Perspectivas tecnológicas para o aprimoramento de chatbots educacionais em AIML

Fábio J. Paz[1], Clóvis Silveira[1], Aliane Krassmann[1], Liane M. R. Tarouco[1]

[1] Pós-Graduação em Informática na Educação, Universidade Federal do Rio do Sul, Porto Alegre, Brasil

fabiojpaz@gmail.com, csclovis@gmail.com, alkrassmann@gmail.com, liane@penta.ufrgs.br

Recibido: 05/10/2017 | Aceptado: 15/12/2017

Resumo

Agentes conversacionais ou chatbots vêm sendo cada vez mais explorados na educação por trazerem inúmeros benefícios, como permitir que estudantes interajam e se relacionem de forma mais humana com Ambientes Virtuais de Aprendizagem (AVA). Entretanto, desde o surgimento do projeto A.L.I.C.E. (1995) e da linguagem AIML poucas soluções open source emergiram para a criação de chatbots. Apesar de sua facilidade de uso, a AIML tem limitações inerentes a uma lógica baseada em regras. Nesse sentido, este artigo traz uma discussão sobre possibilidades open source existentes para o desenvolvimento chatbots, apresentando sugestões de formas de aprimoramento do uso da AIML com o uso da linguagem de programação JavaScript e o acesso a Bancos de Dados (BD) externos. Resultados do estudo apontam algumas soluções envolvendo as técnicas mencionadas, proporcionando ao chatbot maiores possibilidades de busca e respostas.

Palavras chave: Chatbot; Agentes conversacionais; Banco de Dados; JavaScript; AIML.

Abstract

Conversational agents or chatbots have been increasingly exploited in education because they bring numerous benefits, as allowing students to interact and relate more humanly to virtual learning environments. Since the emergence of A.L.I.C.E. Project (1995) and the AIML language few open source solutions have emerged for the creation of chatbots. Despite ease of use, AIML has limitations inherent in rule-based logic about input and output patterns. This article discusses open source possibilities for the development of conversational agents, presenting suggesting ways of improving the use of AIML by adding the use of the JavaScript programming language and the access to external databases. Results of the study point to some solutions the mentioned techniques, providing the chatbot to have greater possibilities of search and answers.

Keywords: Chatbot; Conversational agents; Database; JavaScript; AIML.

1. Introdução

Agentes conversacionais (chatbots) são definidos por Abushawar e Atwell [1] como sistemas que buscam simular uma conversa por meio da troca de entrada e saída de mensagens, tendo em vista uma comunicação natural e passível de não ser reconhecida como linguagem artificial [2]. Já são utilizados há algum tempo em diversos domínios, como atendimento a clientes e suporte técnico online [3], porém seu uso vem sendo ampliado principalmente a nível educacional [4] especialmente em sistemas de aprendizagem de Educação a Distância (EAD) e online (e-learning), chegando também a ambientes imersivos 3D [5]-[6].

Tibola et al. [7] salientam que um estudante de e-learning pode ficar frustrado se a falta de suporte adequado evitar a conclusão bem-sucedida de uma tarefa, sendo portanto, aconselhável fornecer-lhe algum complemento do monitor humano para orientar, ajudar e dar feedback. Chatbots

Cita sugerida: F. J. Paz, C. Silveira, L.M.R. Tarouco, "Perspectivas tecnológicas para o aprimoramento de chatbots educacionais em AIML", *Revista Iberoamericana de Educación en Tecnología y Tecnología en Educación*, Nº 20, pp. 7-15, 2017.

Licencia de distribución: Esta obra se distribuye bajo Licencia Creative Commons CC-BY-NC.

colaboram nesse sentido pois, de acordo com Jacob et al. [8] trazem o oferecimento de informações em respostas diretas a perguntas de estudantes, na forma de diálogo coerente, e conforme Leonhardt et al. [9], permitem que alunos se relacionem de forma mais humana com o computador, permanecendo 24 horas por dia à sua disposição.

Apesar de todos os esforços, esses sistemas ainda apresentam problemas, não só na sua construção, do ponto de vista de engenharia de software (por exemplo, reutilização, extensibilidade) [10], mas também em seu desempenho em conversas com usuários conforme ressaltado por Fryer [11] e Mou [12]. Além disso, entre seus desafios de desenvolvimento está a inserção de informações na sua respectiva base de conhecimento para que tenham um potencial de diálogos suficiente.

Danilava et al. [13] esclarecem que, tendo em vista as conversas com chatbots exigirem investimento no tempo em detrimento de outras atividades, deve haver uma razão clara pela qual um usuário decida utilizá-lo, o que segundo os autores, pode ser proporcional à sua utilidade. Nesse contexto, verifica-se importante e necessário a melhoria contínua no desenvolvimento de chatbots, visando atender com satisfação o requisito da utilidade, fornecendo respostas o mais eficientes, coerentes e relevantes possíveis.

Entre os chatbots o pioneiro foi a ELIZA criado em 1966 por Weizenbaum [14], onde a sua programação era baseada em scripts. No entanto, em 1995 Dr. Richard Wallace criou a linguagem de marcação AIML (Artificial Intelligence Markup Language) que atualmente é o ponto de partida para muitos interessados em processamento de linguagem natural [15]. Funciona com base em regras: quando uma entrada encontra uma correspondência o chatbot executa uma ação, como responder ou abrir uma página da web.

