte: Elaborado pelo autor

A Figura 21 apresenta a página destinada a um fórum de mensagens apoiado pelo AC. Na Figura 21 duas perguntas (fonte maior) são exibidas com as respectivas respostas (fonte menor). Diferentemente da página de conversa individual, a página de fórum não armazena variáveis de estado e destina todas as mensagens para um administrador/moderador, ou seja, todas as respostas são avaliadas por um humano antes de serem respondidas.

Figura 21 - Página Fórum



Enviar

Pergunta: Como usar o wifi da puc?

Administrador: Ao acessar a rede wi-fi da PUC Minas é solicitada uma senha inicial. Nesse caso, utiliza-se a senha "pucminas". Em seguida, basta fornecer o número de matrícula e a senha de acesso ao SGA-Alunos.

Pergunta: O que é sinonímia?

Administrador: Sinonímia: Ocorre quando termos diferentes são utilizados para denotar o mesmo significado. Recuperação de Informação.

Fonte: Elaborado pelo autor

Um AC nem sempre apresenta respostas que façam sentido para uma pergunta. No contexto de um fórum, cada mensagem é visualizada por uma quantidade indeterminada de pessoas, por isso a atuação de um AC moderada por um humano pode ser viável. Dessa forma, erros do AC podem ser evitados por uma pessoa e ainda essa pessoa não tem o trabalho de fazer tudo sozinha.

A Figura 22 mostra a lista de mensagens não lidas pelo administrador. Cada *hiperlink* representa uma mensagem de um usuário diferente (a sequência de caracteres é formada pela identificação da sessão do usuário).

Figura 22 - Mensagens não lidas



Existem 2 mensagens não lidas

[32sb01iow423gbzv4ygwuujf](#)

[u1bjzl2mg3wvt5jyhplwhi25](#)

Fonte: Elaborado pelo autor

Ao escolher uma pergunta para ser respondida, o administrador é redirecionado para a página de criação de respostas. A Figura 23 apresenta essa página. A primeira caixa de texto é destinada para respostas criadas pelo

administrador, a segunda e editável caixa de texto é onde a sugestão do AC aparece. A frase da segunda caixa de texto é enviada quando o botão de envio é pressionado e quando a primeira caixa está vazia. Na área de mensagens recebidas, o histórico de mensagens está em cor azul e a última mensagem está em cinza. Todas as mensagens do Fórum são respondidas por intermédio da interface apresentada nesta figura. As perguntas de conversas individuais só aparecem para o administrador quando alguma marcação AIML fizer essa solicitação.

Figura 23 - Mensagem no segundo nível

The screenshot shows a chat window with a conversation history on the left and a response area on the right. The history contains several messages in blue text, including greetings, questions about the date, and a request to move to the second level. The response area on the right contains a single message in gray text explaining the concept of synonyms. At the bottom is a green button labeled 'ENVIAR'.

qccyj1vf55jjlykttgw0h4t1: Olá, meu nome é hitalo
 Administrador: Oi Hitalo. No que posso ajudar?
 qccyj1vf55jjlykttgw0h4t1: Que dia é hoje?
 Administrador: Hoje é terça-feira, 17 de maio de 2016, 13:37
 qccyj1vf55jjlykttgw0h4t1: va para o segundo nível
 Administrador: Eu vou te tranferir para uma pessoa que talvez possa te ajudar
 qccyj1vf55jjlykttgw0h4t1: O que é sinonimia?

qccyj1vf55jjlykttgw0h4t1: O que é sinonimia?

Sinonimia: Ocorre quando termos diferentes são utilizados para denotar o mesmc

ENVIAR

Fonte: Elaborado pelo autor

Armazenar informações coletadas das conversas entre os usuários e o AC pode ser uma forma de tornar a alimentação da base de arquivos mais fácil. Para este fim todas as conversas são armazenadas em documentos XML no diretório “Logs”.

4 CONCLUSÃO

Esta monografia apresentou o projeto e desenvolvimento de um arcabouço de agente de conversação aplicado ao curso de Sistemas de Informação da Pontifícia Universidade Católica de Minas Gerais. O arcabouço foi formado por três componentes principais, que são: o interpretador AIML; o módulo de Recuperação de Informação (Modelo Probabilístico Clássico e *Best Match* 25) e o módulo de interface com o software *SWI-Prolog* (linguagem *Prolog*).

