Revista
CNEC EDUCAÇÃO

Easy AIML - Gerador de códigos AIML

Clóvis Silveira[1]

Resumo: Devido a necessidade de uma grande quantidade de códigos AIML para aumentar a base de conhecimento de chatbots há uma demanda para gerar códigos de forma mais rápida e fácil. Este artigo apresenta o protótipo Easy AIML, um gerador de códigos AIML desktop, que abrange as tags <pattern>, <template>, <random>, <that>, <star> e <srai>, onde os códigos são gerados através de templates com as tags necessárias e entradas do usuário. O protótipo visa reduzir o tempo gasto programando em AIML e facilitar o processo de escrita de códigos. Para validação do protótipo, realizou-se um experimento com estudantes em uma faculdade privada na cidade de Gravataí. Os testes e validações envolveram a criação de um diálogo usando o Easy AIML e a digitação e elaboração do código do mesmo diálogo manualmente. Resultados do estudo apontam que com o uso do Protótipo Easy AIML é muito mais ágil e rápido gerar códigos em AIML a fim de aumentar uma base de conhecimento de um chatbot.

Palavras-chave: chatbot; chatterbots; mineração de textos; agentes conversacionais, corpus.

Abstract: Due to the need for a large amount of AIML codes to increase the chatbots knowledge base there is a demand to generate codes faster and easier. This article presents the Easy AIML prototype, an AIML desktop code generator, which encompasses the tags <pattern>, <template>, <random>, <that>, <star> and <srai> tags, where codes are generated through templates with the necessary tags and user inputs. The prototype aims to reduce the time spent programming in AIML and facilitate the process of writing codes. To validate the prototype, an experiment was conducted with students at a private college in the city of Gravataí. Tests and validations involved creating a dialog using Easy AIML and typing and crafting the same dialog code manually. Results of the study indicate that with the use of the Easy AIML Prototype it is much faster and faster to generate codes in AIML in order to increase a knowledge base of a chatbot.

Keywords: chatbot; chatterbots; mining of texts; conversational agents, corpus

Introdução

Um agente conversacional consegue interagir com seus usuários utilizando linguagem natural, imitando o diálogo entre seres humanos. Porém, é necessário um grande volume de trabalho manual por parte dos autores de conteúdo para a criação de sua base de diálogos (Santos, 2016).

Ao contrário do conhecimento convencional de programação estruturada, onde menos código é geralmente melhor e considerado boa prática. Em AIML quanto maior for o código, melhor (Wallace, 2003). Isso significa que o chatbot, programa que conduz uma conversa por áudio ou texto com o intuito de simular

[1] Faculdade CNEC Gravataí. UNICNEC - Centro Universitário Cenecista de Osório.

como um humano se comporta como um parceiro conversacional, terá mais conteúdo em sua base de conhecimento, conseguindo entender mais frases e palavras digitadas pelo usuário e sabendo como responder às respectivas entradas.

Por ser exigido uma grande quantidade de código AIML para um chatbot ter um bom funcionamento, acaba-se necessitando muito tempo e desempenho para a criação do código, levando em conta o trabalho constante de criação de códigos novos para ampliar a base de diálogos do chatbot (Wallace, 2003).

A fim de contribuir com a agilidade do código em AIML, este trabalho tem como objetivo auxiliar na criação de códigos AIML para chatbots através de um gerador de códigos a partir de uma interface. Com isto será diminuído o tempo gasto codificando e consequentemente aumentará a quantidade de conteúdo inserido no chatbot no mesmo intervalo de tempo. Para isso, foi criado um protótipo com a finalidade de gerar toda a parte de tags dos códigos AIML automaticamente, onde é apenas necessário inserir o que é desejado para o chatbot aprender, assim agilizando o processo de desenvolvimento.

O teste foi realizado no Projeto ABC Digital do curso de Sistemas de Informação da Faculdade CNEC Gravataí, com uma turma de 9 alunos, onde programaram em AIML puro e com o auxílio do Easy AIML e foi cronometrado o tempo individual de cada questão para cada aluno, preenchendo um questionário para validar o protótipo Easy AIML baseado na experiência dos alunos.