Embora existam outras soluções para a criação de chatbots, a AIML possui ampla utilização devido a sua facilidade de uso. Aguiar e Vitorino [16] alertam que para aperfeiçoar sua base de conhecimento é necessário ou ampliar o conteúdo abrangido ou refinar o padrão de perguntas dos usuários, incluindo mais alternativas e posibilidades. Entretanto, observa-se que alguns ajustes de ordem técnica ou tecnológica também permitem algum nível de aprimoramento da AIML tornando a interação com os mesmos mais próxima do tipo de conversação que pode ser estabelecida com humanos pois esta tem valor educacional superior, conforme estudo comparativo apresentado por Hill et al [17].

A pesquisa visa investigar formas de acrescentar melhorias em chatbots AIML por meio de aprimoramentos realizados nas configurações das bases de conhecimento e nos próprios sistemas. Para isso, inicialmente situa-se o estado da arte no desenvolvimento de chatbots, verificando as principais funcionalidades e limitações das soluções open source existentes. Na sequência são exemplificadas das algumas possibilidades práticas de aprimoramento e avanços no uso da AIML.

O artigo está estruturado da seguinte forma: na seção 2 resgata-se o arcabouço teórico que subsidia este trabalho, e, apresenta a discussão do estado da arte na construção de chatbots, trazendo uma síntese das soluções; as propostas de melhorias em um chatbot AIML são apresentadas na seção 3, onde são abordados o uso de JavaScript e acesso a Banco de Dados (BD); e por fim, as considerações finais do estudo são apresentadas na seção 4.

2. Estado da arte na construção de chatbots

A.L.I.C.E (Artificial Linguistic Internet Computer Entity) é um projeto no campo da Inteligência Artificial (IA), iniciado na década de 1990, com um sistema de código aberto mantido por uma comunidade ativa. O sistema é composto de duas partes: a máquina conversacional e a base de conhecimento construída usando a linguagem de marcação AIML [16].

Segundo Wallace [18] a AIML é baseada em padrões de entrada do usuário que são comparados aos padrões descritos na sua base de conhecimento, selecionando as melhores respostas como saída. De acordo com Wallace [2], sua estrutura é constituída de categorias, as quais consistem de ao menos dois elementos: o "pattern" e o "template", como no exemplo da Figura 1.

```
<category>
      <pattern>possível entrada do usuário</pattern>
      <template>resposta do agente conversacional</template>
</category>
```

Figura 1. Exemplo de código AIML, tag pattern

Embora a autoria em AIML seja fácil, sua característica *stateless*, que considera cada requisição como uma transação independente sem relação a qualquer requisição anterior, ou seja, a linguagem não consegue relacionar os conteúdos de interações sucessivas. Algumas tags amenizam o problema, como <think> (para processar uma entrada transparentemente) e <that> (para redirecionar a resposta de acordo com a pergunta anterior) [19], mas tais tags ainda não são suficientes para formar um diálogo lógicamente conectado que se assemelhe à conversação que poderia ser estabelecida com uma pessoa. Também existe a tag <srai>, que traz recursividade simples e redução simbólica [20], ampliando as possibilidades.

Porém, de acordo com McNeal e Newyear [15], uma grande quantidade de conteúdo é necessária para criar um chatbot em AIML convincente. Estes autores estimam que sejam necessárias aproximadamente 60.000 categorias. Cada pergunta ou conceito na base de conhecimento requer múltiplas categorias para corresponder às permutações da questão e garantir uma resposta adequada ou correta.

Desde o surgimento da A.L.I.C.E. e da linguagem AIML, poucas soluções *open source* surgiram para a criação de chatbots. Entre elas é possível mencionar Word2vec,

ChatScript e Façade, que são apresentadas sucintamente a seguir.

2.1. Word2vec – Deeplearning – Seq2Seq

Criado por uma equipe de pesquisadores do Google, trabalha com modelos de redes neurais treinadas para reconstruir contextos linguísticos de palavras. Sua entrada é um corpus de texto e sua saída é um conjunto de vetores, que são palavras posicionadas de modo que contextos comuns estão localizados próximos uns dos outros no espaço [21].

A Deeplearning implementa uma forma distribuída do Word2vec para Java e Scala, que funciona com Graphics Processing Unit (GPU), devido ao alto grau de processamento necessário para sua execução. Como as palavras são estados discretos, procura-se probabilidades de transição entre esses estados: a probabilidade de co-ocorrerem. O modelo Sequence To Sequence (Seq2Seq) introduzido nesta perspectiva consiste em duas RNNs (Recurrent Neural Network): um codificador e um decodificador. Seu objetivo é converter uma sequência de símbolos em um vetor de tamanho fixo que codifique apenas as informações importantes na sequência [22]. A Figura 2 apresenta um modelo de conversação Seq2Seq.

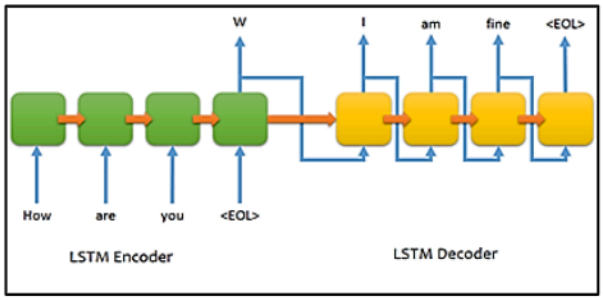

Figura 2. Modelo de conversação utilizando Seq2Seq

Por utilizar técnicas de aprendizado de máquina, o modelo Seq2Se2 pode evoluir e aprender sozinho, permitindo considerá-lo um sistema inteligente.

