DOI: 10.19180/978-85-99968-49-9.7

Ferramentas e métodos para aperfeiçoamento do funcionamento de um agente conversacional

Eliane Vigneron Barreto Aquiar
Instituto Federal Fluminense [eviqneron@iff.edu.br]
Doutora em Informática na Educação na UFRGS
Neemias Vitorino
Instituto Federal Fluminense[neemiasuf@qmail.com]
Graduando em Bacharelado em Sistemas de Informação

Um agente conversacional pode facilitar o aprimoramento das habilidades dos estudantes de pensamento crítico e criativo (MIKIC et al., 2009; ANDRE et al., 1999), o que é fundamental para o desenvolvimento da capacidade de resolução de problemas (SENDAG; ODABASI, 2009; VILA; CALLEJO, 2006). Como resultado de pesquisa anterior (AGUIAR, TAROUCO, REATEGUI, 2011), foi criado um agente conversacional, chamado Blaze, utilizando a linguagem de marcação AIML (*Artificial Intelligence Markup Language),* que permite interagir com o agente em linguagem natural (WALLACE, 1995).

No entanto, existem inúmeras tecnologias que podem contribuir para o aprimoramento do Blaze, tornando-o mais dinâmico e inteligente, além de aumentar sua performance. É possível citar, como exemplo, a implementação de um banco de dados distribuído, de um sistema de predição de perguntas e respostas, e de componentes gráficos *front-end* (*HTML, CSS e JavaScript*), além da reescrita do algoritmo em linguagens de programação mais recentes e dinâmicas, como *Python* e *Ruby*, integração à plataforma de pesquisa *Google* e criação de um aplicativo *offline*, multiplataforma. Com base nisso, este artigo descreve essas tecnologias que podem ser utilizadas para aperfeiçoamento do Blaze, utilmente adaptáveis a outros agentes conversacionais.

Além disso, o artigo comenta sobre a efetiva implementação dessas ferramentas e desses métodos, descrevendo todo o seu processo de desenvolvimento, apontando falhas e erros, assim como pontos positivos. A seção 1 apresenta uma revisão de literatura sobre agentes conversacionais, subsídios necessários para o desenvolvimento da pesquisa. A seção 2 trata do agente conversacional Blaze, a seção 3 aborda a reestruturação do agente e a seção 4 comenta sobre as tecnologias ponderativas, apresentando as linguagens de programação e as ferramentas que podem ser usadas para aprimoramento do funcionamento do agente.

1. AGENTES CONVERSACIONAIS

Na década de 1980, começaram a ser criados sistemas tutores inteligentes com uma base de conhecimento para guiar os estudantes no processo de aprendizagem. Um tutor inteligente é um sistema de *software* que utiliza técnicas de inteligência artificial para representar o conhecimento e interage com os estudantes para lhes ensinar. Nos anos de 1990, com o avanço da psicologia cognitiva, os sistemas tutores inteligentes evoluíram de uma proposta instrucional visando estruturar um ambiente para a experimentação e a descoberta do conhecimento (MURRAY, 1999). Contudo, apesar desses avanços, a complexidade da estrutura desses sistemas limitou bastante sua aplicação prática. Desenvolver um sistema tutor inteligente é uma tarefa difícil. Isso se deve tanto à complexidade da tecnologia necessária para representação do conhecimento, modelagem cognitiva, processamento qualitativo e processo de modelagem causal, quanto à dificuldade de elicitação e representação do conhecimento do domínio (MIKIC et al., 2009).

A.L.I.C.E (*Artificial Linguistic Internet Computer Entity*)[1] foi um projeto bastante original e inovador no campo da inteligência artificial, na década de 90. Tratava-se de um exemplo de agente conversacional, com sistema de código aberto mantido por uma comunidade ativa. O sistema opera até hoje e é composto de duas partes: a máquina conversacional e a base de conhecimento construída usando a linguagem de marcação AIML (*Artificial Intelligence Markup Language*). A linguagem possui estrutura específica constituída de "categorias", as quais consistem de ao menos dois elementos: o "*pattern*" e o "*template*", como no exemplo a seguir (WALLACE, 1995).