Para uma melhor organização deste artigo ele foi dividido da seguinte forma: A seção 2 vai apresentar a fundamentação teórica que de uma forma conceitual traz os métodos e tecnologias que foram utilizados no desenvolvimento do protótipo; Na seção 3 a solução proposta será apresentada, mostrando o desenho da solução, as tecnologias envolvidas e o desenho das telas; A seção 4 vai apresentar os testes realizados para a validação do protótipo e os seus resultados; E por fim a seção 5 traz a conclusão, onde é gerada uma análise com base no estudo realizado.

Fundamentação Teórica

Neste capítulo serão abordadas as tecnologias utilizadas na elaboração deste artigo e do protótipo.

Agentes conversacionais

Quando um agente inteligente tem a capacidade de manter uma conversa com um usuário humano, ele é considerado um agente conversacional. Um agente conversacional que está inserido em um ambiente inteligente de aprendizagem deve ter um domínio do conteúdo e saber as necessidades do usuário, para assim guiá-lo nos conceitos e tarefas a serem resolvidas (Konzen, 2011).

Um agente possui uma base de conhecimento composta por perguntas e respostas sobre um determinado assunto, em que o agente possui domínio. O agente mantém uma conversa com o usuário através de perguntas feitas pelo estudante, por exemplo, representando um tutor pedagógico que provê suporte ao ensino de conteúdos (Paschoal, 2016).

Conhecido como chatbots são tipicamente usados em sistemas de diálogos com vários propósitos práticos incluindo serviços ao consumidor e para aquisição de informações (Abu Shawar e Atwell, 2005). Um dos programas recentes mais notável é a A.L.I.C.E. (Artificial Linguistic Internet Computer Entity), também conhecido como Alicebot, que utiliza a linguagem de marcação AIML.

A.L.I.C.E. é um chatbot de inteligência artificial que utiliza um processamento de linguagem natural. O programa usado para criar Alicebot está disponível de graça e é open source. (Wallace, 2003).

ELIZA foi um dos primeiros chatbots criado de 1964 até 1966. Ele foi desenvolvido para demonstrar a superficialidade na comunicação entre homem e máquina. ELIZA simulava conversas usando uma metodologia de correspondência de padrões (pattern matching) e substituição que dava aos

usuários uma ilusão de entendimento por parte do programa (Weizenbaum, 1966).

PARRY é um chatbot que tenta simular uma pessoa com esquizofrenia paranóica. Foi implementado um modelo de comportamento de uma pessoa com a doença baseando-se em conceitos, conceituações e crenças. Além de ter uma estratégia conversacional, sendo um programa mais sério e avançado do que ELIZA (Colby, 1981).

AIML (Artificial Intelligence Markup Language)

AIML, ou linguagem de marcação de inteligência artificial, permite que seja inserido conhecimento em chatbots baseados na tecnologia A.L.I.C.E., fazendo possível a customização de um Alicebot ou a criação de um.

A linguagem AIML é específica para sua função como um agente conversacional. Ela é baseada em técnicas de correspondência de padrões sem qualquer capacidade de raciocínio (Augello et al 2012).

A unidade básica de conhecimento em AIML é chamada de categoria. Cada categoria possui uma pergunta de entrada, uma resposta de saída, e um contexto opcional. A pergunta é chamada pattern (padrão), a resposta é chamada template (modelo) e os dois tipos de contextos opcionais são that ou topic, tornando possível o chatbot distinguir tópicos diferentes e responder de acordo (Wallace 2003). A figura 1 apresenta um exemplo do código AIML:

```
<category>
<pattern>Sim</pattern>
<that>Você gosta de filmes?</that>
<template>Qual o seu filme favorito?</template>
</category>
```

Figura 1. Exemplo de código AIML

Conforme a figura 1, pode-se observar que o chatbot interpreta o “Sim” digitado pelo usuário dentro do contexto de sua pergunta anterior “Você gosta de filmes?” e então lança uma outra pergunta baseada nessa resposta.