2.2. ChatScript

Criado pelo pesquisador Bruce Wilcox [23], foi desenvolvido em C e C++ com suporte JavaScript JSON (JavaScript Object Notation). É a base de linguagem natural para uma variedade de *startups* de tecnologia [15]-[13]. Assim como a AIML, também funciona baseado em regras, criadas através de um processo chamado *scripts* de fluxo de diálogo. Estes usam uma metalinguagem chamada de "*script*" como seu código-fonte. Um exemplo de *script* do ChatScript é apresentado na Figura 3 [23].

```
# file: food.top
#
topic: ~food []
#! I like spinach
s: ( I like spinach ) Are you a fan of the Popeye cartoons?

a: ( ~yes )  I used to watch him as a child. Did you lust after Oliv
Oyl?
            b: ( ~no ) Me neither. She was too skinny.
            b: ( yes ) You probably like skinny models.

    a: ( ~no ) What cartoons do you watch?
                b: ( none ) You lead a deprived life.
                b: ( Mickey Mouse ) The Disney icon.
```

Figura 3 - Exemplo de arquivo de script do ChatScript

Mas, de acordo com McNeal e Newyear [13], ao contrário da AIML que encontra a melhor combinação de padrões para uma entrada, ChatScript primeiro encontra o melhor tópico correspondente, então executa uma regra contida neste tópico, algo parecido com o que a tag AIML <topic> faz. Um diferencial do ChatScript é que permite os nomes plurais serem reduzidos sua forma singular, os verbos serem reduzidos para as formas infinitivas, bem como, os adjetivos e advérbios serem reduzidos para as suas formas de base. Acredota-se que tal diferencial amplia as posibilidades de diálogo de forma bastante significativa.

Entre as vantagens apresentadas pelos desenvolvedores do ChatScript estão o forte padrão de correspondência, documentação e suporte disponível, e o desempenho comprovado em uso comercial [24]. McNeal e Newyear [15] destacam entre os contras do software o fato de ser uma linguagem mais difícil de aprender do que o AIML, sendo desejável experiência em programação para isto e a maior dificuldade de inseri-lo em uma página web.

2.3. Façade

Os pesquisadores e criadores do Façade, Andrew Stern e Michael Mateas, desenvolveram o Façade com o intuito de ser superior à AIML ao utilizar a linguagem natural. Foi desenvolvido com Jess, uma extensão Java [24], que permite a declaração e retração de regras que se desencadeiam quando todas as condições prévias coincidem [25].

Segundo Wilcox [24] já foram escritas 800 regras de modelo em Façade, que se traduziram em 6.800 regras de Jess, um número pequeno para trabalhar com processamento de linguagem natural. Na Figura 4, um exemplo de regra Façade é exibido.

```
"hello" - iGreet
"grace" _ iCharacter(Grace)
iGreet AND iCharacter(?x) _ DaGreet(?x)
```

Figura 4. Código para geração de regras usando Façade.

O Façade é uma tecnologia para chatbots baseada em IA considerada uma arquitetura inovadora para apoiar o comportamento emocional e interativo de personagens e o enredo gerenciado por drama. Trata-se de um mundo virtual 3D habitado por personagens controlados por computador, no qual o usuário experimenta uma história de uma perspectiva de primeira pessoa. [25].

Segundo Stern e Mateas [26] uma das vantagens do Façade é o desempenho. Devido à necessidade de pouca quantidade de regras, as respostas chegam a 300 milissegundos, também gerando uma ótima performance em ambientes complexos, como mundos virtuais 3D.

Tabela 1 faz uma síntese das possibilidades *open source* para construção de chatbots apresentadas nesta seção em comparação com AIML.

Tabela 1. Síntese de soluções open source para construção de chatbots

	Origem	Lógica	Tecnologias	BD	Vantagem	Desvantagem
Word2vec	Google	Redes neurais/IA	Java Scala	Não informado	Aprendizado de máquina	Computação complexa
ChatScript	Bruce Wilcox	Regras Scripts	C++ JavaScript JSON	PostgreSQL MongoDB	Maior robustez	Dificuldade de aprendiza-gem e inserção em página Web
Façade	Andrew Stern e Michael Mateas	Regras Scripts	JESS (extensão de Java)	Não informado	Desempenho	Computação complexa
AIML	ALICE. Wallace	Regras Scripts	XML, HTML, JavaScript JSON	MySQL SQLSever	Facilidade de uso	Difícil incluir regras novas, falsos positivos

Como é possível observar na Tabela 1, apesar de não possibilitar aprendizado de máquina e possuir menor robustez e desempenho, a AIML traz como principal vantagem a facilidade de uso. Esse fator faz com que seja a solução mais adotada para a criação de chatbots no âmbito educacional, entre os quais pode-se mencionar: Prof. Elektra, com foco educacional para o ensino de física e redes de computadores para alunos de cursos de especialização em Informática na Educação [27]; Doroty, que treina usuários até que se sintam confiantes para administrar redes de computadores [27]; e Blaze, que foi criado com o objetivo de aprimorar habilidades cognitivas de estudantes durante a resolução de problemas matemáticos [28].

Entretanto, como um sistema baseado em regras, sua evolução é muito limitada, ficando restrita aos padrões percebidos nas perguntas dos usuarios e as respectivas respostas tal como incluído na base AIML. Buscando alternativas para superar estas limitações foram investigadas soluções para ampliar as posibilidades de resposta de chatbots baseados em AIML.