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

O funcionamento de um agente usando AIML é baseado em um modelo de estímulo-resposta, no qual o estímulo (a entrada dos usuários) é comparado com padrões ("*pattern*"). Quando ocorre um ou mais casamentos de padrões, é determinada uma resposta associada, contida no "*template*", que o agente conversacional mostrará para o usuário (WALLACE, 1995). Todas essas ações, sobre ver o adequado *pattern* e mostrar o relacionado *template,* são carregadas pela máquina de tratamento de dados.

[1] A.L.I.C.E. *Artificial Intelligence Foundation*. [S.l.: S.n.], 2011. Disponível em: <alicebot.org>. Acesso em: 10 abr. 2011.

Diversos agentes conversacionais foram construídos usando a linguagem AIML. Cybelle é uma agente que interage em português e é capaz de interagir, também, em inglês e francês (AGENTLAND, 2002). Ela dá informações sobre outros agentes, como por exemplo, o ALICE. A Profª. Elektra é uma agente educacional que tem como principal objetivo ser um instrumento de complementação no aprendizado de estudantes de cursos a distância (LEONHARDT, 2005). CHARLIE é um agente responsável por interagir com os estudantes e o sistema educacional inteligente, mostrando conteúdos dos cursos e perguntando sobre o material de aprendizagem (MIKIC et al., 2009). A ampla gama de agentes conversacionais desenvolvidos com a máquina de inferência, baseada no projeto ALICE, levaram à escolha desta para o projeto de um agente conversacional para aprimoramento das habilidades de resolução de problemas.

2. O AGENTE CONVERSACIONAL BLAZE

A base do agente conversacional Blaze foi estruturada na linguagem AIML a partir do conhecimento extraído dos estudantes talentosos durante a resolução de problemas de matemática. Esses estudantes são medalhistas da Olimpíada Brasileira de Matemática das Escolas Públicas e participam de um projeto de iniciação científica no qual a autora atua. O *software* que implementa o agente conversacional Blaze está hospedado em um servidor público (*Pandorabots.com*). O agente é capaz de responder às manifestações dos estudantes, que com ele interagem, a partir das informações armazenadas em sua base de conhecimento. O agente Blaze não resolve os problemas para o aluno, mas pode servir como um assistente capacitado e confiável durante o processo de resolução dos problemas. Por meio de palavras-chave ou de questionamentos, os alunos podem dialogar com o Blaze, que fornece dicas para resolver novos problemas de matemática. Um exemplo de um diálogo entre o Blaze e o estudante é apresentado na figura 1.

No exemplo, o agente responde uma pergunta do estudante sobre o tema “quadrado mágico” e exibe um vídeo com a definição e exemplos de quadrados mágicos.

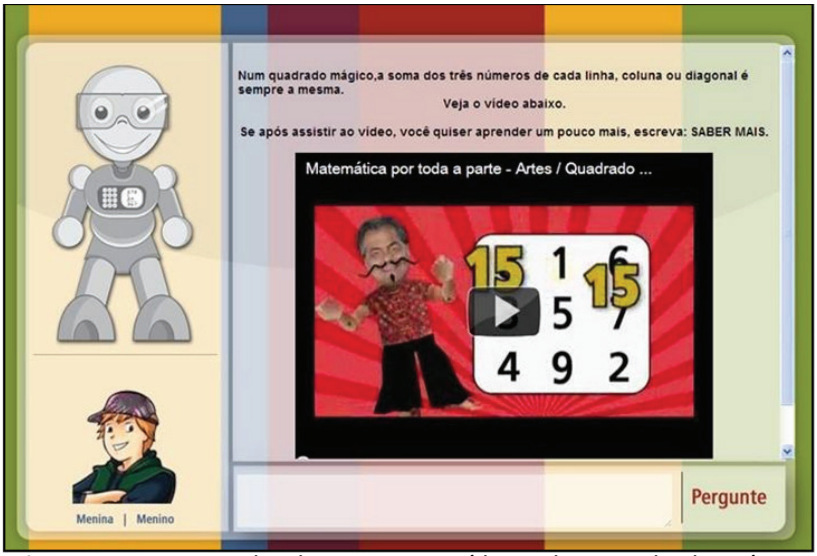

Figura 1 - Resposta do Blaze com um vídeo sobre quadrado mágico
Fonte: Aguiar, 2011, p. 136.