Solução proposta

O foco do Easy AIML é facilitar e agilizar a programação em AIML através da geração automática de suas tags, tendo uma interface simples e de fácil entendimento, resultando em uma experiência agradável a qualquer usuário, até mesmo pessoas que desconhecem da linguagem ou que nunca programaram anteriormente, que venha utilizá-lo.

Um agente conversacional tem a capacidade de manter uma conversa com um usuário humano, tendo inserido neles conteúdo de um determinado assunto para ele conseguir responder perguntas. Uma forma para inserir conhecimento em um agente conversacional é programando com a linguagem de marcação AIML, que utiliza uma técnica de correspondência de padrões através de tags. A linguagem AIML possui diversas tags, elas utilizam regras para ensinar um determinado agente conversar.

Muitas pessoas gastam parte do tempo criando essas tags, assim a solução proposta é agilizar a criação de códigos gerando automaticamente as tags, com isso ajudando a ensinar o agente com mais rapidez. Além de que pessoas não familiarizadas com a linguagem AIML podem ensinar ou fornecer conhecimento para o agente.

Este estudo vai abranger as tags <pattern>, <template>, <random>, <star>, <that> e <srai> como delimitador do estudo. Justifica-se o desenvolvimento das respectivas tags porque não encontrou-se outros softwares com tal finalidade. Vale salientar que o AIML tem uma diversidade outras tags, tais como <set>, <get>, <topic>, <think> com um total de 22 tags mais variações, as quais ficarão para o desenvolvimento em trabalhos futuros.

Desenho da solução

O Easy AIML é uma aplicação desktop desenvolvida em Java. Como pode ser visualizado na Figura 2 o usuário abre o programa onde terá os campos a serem preenchidos de acordo com o que for desejado ensinar ao chatbot.

Após preencher os campos e mandar o programa gerar o código AIML, um método utilizando a biblioteca Freemarker será executado buscando um template com as tags necessárias para gerar o código AIML e retorná-lo ao usuário.

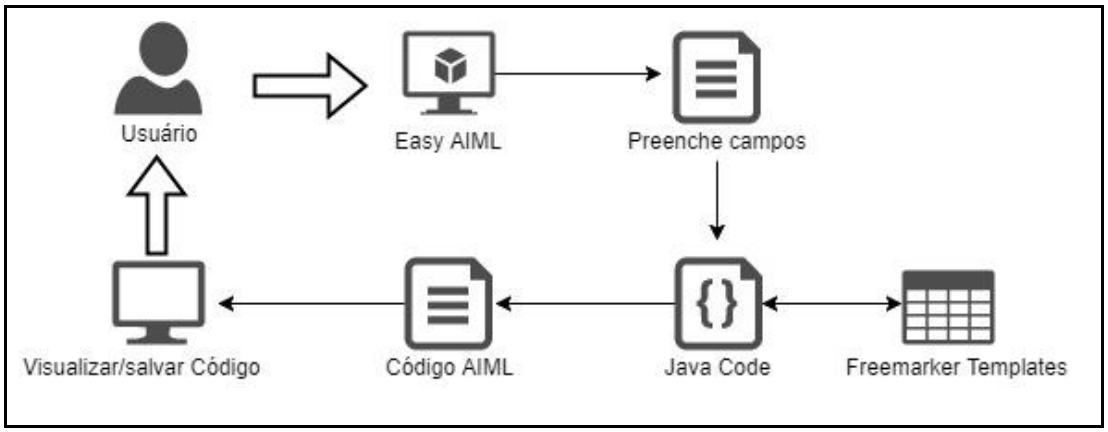

Figura 2. Desenho da Solução

Com o código AIML gerado o usuário pode visualizá-lo e exportar o código para um arquivo .aiml para então inseri-lo no seu chatbot para aumentar o seu conhecimento.

Ambiente de Desenvolvimento

O ambiente de desenvolvimento no qual o protótipo será desenvolvido é formado por 3 itens: A IDE NetBeans, a linguagem Java e a biblioteca Freemarker.

NetBeans IDE

O NetBeans IDE é um ambiente de desenvolvimento grátis, de código aberto, onde é possível programar de forma rápida e fácil em Java, JavaScript, HTML5, PHP, C/C++ e mais.