3. Propostas de melhorias em chatbots baseados em AIML

Ao longo dos anos, uma variedade de interpretadores da AIML surgiram usando Java, Ruby, Python, C++, C#, Pascal e outros. Entre eles PandoraBots que também dispõe de um serviço de hospedagem de chatbots na Internet. Segundo McNeal e Newyear [15] as melhores opções para criar um chatbot em AIML são Program-Z e Program-O, mas atualmente somente o Program AB é capaz de interpretar AIML versão 2.0. Nesta pesquisa instalou-se um chatbot utilizando o Program-O para exemplificar os procedimentos, em razão de ser *open source* e devido à sua praticidade de uso e grande comunidade de utilizadores. A Figura 5 apresenta a arquitetura do sistema.

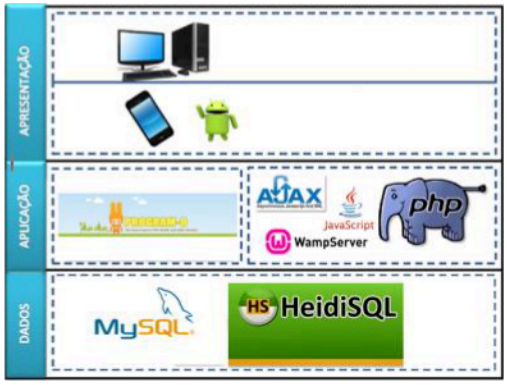

Figura 5. Arquitetura do sistema do chatbot.

Nas subseções subsequentes são apresentadas soluções envolvendo aprimoramentos para um chatbot AIML utilizando JavaScript e Banco de Dados.

3.1. Aprimoramentos em AIML utilizando JavaScript

O Program-O a partir de sua versão 2.6.5 permite a interpretação de JavaScript, o que é viabilizado inserindo dentro de categorias as tags <script>, abrindo diversas possibilidades. Por ex emplo, em uma resposta pode ser aberta uma janela de notificação inserindo o código alert("exemplo"), onde o texto "exemplo" será exibido na janela.

O JavaScript permite que sejam realizados cálculos matemáticos simples, como soma, subtração, divisão e multiplicação. No exemplo apresentado na Figura 6 é possível visualizar um código que soma duas variáveis atribuídas pelo *wildcard* (coringa).

```
<category>
      <pattern>Quanto é * mais * </pattern>
            <template>
                  O resultado é...
                  &lt;p id="demo"&gt;&lt;/p&gt;
                  <script>
                        var x = <star index = "1"/>;
                        var y = <star index = "2"/>;
                        var z = x + y;
                        document.getElementById("demo").innerHTML
                  = z;

                  </script>
            </template>
      </category>
```

Figura 6. Código para a realização de cálculos matemáticos simples

Observa-se na Figura 6 que tudo que é inserido fora das tags <script> deve ser transcrito para HTML. Assim, substituímos o símbolo "<" por "&lt" e o ">" por "&gt". No exemplo seguinte (Figura 7) é possível retornar a data e hora local (do sistema).

```
<category>
<pattern>Que dia é hoje</pattern>
      <template>
            Hoje é...
             &lt;p id="demo"&gt;&lt;/p&gt;
                  <script>
                  document.getElementById("demo").innerHTML
            = Date();
                  </script>
      </template>
</category>
```

Figura 7. Exemplo para retornar data e hora local do sistema

Aguiar e Vitorino [16] sugerem como um aperfeiçoamento de chatbots que o sistema poderia oferecer como resposta uma pesquisa realizada diretamente no motor de busca Google, sem que haja a necessidade do estudante fechar o ambiente de interação. Nesse sentido, com JavaScript é possível fazer com quem algum termo digitado pelo usuário seja automaticamente inserido em um site de busca, abrindo uma nova aba do navegador (Figura 8).

```
<category>
      <pattern>BUSCAR *</pattern>
            <template>
                Aí vai...
                      <script>
                      var x =
                window.open('http://google.com?q=<star />');
                      </script>
            </template>
      </category>
```

Figura 8. Exemplo para retornar data e hora local do sistema

Isso permite uma possibilidade destacada por Leonhardt et al. [18], de direcionar o interlocutor para o conhecimento, ou seja, o chatbot leva o estudante a endereços com mais informações relevantes sobre um determinado assunto.

Também é possível trabalhar com imagens de forma dinâmica utilizando JavaScript. Por exemplo, inserindo um botão que permite intercalar imagens que são apresentadas como resposta a uma entrada do usuário. No exemplo a seguir imagens de uma lâmpada apagada e acesa podem ser intercaladas. O código na Figura 9 exemplifica esta opção.

```
      <category>
      <pattern>vamos brincar </pattern>
            <template>
                  &lt;button
            onclick="document.getElementById('myImage').src='https://www.w3sc
            hools.com/js/pic_bulbon.gif'"&gt;Liga&lt;/button&gt;
                  &lt;img id="myImage"
            src="https://www.w3schools.com/js/pic_bulboff.gif"
            style="width:100px"&gt;
            &lt;button
onclick="document.getElementById('myImage').src='https://www.w3schools.com/js/
pic_bulboff.gif'"&gt;Desliga&lt;/button&gt;
            </template>
      </category>
```

Figura 9. Exemplo para intercalar imagens (lâmpada liga/desliga)

A Figura 10 mostra a interface do Program-O ao responder o comando mencionado. Assim, dentro da mesma resposta o estudante pode interagir e obter mais informações.