3. REESTRUTURAÇÃO DO PROJETO

Em 2013, houve a necessidade de dar continuidade à pesquisa original (AGUIAR, 2011), com o objetivo de aprimorar o agente conversacional Blaze, acrescentando-lhe recursos e aprimorando seu motor de pesquisa. Para continuar o desenvolvimento do *software* Blaze, haveria a necessidade de reaproveitar o código-fonte implementado nos servidores da *Pandorabots*, *Inc*. e implantado em 2011, com uma amostra de estudantes. Infelizmente, os servidores da empresa sofreram queda, resultando na perda do código *front-end,* responsável pela interface com a qual os usuários interagem (K19 Treinamentos). Em contato com a empresa, foi possível recuperar apenas o código *back-end*, responsável pelo funcionamento interno da aplicação Blaze e seu motor de pesquisa (K19 Treinamentos). Assim, a reestruturação do projeto não só ponderou tecnologias de aprimoramento do código *back-end*, quanto à reiniciação do desenvolvimento do código *front-end*.

Efetivamente, foi reiniciado o desenvolvimento do código *front-end,* utilizando linguagens como HTML, CSS*, jQuery e* PHP. A página criada possui três tópicos: informações sobre o projeto, ambiente de interação com o Blaze e formulário de contato. No entanto, o código ainda está em fase de testes e não representa reais funcionalidades de interação com o Blaze. Temporariamente, ele está disponível publicamente no endereço eletrônico

<http://blaze.url.ph>. Em momento posterior, esse código será implantado em um servidor do Instituto Federal Fluminense (IFFluminense), quando atingir versões mais estáveis.

4. TECNOLOGIAS PONDERATIVAS

4.1 Aperfeiçoamento do código AIML

O primeiro método de aprimoramento do Blaze é o mais simples e evidente. Trata-se de aperfeiçoar sua base de conhecimento. Isso pode ser feito de duas maneiras: ampliar o conteúdo matemático abrangido e refinar o padrão de perguntas dos usuários. Ambas são independentes, mas baseiam-se no desenvolvimento de padrões estímulo-resposta. A primeira induz à criação de mais categorias, incluindo seus elementos de entrada do usuário (pergunta) e saída do Blaze (resposta), em que cada categoria representará um assunto matemático diferente. A segunda maneira prevê a criação de mais opções de pergunta para um mesmo assunto matemático, ramificando os padrões "*pattern*", que representam a possível pergunta do usuário. Ao fazer isso, haverá um leque maior de possibilidades de perguntas para um mesmo assunto, tornando mais eficiente o sistema de pesquisa do Blaze.

4.2 Cadastro de usuários

Atualmente, o Blaze é acessado de forma anônima, ou seja, sem a necessidade de autenticação de identidade, usualmente feita por meio de um *login* e senha de um usuário registrado no sistema. Acredita-se que seria interessante acrescentar esse recurso ao Blaze, mas, em um sistema no qual praticidade e objetividade são primordiais, o uso dessa ferramenta seria questionável. Todavia, com um usuário registrado, é possível associar informações relacionadas ao seu perfil, suas principais buscas, suas preferências e inúmeras ferramentas que tornam o ambiente mais amigável e pessoal. Avaliando-se somente os requisitos por parte do usuário, ao se utilizar esse recurso, fica evidente que a oferta independente de registro no sistema do Blaze é ponderativa. Em contrapartida, do ponto de vista do gerente do sistema, é possível gerar informações estatísticas com base nos dados de cada usuário, coletando principalmente informações sobre erros no sistema e *feedback* do usuário. A partir disso, o gerente do projeto será capaz de avaliar pontos positivos e negativos do sistema, auxiliando-o na tomada de decisões e gerando uma melhoria contínua deste.

4.3 Implementação em outras linguagens

A Fundação de Inteligência Artificial (*AI Foundation*, em inglês) e sua comunidade, mantenedores do projeto A.L.I.C.E., possuem várias implementações de *software* livre que interpretam a linguagem de marcação AIML, nativa do Blaze. No projeto original, ele estava hospedado nos servidores da *PandoraBots, Inc*., que não oferecem suporte para os recursos avançados propostos neste artigo. Entretanto, com a implantação do Blaze em um servidor do Instituto Federal Fluminense, é possível utilizar qualquer uma das implementações AIML do projeto A.L.I.C.E., em linguagens de programação, desde *C++* e *Java* até *Python* e *Ruby*. Por se tratar de um projeto de *software* livre, o código-fonte está disponível para *download* e alterações, de forma legal. Portanto, inúmeras implementações podem ser feitas em qualquer uma das linguagens em que o Blaze já é implementado, criando um *software* ramificado que atende as expectativas deste projeto. Essa ação é comumente chamada de "*fork*" na área de desenvolvimento de *software* (OSI, 2013).