Para formar uma base de documentos, realizou-se a obtenção de uma coleção de 2073 documentos provenientes da conta oficial da PUC Minas no sítio *ask.fm/pucminas*. Estes documentos geraram 51820 perguntas com respostas. Em seguida realizou-se a avaliação da capacidade do sistema recuperar essas informações, aplicando a métrica MRR. Obteve-se uma taxa de acerto de 91,3% para o Modelo Probabilístico Clássico e 97,5% para o *Best Match* 25.

4.1 Contribuições

Algumas perguntas são simples e podem ser respondidas por máquinas. Os custos para manter um sistema de resposta automática podem ser menores que os para manter um funcionário para responder dúvidas. Algumas empresas e instituições como PUC Minas, Sony¹² e Motorola¹³ disponibilizam uma coleção de perguntas respondidas na *Web* e/ou um serviço de atendimento com funcionários. Porém, percorrer uma coleção destas pode levar muito tempo e responder a questionamentos simples pode levar a sobrecarga de uma equipe.

O Projeto e Desenvolvimento do Arcabouço de Agente de Conversação aplicado ao curso de Sistemas de informação da Pontifícia Universidade Católica de Minas Gerais apresentado nesta monografia demonstra uma forma de obtenção de respostas automáticas.

A dificuldade encontrada na tentativa de abranger vários assuntos possíveis, as limitações dos recursos computacionais disponíveis e a recuperação de respostas não relevantes foram os principais problemas encontrados. Destes problemas podem ser citadas algumas observações: algumas versões funcionavam apenas em arquiteturas de 64 *bits*; em algumas ocasiões os computadores não

¹² Empresa dos ramos eletrônico, entretenimento e outros.

¹³ Empresa do ramo de eletrônicos.

suportavam os processos e a recuperação de documentos baseada na probabilidade nem sempre gera resultados corretos. Entretanto, com o crescimento da base de documentos AIML, com a adoção de estratégias para economia de memória e com aplicação de tratamento de texto (como eliminação de *stopwords*) estes problemas foram diminuídos.

O Capítulo 2 apresentou a importância dos AC, IA, SE e RI para o projeto e desenvolvimento de um arcabouço de Agente de Conversação. Também foram descritas algumas ferramentas, tais como, AIML, modelo probabilístico de RI clássico e uma variação que seria o BM25, além da métrica MRR, utilizadas para projetar e desenvolver o AC proposto.

O Capítulo 3 dedicou-se a descrição da metodologia utilizada no Projeto e Desenvolvimento do Arcabouço de Agente de Conversação aplicado ao curso de Sistemas de Informação da PUC Minas. Assim como também, a apresentação do AC disponibilizado em forma de aplicação *Web* e dos resultados experimentais.

Constata-se que o projeto e o desenvolvimento de um arcabouço de um AC não são tarefas simples. As diferentes abordagens para busca de respostas possuem vantagens e desvantagens. Durante toda a monografia os resultados, abordagens e dificuldades encontradas foram comentados. Pode-se dizer que uma base de documentos grande e atualizada é fundamental para utilização deste tipo de sistema.

4.2 Trabalhos Futuros

Seguindo a mesma linha da monografia, podem ser indicados os seguintes caminhos para trabalhos futuros:

- A construção de interfaces que alimentarão a base AIML de maneira mais eficiente;
- Incremento da base AIML e da base de conhecimentos gerais e específicos;
- Criar um mecanismo de avaliação de respostas do AC;
- Construir um sistema especialista completo em *Prolog* que faça comunicação com o Agente de Conversação;
- Realizar testes com usuários finais a fim de identificar possíveis necessidades de ajustes.