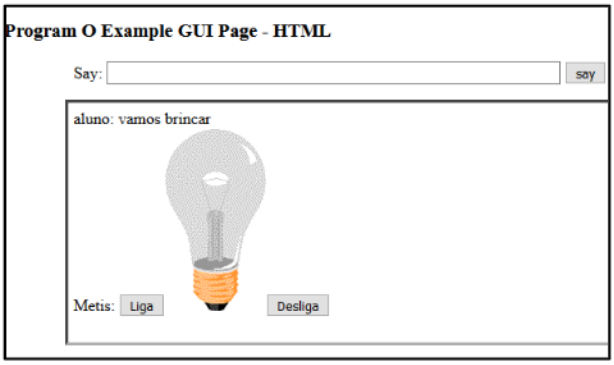

Figura 10. Imagem com botões de troca implementados em JavaScript

Os códigos apresentados para a função da imagem da lâmpada liga/desliga, bem como, o exemplo que retorna a data e horário local, o que realiza cálculos matemáticos e o que abre uma aba no navegador são apenas algumas sugestões investigadas neste estudo que abrem caminho para uma diversidade de novas possibilidades com a inserção de códigos JavaScript junto à AIML.

3.2. Aprimoramentos em AIML utilizando Banco de Dados

Para amplicar o conjunto de informações as quais o chatbot pode aceder na produção de uma resposta, uma solução interessante envolve utilizar Banco de Dados (BD) externos, permitindo a busca em outros repositórios, que podem conter textos, arquivos multimídia, entre outras informações. Nesta subseção são apresentadas algumas posibilidades para implementar esta conexão.

A tag <script> do JavaScript permite acessar BD utilizando a tecnologia AJAX (Asynchronous JavaScript And XML). Isto permite incluir no código sequências de comandos para acesso a banco de dados externo. No exemplo a seguir (Figura 11) o *script* acessa uma página em PHP que faz requisição a um BD MySQL.

```
<category>
      <pattern>teste</pattern>
            <template>
&lt;button id="ajaxButton" type="button"&gt;Clique aqui&lt;/button&gt;
&lt;div id="test"&gt;  &lt;/div&gt;
                  <script>
document.getElementById("ajaxButton").addEventListener('click', test);
            function test(){
                  $.ajax({
                        url: 'http://exemplo.br/~exemplo/exemplo.php';
                        type:'post',
                        success: function(response){
                              console.log(response);
                              $("#test").html(response);
                              }
                        });   }
                  </script>
            </template>
</category>
```

Figura 11. Acesso ao BD via JavaScript

Entretanto, para funcionar este procedimento é necessário ter uma extensão (*plugin*) no navegador do usuário que faz a requisição, chamada Allow-Control-Allow-Origin, que permite usar AJAX para solicitar qualquer site de qualquer fonte. No entanto, instalar esse *plugin* no navegador de todos os usuários que forem utilizar o chatbot se torna oneroso. Para facilitar esse processo, é possível alterar o código do Program-O para forçar uma busca no BD sempre que uma nova entrada é inserida no chatbot e não seja localizada uma resposta em sua própria base de conhecimento (categorias AIML). Para alcançar este objetivo foi criado um BD, conforme mostrou a Figura 12

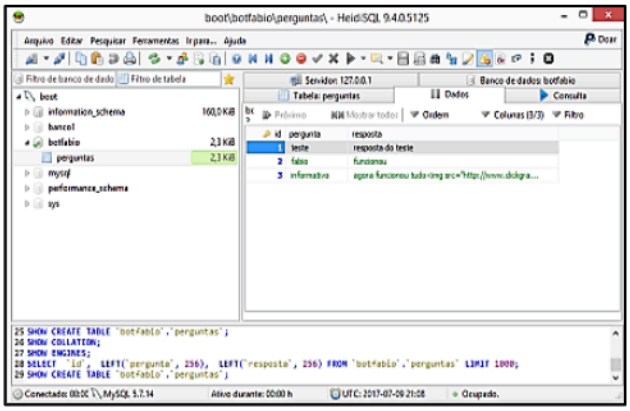

Figura 12. Imagem do Banco de Dados criado

Para esta conexão funcionar é necessário alterar o arquivo index.php que está localizado na pasta "\Program-O-dev\gui\xml", conforme ilustrado no código na Figura 13.

```
      <script type="text/javascript">
            $(document).ready(function() {
            var string_button = '<input type="submit" style="display: none"
id="enviar">';
            $('form').append(string_button);
            });
// criando o evento para quando teclar enter //
      $('input[type="submit"]').click(function(e) {
            e.preventDefault();
// cria uma requisição AJAX //
    var xhr = new XMLHttpRequest();
      xhr.open("GET", 'http://localhost/bot.php?pergunta=' + $('#say').val(),
true);
    xhr.onreadystatechange = function() {
// status 4 é sucesso.....//
   if (xhr.readyState == 4) {
        dados = $.parseJSON(xhr.responseText);
        if (dados.status == "ok") {
          // mostra a resposta na DIV
                $('div#response').html(dados.resposta);
        } else {
            // aqui abaixo clica o botão invisivel
                $('#enviar').trigger('click');
              }                    }
        }     xhr.send();
      }); </script>
```

Figura 13. Código de alteração do arquivo index.php do Program-O

No mesmo arquivo index.php é necessário incluir um *script* com o objetivo de acionar um evento a cada entrada no sistema (digitar uma frase e teclar enter). O respectivo evento faz uma requisição para o arquivo de conexão que efetuará a busca junto ao BD (Figura 14).

```
//  na linha 60 foi modificado o IF para entrar sempre na página correta.
if (!empty ($post_vars) || empty($post_vars))
```

Figura 14. Script para conexão com o Banco de Dados

O *script* apresentado na Figura 14 funciona da seguinte maneira: um acesso à base de conhecimento do chatbot é feito na primeira pesquisa e, caso não encontre a resposta, a busca é direcionada para o BD externo. Caso ainda não seja encontrada a resposta, o agente retorna uma resposta padrão. A Figura 15 ilustra esse processo.