O Editor do NetBeans combina palavras e colchetes e destaca código fonte de forma sintática e semântica. Ele permite que você facilmente refatore código, com uma série de ferramentas úteis, enquanto também fornece modelos de código, dicas de codificação e geradores de código (NetBeans.org, 2017).

Java

Java é uma linguagem de programação com múltiplos propósitos, baseada em classes, orientada a objetos, e é compilada para um bytecode que é interpretado por uma máquina virtual.

A linguagem de programação Java pode ser usada para programar tanto para desktop quanto para web e mobile (Oracle.com, 2017).

Apache Freemarker

Freemarker é uma biblioteca Java que gera textos baseado em templates e dados dinâmicos (Freemarker.apache.org, 2017).

Os templates são escritos em FreeMarker Template Language (FTL), que é uma linguagem especializada simples.

Telas

O Easy AIML é uma aplicação desktop, com uma tela principal para a escolha de criação de um código AIML padrão, criar a função específica da tag <that> ou para criar uma função Srai, todas com uma descrição, conforme a Figura 3.

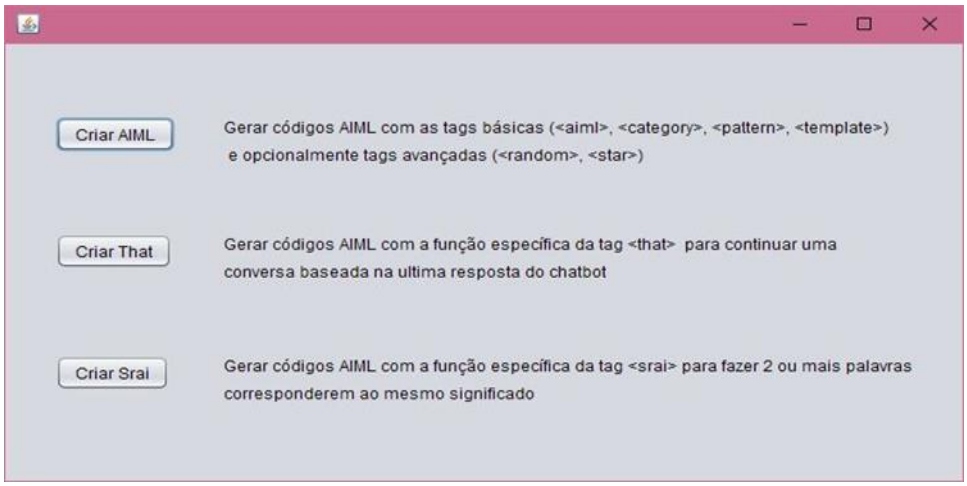

Figura 3. Tela principal EasyAIML

Após selecionar a opção de criar AIML o programa abrirá uma nova tela onde o usuário deve digitar a pergunta (pattern), selecionar caso queira incluir a tag <random> ou não, e digitar a resposta (template), caso a opção Sim de Random esteja selecionada deve-se separar as respostas em linhas diferentes. Também é possível criar a tag <star/>, basta marcar a opção Sim e colocar um asterisco (*) na resposta (template) que ela será gerada e agirá de acordo com a pergunta (pattern).

Ao clicar em Gerar AIML o programa irá buscar um template com as tags AIML, gerar o código de acordo com o que foi preenchido e mostrar na área do Código AIML, se o usuário desejar, ele poderá exportar o código para um arquivo .aiml clicando em Exportar código, como pode ser observado na Figura 4.

Figura 4. Tela para criação de código AIML do EasyAIML

Na Figura 5 é mostrado a tela para criar a função That, onde a partir de uma resposta anterior se da continuação para uma conversa, ou seja, uma resposta baseada em um contexto. Então é digitado a pergunta (Pattern) dentro do contexto (That) e em seguida a nova resposta (Template). E clicando em Gerar AIML o código será gerado e apresentado no campo Código AIML, então podendo ser exportado para um arquivo .aiml clicando no botão Exportar Código.