REFERÊNCIAS

- BAEZA-YATES, R; RIBEIRO, Berthier de Araújo Neto. **Recuperação de informação: conceitos e tecnologia das máquinas de busca**. 2. ed. Porto Alegre, RS: Bookman, 2013. xxiii, 590 p. ISBN 9788582600481
- BARBOSA, Alexandre A.;CUNHA, Joseluze F. **Introdução à Programação em Lógica**. Campina Grande: Universidade Federal de Campina Grande, 2006. 34 p.
- BRANDÃO, César F. M.. **Avaliação de agentes de conversação: a influência de elementos multimédia**. 2012. 78 f. Monografia (Mestrado) - Faculdade De Engenharia Da Universidade Do Porto, Portugal, 2012.
- BRATKO, Ivan. **Prolog: programming for artificial intelligence**. Harlow: Addison Wesley, 1990.
- CONPET, Petrobrás. **Robô Ed**. Disponível em:
<<http://www.ed.conpet.gov.br/br/converse.php>> Acesso em: 27 de abril de 2016.
- DEITEL, Harvey M. et al. **C#: como programar**. São Paulo: Pearson Education, 2003. xliii, 1153 p. ISBN 8534614598.
- DEITEL, Harvey M. et al. **XML como programar**. Porto Alegre: Bookman, 2003. xvii, 972 p. ISBN 8536301473.
- GRAVES, Mark. **Projeto de bancos de dados com XML**. São Paulo: Pearson Education do Brasil, 2003. xv, 518 p. ISBN 8534614717.
- LEONHARDT, Michelle. D. **Doroty: um chatterbot para treinamento de profissionais atuantes no gerenciamento de redes de computadores**. Porto Alegre: PPGC da UFRGS, 2005.
- LESTA, Uwe. **Swiplcs**. Disponível em:
<<http://www.lesia.de/prolog/swiplcs/Generated/ Index.aspx>>. Acesso em: 25 out. 2015
- LINGUATECA, **Projeto Floresta Sintá(c)tica**. Disponível em:
<<http://www.linguateca.pt/floresta/principal.html>>. Acesso em: 10 out. 2015
- LOEBNER, **What is the Loebner Prize?** Disponível em:
<www.loebner.net/Prizef/loebner-prize.html>. Acesso em: 10 maio 2016

MORAIS, José J. G. **Obtenção de expressões regulares pequenas a partir de autômatos finitos**. Porto: Faculdade de Ciências da Universidade do Porto, 2004. 119 p.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS. **PUC Minas**. Belo Horizonte, 26 fev. 2016. Disponível em: <<http://ask.fm/pucminas>>. Acesso em: 26 fev. 2016.

PRESSMAN, Roger. S. **Engenharia de Software**. 7.ed. Rio de Janeiro: McGraw-Hill, 2011.

QUINTÃO, Frederico P. **Projeto e Análise de Algoritmos: Casamento de Padrões**. Belo Horizonte: Universidade Federal de Minas Gerais, 2006. 80 p.

RINGATE, Thomas. **AIML reference manual**. Disponível em: <www.alicebot.org/documentation/aiml-reference.html>. Acesso em: 05 nov. 2015.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência artificial**. Rio de Janeiro: Elsevier, c2013. xxi, 988 p. ISBN 9788535237016.

SCHIESSL, José M. **Descoberta de conhecimento em texto aplicada a um sistema de atendimento ao consumidor**. Brasília: Universidade de Brasília, 2007. 106 p.

SETZER, Waldemar W.; HOMEM DE MELO, Inês S.. **A construção de um compilador**. Rio de Janeiro: Campus, 1989. ISBN 8570013345.

SGANDERLA, R. B.; Ferrari, D. N.; Geyer, C. F. R. **BonoBOT: um chatterbot para interação com usuários em um sistema tutor inteligente**. XIV Simpósio Brasileiro de Informática na Educação - SBIE - 2003, Rio de Janeiro, RJ, Brasil, p. 463-472, 12 de Novembro de 2003.

SILVA, Aluísio. E. da. **Vetores: algoritmos**. Betim, 2012. 21 p.

SPARK JONES, K.; WALKER, S.; ROBERTSON, S. **A probabilistic model of information retrieval: development and comparative experiments**. Parts 1 and 2, 2000.

TEIXEIRA, Sérgio. **Chatterbots - Uma proposta para a construção de bases de conhecimento**. Vitória: Universidade Federal do Espírito Santo, 2005. 101 p.

TURING, Alan. **On computable numbers, with application to the Entscheidungs**, Proceedings of the London Mathematical Society, Series 2, Volume 42, 1936.

VIEIRA, Newton. J. **Linguagens e Máquinas: Uma Introdução aos Fundamentos da Computação**. Belo Horizonte: Universidade Federal de Minas Gerais, 2004. 303 p.

WALLACE, R. S. **ALICE Silver Edition**. Disponível em:

<<http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa1>> Acesso em: 04 nov. 2015.

WEIZENBAUM, Joseph. **ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine**. Disponível em:

<<http://i5.nyu.edu/~mm64/x52.9265/january1966.html>>. Acesso em: 4.nov. 2015.