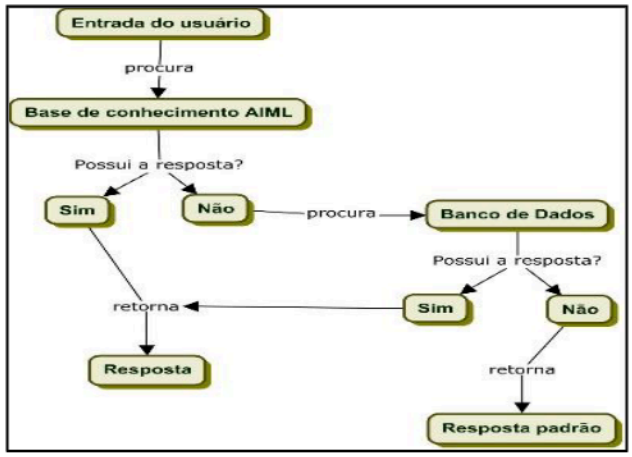

Figura 15. Processo do script que faz acesso ao Banco de Dados externo

Também é necessário criar o arquivo de conexão, que efetua a busca no BD e devolve a resposta da requisição efetuada. Este arquivo, chamado nesse exemplo de bot.php, deve ser colocado na pasta raiz do servidor. A questão chave nesse processo é a inserção da extensão Allow-Control-Allow-Origin diretamente no código interno do Program-O, evitando a instalação deste *plugin* em navegadores dos usuários, conforme mencionado anteriormente. Na Figura 16, visualiza-se o *header* do *plugin* e do JSON, que serve para a resposta da requisição ser em um formato de fácil entendimento e mais leve que o XML.

```
<?php
// para funcionar em qualquer domínio
      header('Access-Control-Allow-Origin: *');
// set para responder em json
      header('Content-Type: application/json');
// aqui faz a conexão
      $conexao = 'mysql:host=127.0.0.1;dbname=nomedobanco';
      try {
      $conecta = new PDO ($conexao, 'root', '');
      $conecta ->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            } catch(PDOexception $error_conecta) {
                  echo htmlentities('Erro ao conectar ao banco de dados
'.$error_conecta->getMessage().", por favor, informe no e-mail xxx@xx.com.");
                                                      }
      $pergunta = $_GET['pergunta'];
      $query = $conecta->prepare("SELECT resposta FROM perguntas WHERE
pergunta = ?");
      $query->bindValue(1, $pergunta);
      $query->execute();
      $resposta = $query->fetch(PDO::FETCH_COLUMN);
            if ($resposta) {
               echo json_encode(array('status' => 'ok', 'resposta' =>
      $resposta));
                        } else {
                         echo json_encode(array('status' => 'erro'));
                               }
```

Figura 16. Visualização do código header do plugin e do JSON

A partir do código apresentado na Figura 16, permite-se que no momento em que o usuário insira uma entrada no sistema, este verifique se aquela informação está na base de conhecimento AIML, e caso não seja encontrada, a busca será realizada no BD externo, retornando a informação para o usuário. A Figura 17 apresenta o resultado da conexão do Program-O com o BD exemplificado na Figura 12, e o resultado aparecendo na interface do Program-O.

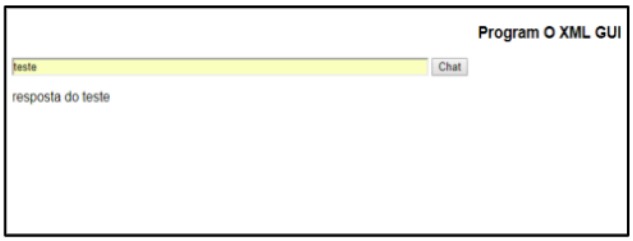

Figura 17. Resposta do Banco de Dados externo

Na Figura 18 observa-se outro exemplo de resposta. Neste caso, por meio da entrada "informativo" a busca realizada na base de conhecimento AIML do Program-O não encontrou a informação, mas essa entrada correspondeu a um link de imagem e texto armazenados no BD.

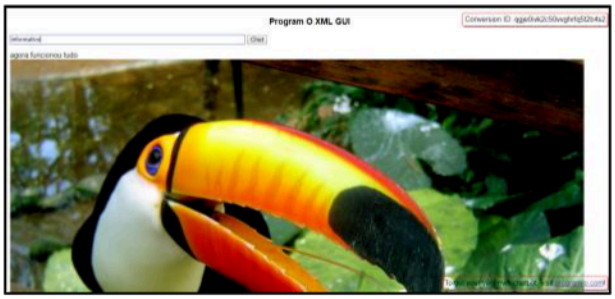

Figura 18. Resposta com imagem e texto armazenados em BD externo

No caso de o usuário digitar uma entrada e esta não possuir um padrão compatível nem na base de conhecimento AIML nem no BD externo, o processo retorna novamente para a base AIML e apresenta uma resposta padrão do Program-O, o que sugere-se que seja algo instigando o estudante a retomar diálogos sobre o qual o agente tenha conhecimento. Na Figura 19 é exibido um exemplo de resposta padrão, que pode ser aleatória dentre uma lista de opções de respostas pré-definidas.