A escolha da linguagem a ser implementada depende dos requisitos do projeto, pois estes definirão a necessidade principal do sistema, que pode ser melhor atendida em um linguagem do que em outra. As linguagens de programação dinâmicas, como Python e Ruby, são boas escolhas, pois possuem características peculiares como as outras não dinâmicas.

Segundo Kiczales et al. (1991) e Botelho (2011), o que diferencia uma linguagem (orientada a objeto) dinâmica de uma não dinâmica são dois aspectos fundamentais: introspecção e interseção. Introspecção é a habilidade que uma linguagem tem de fazer com que seus objetos ou classes consigam ler seus atributos. Isso nos possibilita saber quais atributos ou métodos fazem parte de uma determinada classe. Interseção é a habilidade que uma linguagem tem de poder alterar o comportamento de um determinado objeto em tempo de execução. Esse aspecto nos permite fazer um *binding,* que é uma técnica de associar um identificador a uma classe, método ou objeto, ou seja, fazer uma referência por meio de um identificador de um método para um objeto/classe.

O *Meta Object Protocol* (MOP) introduz, em uma linguagem de programação, o suporte à introspecção e à interseção por meio do armazenamento do estado de seus objetos ou classes em meta-objeto ou meta-classe, associados a cada uma das entidades em questão (KICZALES et. al., 1991; BOTELHO, 2011). Algumas linguagens implementam o MOP

parcialmente, como é a do *Java* que é puramente introspectiva para dar suporte a API *Reflection*. Outras linguagens implementam por completo como é a do *Python*, *Scala, Ruby, Smalltalk* e *Groovy.*

Portanto, a escolha de uma dessas linguagens potencializa o desenvolvimento do Blaze. Apesar da implementação em *Ruby* (Program R) do interpretador AIML proveniente da AI *Foundation* ainda precisar de inúmeros testes e correções, isso não a descarta como uma das melhores opções. Em contrapartida, existe também a implementação em *Python*, o Program Y, mais estável e desenvolvido, utilizando 100% de bibliotecas e pacotes nativos da linguagem. Ainda assim, existem outras opções que podem ser implementadas, tomando como critério de escolha o escopo do projeto.

De qualquer forma, haverá a necessidade de implantação do Blaze na *web*. Para isso, será necessário utilizar um *framework* de desenvolvimento *web*. As mesmas linguagens de programação citadas acima possuem excelentes *frameworks*, como *Ruby on Rails* e *Django*, respectivamente, das linguagens *Ruby* e *Python*.

4.4 Inteqração com o *Google*

Ainda que todas as tecnologias mencionadas neste artigo fossem implementadas, haveria situações excepcionais para o Blaze, como, por exemplo, a entrada de uma pergunta que não estivesse em sua base de conhecimento. Como solução alternativa, o sistema poderia oferecer a pesquisa do assunto relacionado à pergunta diretamente no motor de pesquisa da web Google, sem que haja a necessidade de sair do ambiente de interação Blaze. Assim que o usuário encontrasse sua resposta, ele poderia indicar ao Blaze onde a encontrou. Nesse caso, o Blaze monitoraria constantemente os resultados de pesquisa do Google em seu próprio ambiente e, a cada página em que o usuário visualizasse, o Blaze perguntaria se ele encontrou o que procurava. Se o usuário respondesse positivamente, o Blaze armazenaria, em seu banco de dados, as informações relevantes àquela sessão e os dados da nova solução para aquele problema específico. Essa é mais uma forma de *feedback* do usuário, que faz com que o Blaze aprenda dinamicamente e apresente a solução aprendida posteriormente, quando outro usuário fizer um pergunta igual ou semelhante.