WIELEMAKER, Jan. **SWI prolog reference manual**. Amsterdam: VU University Amsterdam; University of Amsterdam, 2015. 533 p.

APÊNDICE A – Marcação processadas pelo Interpretador AIML

O interpretador AIML é capaz de interpretar um conjunto de marcações. Estas marcações devem ser estruturadas de forma a serem validas de acordo com os arquivos de definição de documento (DTDs). Os itens seguintes descrevem as marcações que podem ser utilizadas:

- **<aiml>**: Indica o início e o fim de um arquivo construído com o padrão AIML.
- **<bot name="name"/>**: retorna o nome definido para o agente;
- **<bot name="XXX"/>**: busca algum dos nomes definidos para o agente. Equivalente a `<srai>BOT XXX</srai>`;
- **<that>**: funciona em conjunto com **<pattern>** e representa o que foi dito anteriormente pelo AC. Para que a categoria que possua essa marcação seja encontrada, a última saída de texto gerada pelo AC deve ser igual ao conteúdo definido nela, além do casamento de texto com *pattern*. Pode conter o caractere “*”;
- **<category>**: delimita cada categoria definida no arquivo. Cada categoria contém um padrão e uma resposta e ainda pode possuir a marcação **<that>**.
- **<input index="n"/>**: busca entrada do usuário de índice “n”;
- **<condition name="X" value="Y">**: O atributo X pode ter seu valor (*value*) verificado. X pode assumir os valores “name”, “infoprolog” e “infomp”, Y pode receber qualquer sequência de caracteres.
- **<condition name="X" contains ="Y">**: O atributo X pode ter seu conteúdo (*contains*) verificado. X pode assumir os valores “name”, “infoprolog” e “infomp”, Y pode receber qualquer sequência de caracteres.
- **<condition name="X" exists ="Y">**: O atributo X pode ter sua “existência” (*exists*) verificada. X pode assumir os valores “name”, “infoprolog” e “infomp”, Y pode receber qualquer sequência de caracteres.
- **<gender>**: muda o gênero (ele, ela) de termos em uma frase;
- **<date/>**: busca data e horário atual em formato como “quarta-feira, 13 de janeiro de 2016, 01:19”;
- **<get name="user"/>**: nome do usuário atual;
- **<get name="location"/>**: localidade do usuário;
- **<get name="gender"/>**: gênero do usuário.

- **<size/>**: retorna a quantidade de categorias do AC;
- **<star index="n"/>**: busca o valor representado pelo caractere "*" de índice "n" contido em *pattern*. Observação: a marcação **<star/>** da versão AIML 0.9 também é utilizada. Os índices começam em zero;
- **<star/>**: equivalente a **<star index="1"/>**;
- **<thatstar index="n"/>**: busca o valor representado pelo caractere "*" de índice "n" contido em **<that>**;
- **<version/>**: retorna a versão do padrão AIML (1.0);
- **<gossip>**: acrescenta texto no documento *"gossip.txt"*;
- ****: elemento de lista usado em **<random>**;
- **<pattern>**: contém o padrão que pode ser identificado pela busca;
- **<random>**: marcação para resposta aleatória;
- **<set name="name">**: armazena o nome do usuário;
- **<set_male/>**: define o gênero do usuário como *"male"* (para gênero masculino);
- **<set_female/>**: define o gênero do usuário como *"female"* (para gênero feminino);
- **<set_name>**: define o nome do usuário;
- **<set_location>**: define localização do usuário.
- **<sr/>**: equivalente a **<srai><star/></srai>**;
- **<srai>**: marcação para busca recursiva;
- **<template>**: contém a resposta e texto que deve ser processado quando a categoria a que pertence é encontrada;
- **<think>**: elimina o possível texto remanescente de operações internas do computador;
- **<uppercase>**: converte o texto para caixa alta;
- **<lowercase>**: converte o texto para caixa baixa;
- **<sentence>**: converte o primeiro caractere do texto para caixa alta;
- **<formal>**: converte cada primeira letra de cada palavra para caixa alta;
- **<if name="X" value="Y">**: O atributo X pode ter seu valor (*value*) verificado. X pode assumir os valores *"name"*, *"infoprolog"* e *"infomp"*, Y pode receber qualquer sequência de caracteres.