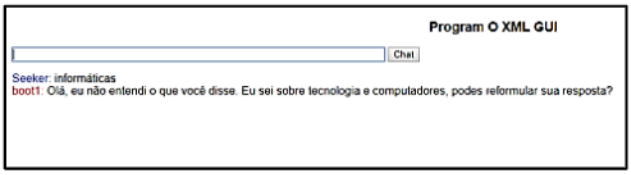

Figura 19. Resposta padrão Program-O

Assim, responder perguntas dos estudantes com registros armazenados em BD externos traz uma gama de possibilidades, como, por exemplo, acessar informações de outros sistemas de gerenciamento de aprendizagem, como Moodle. O chatbot pode acessar dados do estudante tais como notas, atividades e materiais didáticos disponibilizados nesses ambientes posibilitando ao aluno obter respostas do chatbot sobre estas informções.

Conclusões

Chatbots e assistentes virtuais são uma tendência na atualidade, trazendo informações de forma interativa, rápida e conveniente. Este artigo apresentou o estado da arte na construção de chatbots fazendo uma síntese das possibilidades *open source* disponíveis para sua construção. A AIML apesar de limitada ainda é a tecnologia mais utilizada para a construção de agentes conversacionais na educação devido a sua facilidade de uso, porém novas possibilidades com grande potencial estão em desenvolvimento como Word2vec, ChatScript, Façade.

Com o intuito de explorar o uso da AIML, foram apresentados alguns métodos de aprimoramento utilizando JavaScript e acesso a Banco de Dados externo, ampliando e permitindo possibilidades de melhoria na conversação com chatbots. No caso de uso de JavaScript foram apresentados exemplos de cálculos matemáticos, o retorno da função data e hora do sistema, a possibilidade do uso do motor de busca Google e a interação com imagens (lâmpada liga/desliga). Quanto ao acesso a BD externo apresentou-se a possibilidade de fazer requisições com AJAX (necessitando instalação de plugin) e alterando o código-fonte do sistema (sem necessidade de plugin), exemplificando respostas geradas no respectivo estudo.

Esta pesquisa traz como contribuição uma investigação sobre algumas formas de superar limitações da AIML de forma simples, como o apoio de recursos de ordem técnica e tecnológica, apresentando as configurações, códigos e *scripts* necessários para este fim. As soluções apresentadas têm a capacidade de trazer mais flexibilidade na construção de chatbots, aproveitando ferramentas e repositórios já existentes.

Visando disseminar o uso de chatbots na educação, foi explorada a possibilidade de expansão das possibilidades da popular máquina de inferência (A.L.I.C.E), por meio de

novas formas de recuperar e apresentar conteúdo além da AIML, que permite integrar-se com sistemas já existentes.

A investigação relatada envolveu o uso do banco de dados MySQL, criado para teste. Trabalhos futuros envolverão o teste das soluções aquí apresentadas com outros sistemas gerenciadores de banco de dados. Isto permitirá ampliar o uso do chatbot. Dessa forma, pretende-se ampliar as opções de interoperação do chatbot com vistas a incluir integração a bases de dados como a do ambiente Moodle, para obter informações sobre o desempenho e atividades dos estudantes, e também posibilitar a integração com de bibliotecas digitais e repositórios de objetos educacionais, aprimorando o uso da AIML na construção de chatbots para a uso educacional.

Referencias

[1] B. AbuShawar, E. Atwell, "ALICE Chatbot: Trials and Outputs". Computación y Sistemas, vol. 19, no. 4, pp. 625–632, 2015

[2] R. Wallace (1995, jun 24), ALICE: Artificial Linguistic Internet Computer Entity: The A.L.I.C.E A.I. Foundation. [online] Disponível: http://alicebot.blogspot.com.br

[3] B. AbuShawar and E. Atwell, Different measurements metrics to evaluate a chatbot system. Association for Computational Linguistics, 2007.

[4] J. Jacob et al. "Processo de Criação de um Modelo de Computação Afetiva para Chatbots", Anais do XXII SBIE - XVII WIE, 2011, pp. 1784-1791.

[5] A. Krassmann, et al., "Initial Perception of Virtual World Users: A Study about Impacts of Learning Styles and Digital Experience". International Educative Research Foundation and Publisher. 2017, pp 95-112.

[6] F. Sgobbi, et al., "Mundo virtual 3D e Internet das Coisas para motivar mudança de comportamento saudável". Revista Iberoamericana de Educación en Tecnología y Tecnología en Educación. N°19, 2017, pp 7-15.

[7] L. Tibola, et al., "Virtual laboratory for promoting engagement and complex learning". In E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education. Association for the Advancement of Computing in Education (AACE), 2014, pp. 1933-1938.

[8] Jr. Jacob, et al., "Processo de Criação de um Modelo de Computação Afetiva para Chatbots". Anais do XXII SBIE - XVII WIE, 2011, pp. 1784 – 1791.

[9] M. D. Leonhardt, et al., "Elektra: Um Chatbot para uso em ambiente educacional". RENOTE, no.1 2003.

[10] M. M. N. André, et al., "Iaiml: A mechanism to treat intentionality in aiml Chatbots." Tools with Artificial Intelligence, 18th IEEE International Conference on. IEEE, 2006, pp. 225-231.

[11] L. K. Fryer, et al., "Stimulating and sustaining interest in a language course: An experimental comparison of Chatbot and Human task partners." Computers in Human Behavior, 2017, pp. 461-468.