Figura 5. Tela para criação da função That em AIML do EasyAIML

Na Figura 6 é apresentada a opção de criar um código com a função Srai, que permite identificar ao chatbot que duas questões são similares, ou seja, apesar de formuladas de um modo diferente, tem um mesmo significado (Tutorialspoint.com, 2017). O usuário digita a pergunta e diferentes frases/palavras que significam a mesma coisa em linhas diferentes, e assim como na tela de criação de código padrão AIML, ao clicar em Gerar Srai o programa busca um template no banco de dados que irá gerar o código de acordo com o que foi digitado e então será mostrado na área do Código AIML, e caso necessário é possível exportar o código gerado para um arquivo .aiml clicando no botão Exportar código.

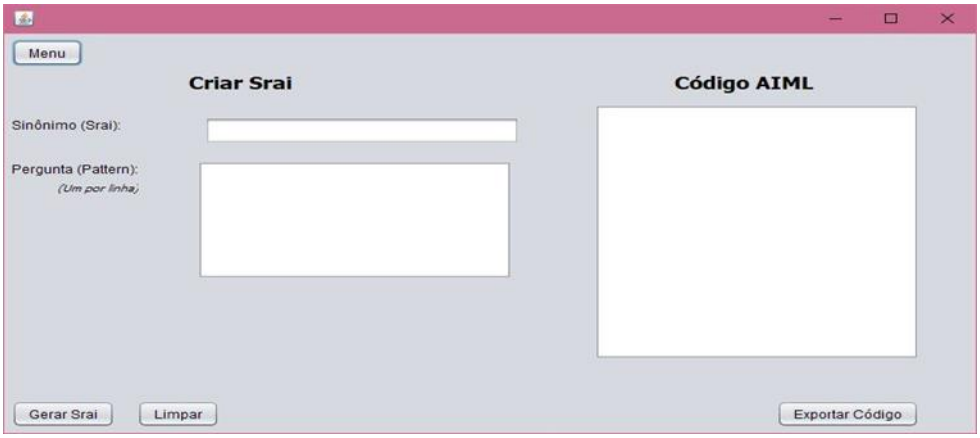

Figura 6. Tela para criação da função Srai em AIML do EasyAIML

Testes e Validação

Os testes e validação da pesquisa serão norteados pela metodologia de pesquisa de natureza aplicada. Conforme Prodanov (2013), uma pesquisa aplicada objetiva gerar conhecimentos para aplicação prática dirigidos à solução de problemas específicos. Envolve verdades e interesses locais.

Já do ponto de vista de seus objetivos a pesquisa é uma pesquisa exploratória. Pesquisa exploratória é quando a pesquisa está numa fase preliminar, tem como finalidade proporcionar mais informações sobre o assunto que será investigado, possibilitando facilitar a delimitação do tema da pesquisa; orientar a fixação dos objetivos e a formulação das hipóteses. Assume, em geral, as formas de pesquisas bibliográficas e estudos de caso (Prodanov, 2013).
O experimento foi realizado em um Projeto de Inclusão Digital na Faculdade CNEC Gravataí, com uma turma de 9 alunos com idade entre 14 e 19 anos os quais estão estudando conteúdos relacionados a lógica de Programação. Durante o experimento os alunos digitaram códigos AIML explorando as tags template, pattern, random, star, that, srai) primeiramente sem o auxílio do Easy AIML para depois criar códigos com o seu auxílio.

Os testes e validações se dão a partir de um questionário de 10 questões aos alunos onde tinha dois tipos de questões, um para digitar o código manualmente e outro para utilizar o Easy AIML para gerar o mesmo código, então foi cronometrado o tempo levado de cada aluno para a resolução de cada exercício individual a fim de validar o protótipo Easy AIML. Como se pode ver na Figura 7 os resultados mostram a diferença de performance e o tempo levado, média em segundos de todos alunos, de cada exercício com o foco em funções diferentes para criar um mesmo código manualmente e com o apoio do Easy AIML.