- **<if name="X" exists="Y">** O atributo X pode ter sua “existência” (*exists*) verificada. X pode assumir os valores “*name*”, “*infoprolog*” e “*infomp*”, Y pode receber qualquer sequência de caracteres.
- **<if name="X" contains="Y">** O atributo X pode ter seu conteúdo (*contains*) verificado. X pode assumir os valores “*name*”, “*infoprolog*” e “*infomp*”, Y pode receber qualquer sequência de caracteres.
- **<else>**: alternativa processada quando a condição a que pertence não é validada;
- **<mprob>**: envia o texto entre **<mprob>** e **</mprob>** para o modelo probabilístico;
- **<prolog>**: envia o texto entre **<prolog>** e **<improb>** para o *SWI-Prolog*.

Exemplos de utilização das marcações estão em documentos no caminho “*ArquivosAIML\Exemplos*”.

APÊNDICE B – Teste de desempenho com MPC e BM25

A Tabela 3 apresenta o tempo de resposta em segundos para cada uma das cem consultas aplicadas aos modelos probabilísticos MPC e BM25, assim como a média e o desvio padrão para esses valores.

Tabela 3 - Teste de Desempenho

Consulta	MPC	BM25
1	0,1908111	0,1892329
2	0,1144674	0,1083529
3	0,1615582	0,1210801
4	0,1630286	0,0786220
5	0,1792626	0,1374392
6	0,0955429	0,0973729
7	0,1312794	0,1221846
8	0,1180990	0,0800668
9	0,1218518	0,1366561
10	0,1331930	0,0772385
11	0,1064617	0,0938088
12	0,0981734	0,1247077
13	0,1295954	0,0677742
14	0,1016866	0,0962546
15	0,0739767	0,0988176
16	0,0980676	0,0845830
17	0,0790456	0,0637230
18	0,1356898	0,1271468
19	0,1336742	0,1052491
20	0,1398475	0,1063591
21	0,1493525	0,0755284
22	0,1650560	0,1077746
23	0,1095897	0,0954079
24	0,1444202	0,0892172
25	0,0811764	0,1272684
26	0,0876248	0,0857064
27	0,1006958	0,0863688
28	0,1560756	0,1056778
29	0,0865949	0,1336067
30	0,0972679	0,1004743
31	0,1111532	0,1006717
32	0,1082262	0,1077628
33	0,0758853	0,0849513
34	0,1057753	0,0890557
35	0,0827340	0,1414068

Continua

Continuação

Consulta	MPC	BM25
36	0,1236633	0,1041407
37	0,0992404	0,1030792
38	0,1387000	0,1301713
39	0,1306904	0,1117536
40	0,1308897	0,0780781
41	0,0921572	0,0813465
42	0,1040807	0,1056367
43	0,2126898	0,1694529
44	0,0727278	0,1135718
45	0,1400090	0,0689785
46	0,0514561	0,0811302
47	0,2069535	0,2047568
48	0,0840974	0,0624898
49	0,0973010	0,1077454
50	0,1242195	0,1353215
51	0,1017861	0,1068837
52	0,1228019	0,1657992
53	0,2094380	0,2350396
54	0,0823847	0,1155656
55	0,0632860	0,0886965
56	0,1071581	0,1335657
57	0,0675247	0,1039157
58	0,0795781	0,0877140
59	0,0668852	0,0817582
60	0,0833995	0,1180892
61	0,1152770	0,1086715
62	0,0792331	0,0918872
63	0,0972774	0,1788659
64	0,1037163	0,0953274
65	0,0654073	0,1194968
66	0,0710537	0,1076029
67	0,0651117	0,0988417
68	0,1687381	0,1342581
69	0,0687058	0,0952113
70	0,0688207	0,0726129
71	0,1298216	0,1097965
72	0,1049495	0,0847416
73	0,0623169	0,0953834
74	0,0976859	0,1201967
75	0,0690172	0,0816520
76	0,1192221	0,1327320
77	0,0685353	0,1596630

Continua

Conclusão

Consulta	MPC	BM25
78	0,0786121	0,1276355
79	0,0708745	0,0912379
80	0,1386041	0,1228991
81	0,0839518	0,1034598
82	0,0628747	0,0760665
83	0,1009543	0,1083348
84	0,0852899	0,1081319
85	0,1426719	0,0773131
86	0,0837327	0,1184602
87	0,0724120	0,0949788
88	0,1001858	0,1010593
89	0,0848739	0,1129228
90	0,1009934	0,1390936
91	0,0986222	0,1414522
92	0,0713367	0,1246623
93	0,1389969	0,1630392
94	0,0582156	0,0969687
95	0,0826318	0,0986207
96	0,0863182	0,1313733
97	0,0717133	0,1102169
98	0,0872234	0,1097460
99	0,1015496	0,1431270
100	0,0657448	0,0800474
Média	0,1058333	0,1101002
Desvio padrão	0,0349582	0,0296586

Fonte: Dados dos testes aplicados.