[12] Y. Mou, and X. Kun, "The media inequality: Comparing the initial human-human and human-AI social interactions." Computers in Human Behavior, 2017, pp. 432-440.

[13] S. Danilava, et al., "Artificial conversational companions a requirements analysis". Proceedings 4th International Conference on Agents and Artificial Intelligence, SciTePress, pp. 282-289, 2012.

[14] J. E. Weizenbaum, "A computer program for the study of natural language communication between man and machine". Communications of the ACM, v. 9, n. 1, pp. 3644, 1966.

[15] M. L. McNeal and D. Newyear, "Chatbot creation options". Library Technology Reports, 49(8), pp. 11-17, 2013.

[16] E. V. B Aguiar, N. Vitorino, "Ferramentas e métodos para aperfeiçoamento do funcionamento de um agente conversacional". Tecnologias Digitais na educação: Pesquisas e Práticas Pedagógicas. Essentia Editora. DOI: 10.19180/978-85-99968-49-9.7. , pp. 106-114, 2015.

[17] J. Hill et al, "Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations". Computers in Human Behavior. 49. DOI: 10.1016/j.chb.2015.02.026, pp. 245-250, 2015.

[18] R. Wallace. (2003, march, 28), Elements of AIML Style. ALICE A. I. Foundation. [online] Avaliable: http://www.alicebot.org/style.pdf

[19] S. Ghose and J. J. Barua, "Toward the implementation of a topic specific dialogue based natural language chatbot as an undergraduate advisor." Informatics, Electronics & Vision (ICIEV), International Conference on. IEEE, 2013.

[20] B. Abushawar and E. Atwell, "Chatbots: are they really useful?", LDV-Forum – Band 22(1), pp. 31-50, 2007.

[21] .J. Patterson and A. Gibson, "Foundations of Neural Networks and Deep Learn" in Deeplearning. O'Reilly Media, Inc., 1005. 1. Ed., pp. 41-80, 2017.

[22].S. Ramamoorthy. (2017, june, 17). Chatbots with Seq2Seq..[online]..Avaliable:.http://suriyadeepan.github.io/2016-06-28-easy-seq2seq/

[23] B. Wilcoxe. (2017, june, 02). Chatscript. Natural Language tool/dialog manager. [online]. Avaliable: https://github.com/bwilcox-1234/ChatScript

[24] B. Wilcox. (2017, june, 15). Beyonce Façade: Pattern Matching for Natural Language Applications..[online]..Avaliable:.http://www.gamasutra.com/view/feature/6305/beyond_fa%C3%A7ade_pattern_matching_.php?page=1

[25] M. Mateas and A. Stern. (2002, december, 5). Architecture, Authorial Idioms and Early Observations of the Interactive Drama Façade. [online]. Avaliable: http://reports-archive.adm.cs.cmu.edu/anon/2002/CMU-CS-02-198.pdf

[26] M. Mateas and A. Stern. (2017, june, 02). Natural Language Understanding in Façade: Surface-text Processing..[online]..Avaliable:.http://www.interactivestory.net/papers/MateasSternTIDSE04.pdf

[27] M. D. Leonhardt and L. Tarouco. Aplicando Linguagem Natural ao Gerenciamento de Redes de Computadores através do Chatbot Doroty. XXV Congresso da Sociedade Brasileira de Computação. UNISINOS – São Leopoldo – RS, 2005.

[28] E. Aguiar, et al., "A construção do conhecimento matemático com engajamento e aprimoramento de habilidades cognitivas apoiada por um agente conversacional". RELATEC Revista Latinoamericana de Tecnología Educativa. vol. 10, no. 2, pp. 21-35, 2011.

Información de Contacto de los Autores:

Fábio Josende Paz
Agentes Conversacionais
Universidade Federal do Rio Grande do Sul
Av. Paulo Gamo, 110 – Porto Alegre -BrasilPaís
fabiojpaz@gmail.com

Clóvis da Silveira
Agentes Conversacionais
Universidade Federal do Rio Grande do Sul
Av. Paulo Gamo, 110 – Porto Alegre -BrasilPaís
csclovis@gmail.com

Aliane Krassmann
Mundos Virtuais 3D
Universidade Federal do Rio Grande do Sul
Av. Paulo Gamo, 110 – Porto Alegre -BrasilPaís
alkrassmann@gmail.com

Liane Margarida Rockenbach Tarouco
Mundos Virtuais 3D
Universidade Federal do Rio Grande do Sul
Av. Paulo Gamo, 110 – Porto Alegre -BrasilPaís
liane@penta.ufrgs.br

Fábio Josende Paz

Doutorando em Informática na Educação (PPGIE/UFRGS), Mestrado em Mestrado em Sistemas e Processos Industriais, Especialista em Educação a Distância, graduado em Sistemas de Informação.

Clóvis da Silveira

Doutorando em Informática na Educação (PPGIE/UFRGS), Mestrado em Diversidade Social e Inclusão Social (linha de Pesquisa Inclusão Digital), Especialista em Educação à Distância, Licenciado em Computação.

Aliane Krassman

Doutoranda em Informática na Educação (PPGIE/UFRGS), Mestrado em Ciência da Computação, Especialista em Redes de Computadores e Bacharel em Ciência da Computação.

Liane Margarida Rockenbach Tarouco

Doutorado Engenharia Elétrica-Sistemas Digitais, Mestrado em Ciência da Computação, Licenciatura em Física Professora titular da Faculdade de Educação da UFRGS.