4.5 Aplicativo *off-line*

Até o momento, o Blaze estaria limitado a usuários com acesso à Internet, visto que o projeto prevê sua implantação em um servidor do IFFluminense, que pode ser acessado somente por meio da Internet (*on-line*). Com o intuito de ampliar a disponibilidade do sistema Blaze, pondera-se a criação de um aplicativo *off-line,* ou seja, sem a necessidade de conexão à Internet. Esse aplicativo pode ser implementado para funcionamento *on-line* e *off-line* em diversos dispositivos, como *notebooks*, *desktops*, *tablets* e *smartphones*. Esse aplicativo disponibilizaria os mesmos recursos do ambiente original do Blaze, porém adaptado e otimizado para cada dispositivo em particular, em diferentes plataformas, como *Android* e *iOS*.

5. CONSIDERAÇÕES FINAIS

Este artigo apresentou ferramentas e métodos de aprimoramento de um agente conversacional, ocasionalmente chamado Blaze, capaz de representar os processos de resolução de problemas de estudantes talentosos com vistas a apoiar outros estudantes na resolução de problemas matemáticos.

Existem inúmeras possibilidades de aprimoramento e de implementação de tecnologias que tornariam agentes conversacionais mais eficientes, inteligentes e dinâmicos. Em trabalhos futuros, pretende-se efetivamente implementar e implantar os métodos e as ferramentas apresentados no presente artigo, evidentemente suscetíveis à utilização de novos recursos.

REFERÊNCIAS

AGENTLAND. *Cybelle*. [S.l.: S.n.], 2002. Disponível em: <http://www.agentland. com>. Acesso em: 11 abr. 2011.

AGUIAR, E. V. B.; TAROUCO, L.; REATEGUI, E. Um Agente Conversacional para Aprimoramento das Habilidades de Resolução de Problemas. In: CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN - CACIC*,* 17., 2011, La Plata. *Anais*... La Plata: CACIC, 2011.

AGUIAR, E. V. B. *Aprimoramento das Habilidades Cognitivas de Resolução de Problemas com o Apoio de Um Agente Conversacional*. 200p. Tese (Doutorado em Informática na Educação) - Universidade Federal do Rio

Grande do Sul. Porto Alegre: RS, 2011.

A.L.I.C.E. *Artificial Intelligence Foundation*. [S.l.: S.n], 2011. Disponível em: <alicebot.org>. Acesso em: 11 abr. 2011.

ANDRE, E.; RIST, T.; MULLER, J. Employing AI methods to control the behavior of animated interface agents. *Applied Artificial Intelligence*, [S.l.], v. 13, n. 4-5, p. 415-448, may 1999.

BOTELHO, R. Uma visão Geral sobre Linguagens de Programação Dinâmicas (LPD). *Devmedia*, [Belém], 2011. Disponível em: <bit.ly/IOhrPS>. Acesso em: 11 dez. 2013.

KICZALES, et. al. *The Art of the Metaobject Protocol*. [S.l.]: The MIT Press, 1991.

K19 TREINAMENTOS. *Desenvolvimento Web com HTML, CSS e Javascript*. São Paulo, [2013?]. Disponível em: <bit.ly/19qPt70>. Acesso em: 11 dez. 2013.

LEONHARDT, M. D. *Doroty*: um Chatterbot para treinamento de profissionais atuantes no gerenciamento de redes de computadores. 2005. Dissertação (Mestrado em Computação) – UFGRS, Porto Alegre, 2005.

MIKIC, et al. *An AIML-based Chatterbot which Works as an Interface among INES and Humans*. Spain: Telematics Engineering Department, University of Vigo, 2009.

MURRAY, T. Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art, Internat. *Journal of Artificial Intelligence in Education*, v.10, p. 98-129, 1999.

SENDAG, S.; ODABASI, H. F. Effects of an online problem based learning course on content knowledge acquisition and critical thinking skills. *Computers and Education*, v. 53, n. 1, p. 132-141, 2009.

VILA, A.; CALLEJO, M. L. Matemática para aprender a pensar: o papel das crenças na resolução de problemas. Porto Alegre: Artmed, 2006.

WALLACE, R. ALICE: Artificial Linguistic Internet Computer Entity: *The A.L.I.C.E A.I. Foundation* [*Blog*]. [S.l.: S.n.], 1995. Disponível em: <alicebot.blogspot.com>. Acesso em: 11 jul. 2014.