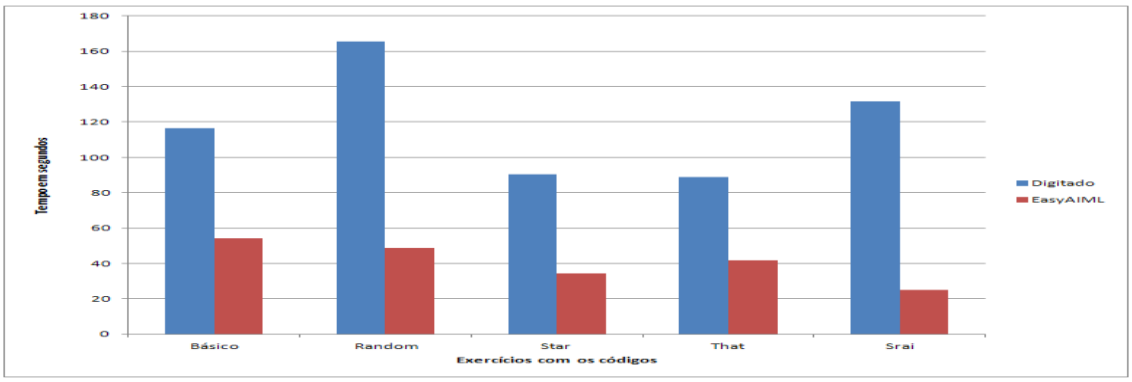

Figura 7. Gráfico do tempo levado para resolver os exercícios individuais.

É possível ver a grande diferença para gerar o mesmo código digitando e utilizando o Easy AIML comparando o tempo necessário, apresentando cada questão separada.

Figura 8. Gráfico da diferença tempo levado para resolver todos os exercícios.

Na Figura 8 apresenta-se o tempo total de todos os alunos para a realização de todos exercícios, com uma média de 13.2 minutos para a realização de todo o teste foi necessário 9.8 minutos para terminar os exercícios de digitação de código e 3.4 minutos para os exercícios de geração do código através do Easy AIML, sendo quase três vezes mais rápido a geração de um mesmo código utilizando o protótipo.

Isso deixa mais evidente a facilidade de uso do protótipo mesmo com pessoas que desconhecem a linguagem e a agilidade do protótipo para acelerar o processo de programação na linguagem AIML a fim de fornecer novas bases de conhecimentos para agentes conversacionais.

Conclusão

Chatbots são agentes importantes usados em sistemas de diálogos, com varios

propósitos prácticos incluindo serviço ao consumidor ou aquisição de informações. Eles são muito usados como assistentes virtuais, substituindo a necessidade de uma pessoa para tal tarefa, tornando essas interações mais rápidas e práticas.

A dificuldade de criar um bom chatbot é a necessidade de uma grande quantidade de código, quanto maior o código de um agente conversacional mais conhecimento ele possui para interagir com algum usuário, é necessário uma grande quantidade de tempo para programar e muito desse tempo acaba sendo perdido escrevendo tags que geralmente são padrão no AIML.

Surge então uma demanda de uma ferramenta como o Easy AIML para agilizar a criação de códigos AIML gerando automaticamente toda a parte das tags, sendo apenas necessário digitar o que é desejado ensinar ao chatbot. Assim tirando o tempo perdido escrevendo as tags padrões manualmente, diminuindo o tempo no processo de desenvolvimento de um chatbot.

Além disso, por o Easy AIML possuir uma interface simples e com instruções, é possível pessoas que não sabem programar em AIML criar códigos facilmente com a utilização dele.

Após ser apresentado os resultados dos testes é possível concluir que o Easy AIML cumpriu a sua proposta de acelerar o processo de programação em AIML e ter uma interface simples e intuitiva para pessoas que desconhecem da linguagem AIML, facilitando a implementação e incrementação de conhecimento em chatbots.

Referências

Abu Shawar, B. & Atwell, E. (2005) “Using corpora in machin-learning chatbot systems.” Disponível em: http://www.academia.edu/9301712/Using_corpora_in_machine-learning_chatbot_systems> Acesso em 14 de junho, 2017.