Como é possível observar na tabela e como esperado, o BM25 é mais lento que o MPC, entretanto oferece resultados melhores. O MPC é mais rápido que o BM25, no entanto tem desempenho inferior de acordo com experimentos com a métrica MRR. O desvio padrão foi baixo nos dois casos, o que indica que houve pouca variação no tempo entre as consultas em um mesmo modelo probabilístico.

ANEXO A – AIML 1.0

O Quadro 6 apresenta as marcações do padrão AIML 1.0. A primeira coluna exibe a marcação, a segunda coluna o tipo a que a marcação pertence e a terceira coluna apresenta observações.

Quadro 6 - Marcações AIML

AIML 1.0	Tag Type	Note
<aiml>	<i>AIML block delimiter</i>	<i>[Closing tags not shown]</i>
<bot name="name"/>	<i>Built-in bot parameter</i>	<i>may appear in pattern</i>
<bot name="XXX"/>	<i>Custom bot parameter</i>	<i><srai>BOT XXX</srai></i>
<that index="2,1"/>	<i>Built-in predicate</i>	
<that index="nx,ny"/>	<i>Built-in predicate</i>	<i>default "that"</i>
<that>	<i>AIML that pattern</i>	<i>contains AIML pattern</i>
<category>	<i>AIML category</i>	
<input index="2"/>	<i>Built-in predicate</i>	
<input index="3"/>	<i>Built-in predicate</i>	
<condition name="X" value="Y">	<i>Conditional branch</i>	
<condition>	<i>Conditional branch</i>	
<gender>	<i>Gender substitution</i>	<i>Exchange "he" and "she"</i>
<date/>	<i>Built-in predicate</i>	<i>date and time</i>
<id/>	<i>Built-in predicate</i>	<i>default "localhost"</i>
<get name="xxx"/>	<i>Built-in predicate</i>	<i>default "X-person"</i>
<size/>	<i>Built-in predicate</i>	<i># of categories loaded</i>
<star index="n"/>	<i>Built-in predicate</i>	<i>binding of *</i>
<thatstar index="n"/>	<i>Built-in predicate</i>	<i>binding of * in that</i>
<get name="topic"/>	<i>Built-in predicate</i>	<i>default "you"</i>
<topicstar index="n"/>	<i>Built-in predicate</i>	<i>binding of * in topic</i>
<version/>	<i>Built-in predicate</i>	<i>AIML program version</i>
<get name="xxx"/>	<i>Custom predicate</i>	<i>Botmaster defined XXX</i>
<gossip src="X"/>	<i>Append to file</i>	
<learn>X</learn>	<i>AIML loading</i>	

Continua

Conclusão

AIML 1.0	Tag Type	Note
<li name="X" value="Y">	Conditional branch item	used by <condition>
<li value="Y">	Conditional branch item	used by <condition name="X">
	General list item	used by <random>, <condition>
<pattern>	AIML Pattern	contains AIML pattern
<person/>	Prounoun transform macro	<person><get_star/></person>
<person2>	Prounoun transform	swap 1st & 2nd person
<person2/>	Prounoun transform macro	<person2><get_star/></person2>
<person>	Prounoun transform	swap 1st & 3rd person
<random>	Random selection	Random uniform selection
<set name="name">	Built-in predicate	returns contentes
<set name="topic">	Built-in predicate	returns contentes
<set name="XXX">	Custom predicate	See Note 3.
<sr/>	Recursion macro	<srai><get_star/></srai>
<srai>	Recursion	
<system>	Execute OS shell	platform-dependent
<template>	AIML template	
<think>	Nullify output	
<topic name="X">	AIML topic group	X is AIML pattern
<uppercase>	Text manipulation	convert all text to Uppercase
<lowercase>	Text manipulation	convert all text to Lowercase
<sentence>	Text manipulation	capitalize the first word
<formal>	Text manipulation	capitalize every word
<if name="X" value=Y">	Conditional branch	
<else>	Conditional branch	
<javascript>	AIMLScript	Javascript

Fonte: AIML Reference Manual, 2001.