AIML Tutorial. (2017) Disponível em:

<https://www.tutorialspoint.com/aiml/index.htm> Acesso em 20 de abril, 2017.
Augello, A., Pilato, G., Machi, A., Gaglio, S. (2012) "An approach to enhance chatbot semantic power and maintainability: Experiences within the frasi project" Disponível em: <https://www.researchgate.net/profile/Agnese_Augello/publication/261350701_An_Approach_to_Enhance_Chatbot_Semantic_Power_and_Maintainability_Experiences_within_the_FRASI_Project/links/558e6e1d08ae15962d8963d0/An-Approach-to-Enhance-Chatbot-Semantic-Power-and-Maintainability-Experiences-within-the-FRASI-Project.pdf> Acesso em 2 de junho, 2017.

Colby, K. (1981) "Modeling a paranoid mind" Disponível em: <https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/modeling-a-paranoid-mind/7BF489C5A50373179EB7A3D930C13951> Acesso em 19 de maio, 2017.
Colby, K. (1999) "Human-computer conversation in a cognitive therapy program." Disponível em: <https://link.springer.com/chapter/10.1007%2F978-1-4757-5687-6_3> Acesso em 19 de maio, 2017.

Gasperis, G. (2016) "PyGenbot for IoT: a demonstration of how to generate any restricted stateless AIML FAQ-chatter bot from text files" Disponível em: <https://www.scopus.com/record/display.uri?eid=2-s2.0-84985947526&origin=resultslist> Acesso em 01 de junho, 2017.

Gasperis, G., Chiari, I., Florio, N. (2013) "AIML Knowledge Base Construction from Text Corpora" Disponível em: <https://www.scopus.com/record/display.uri?eid=2-s2.0-84867569862&origin=resultslist> Acesso em 01 de junho, 2017.

Konzen, A., Oliveira, O., Kist, L., Rosa, A., Moraes, L., Freitas, C., Müller, D. e Axt, M. (2011) "Maga Vitta – agente conversacional aplicado ao jogo educacional Città". Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/1579> Acesso em 27 de abril, 2017.

NetBeans IDE Features. Disponível em: <https://netbeans.org/features/index.html> Acesso em 11 de maio, 2017.

Paschoal, L., Chicon P. e Falkembach, G. (2016) "UBIBOT: UM AGENTE CONVERSACIONAL CIENTE DO CONTEXTO DE APRENDIZAGEM DO USUÁRIO" Disponível em: <http://www.seer.ufrgs.br/renote/article/view/67362> Acesso em 27 de abril, 2017.
Java Documentation. Disponível em: <https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html> Acesso em 11 de maio, 2017.

Pilato, G., Augello, A., Gaglio, S. (2011) "A modular architecture for adaptive chatbots" Disponível em: <https://www.hindawi.com/journals/isrn/2012/363840/> Acesso em 19 de maio, 2017.

Prodanov, C., Freitas, E. (2013) "Metodologia do trabalho científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico " Disponível em:

<https://docente.ifrn.edu.br/valcinetemacedo/disciplinas/metodologia-do-trabalho-cientifico/e-book-mtc> Acesso em 29 de junho, 2017.
Santos, F. (2016) “Um modelo semântico para integração automática de conteúdo com um agente conversacional”. Disponível em: <http://www.repositorio.jesuita.org.br/handle/UNISINOS/5243> Acesso em 27 de abril, 2017.

Shawar, B.A., Atwell, E. (2015) “ALICE chatbot: Trials and outputs” Disponível em: <https://www.scopus.com/record/display.uri?eid=2-s2.0-84953320591&origin=resultslist> Acesso em 01 de junho, 2017.

What is Apache Freemarker. Disponível em: <https://freemarker.apache.org/> Acesso em 23 de maio, 2017.

Wallace, R. (2003) “The Elements of AIML Style” Disponível em: <http://www.alicebot.org/style.pdf> Acesso em 20 de abril, 2017.

Weizenbaum, J. (1966) “ELIZA—a computer program for the study of natural language communication between man and machine” Disponível em: <http://dl.acm.org/citation.cfm?id=365168> Acesso em 19 de maio, 2017.