



Programa Interdisciplinar de Pós-Graduação em
Computação Aplicada
Mestrado Acadêmico

Fabio Rodrigues dos Santos

Um Modelo Semântico para Integração Automática de
Conteúdo com um Agente Conversacional

São Leopoldo, 2016

FABIO RODRIGUES DOS SANTOS

UM MODELO SEMÂNTICO PARA INTEGRAÇÃO AUTOMÁTICA DE
CONTEÚDO COM UM AGENTE CONVERSACIONAL

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Computação Aplicada, pelo Programa de Pós-Graduação em Computação Aplicada - PIPCA da Universidade do Vale do Rio dos Sinos – UNISINOS.

Orientador: Prof. Dr. Sandro José Rigo

São Leopoldo

2016

S237m Santos, Fabio Rodrigues dos
Um modelo semântico para integração automática de conteúdo com um agente conversacional / por Fabio Rodrigues dos Santos. – 2016.
109 f.: il. ; 30 cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2016.
“Orientação: Prof. Dr. Sandro José Rigo.”

1. Agente conversacional. 2. Web semântica. 3. Sistema de gerenciamento de conteúdo web. I. Título.

CDU: 004.738.5:004.8

Fabio Rodrigues dos Santos

Um modelo semântico para integração de conteúdo com um agente conversacional

Dissertação apresentada à Universidade do Vale do Rio dos Sinos – Unisinos, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Aprovado em 26 / 02 / 2016

BANCA EXAMINADORA

Prof^a. Dr^a. Milene Selbach Silveira - PUCRS

Prof. Dr. Jorge Luis Victória Barbosa - UNISINOS

Prof. Dr. Sandro José Rigo (Orientador)

Visto e permitida a impressão
São Leopoldo,

Prof. Dr. Sandro José Rigo
Coordenador PPG em Computação Aplicada

AGRADECIMENTOS

Primeiramente a Deus, este ser incognoscível, inexplicável e imensurável que além de ser onipresente, onipotente e onisciente ainda acha de ser onimisericordioso ao me conceder tantas oportunidades.

Agradeço ao meu orientador, professor Dr. Sandro Rigo, um verdadeiro guia, por ter trilhado o caminho e usar sabiamente sua experiência para apontar a direção que deveria tomar.

Agradeço a minha esposa, por sempre me apoiar, me incentivar e acreditar em mim, por me dar a retaguarda e assumir minhas responsabilidades durante minha ausência, sem ela nada disso seria possível, ela e minhas filhas são o meu refugio e o maior estímulo para edificar-me.

Ao meu pai e a minha mãe, por me ensinarem o valor do trabalho duro e da perseverança.

Ao IFRR (Instituto Federal de Roraima) pela concessão da bolsa de ajuda de custo durante todo o período de realização deste mestrado, por acreditar em seus servidores e investir em sua qualificação pessoal e profissional.

Aos meus colegas e amigos, por trilharem junto comigo este importante desafio. A Unisinos por nos aceitar como alunos, oferecer qualidade de ensino, um ótimo corpo docente, infraestrutura e conforto além da hospitalidade gaúcha.

Os meus sinceros e humildes agradecimentos a todos vocês!

RESUMO

Um agente conversacional é capaz de interagir com usuários em linguagem natural, imitando o diálogo realizado entre seres humanos. Entretanto, a necessidade de trabalho manual por parte dos autores de conteúdo para a construção da sua base de diálogos não o torna muito atrativo para ser integrado a websites. Seria possível facilitar a criação de novos diálogos e automatizar a atualização dos mesmos usando como base os recursos disponibilizados pela Web Semântica para integrar o agente conversacional a um Sistema de Gerenciamento de Conteúdo Web (SGCW) de forma que seu conteúdo seja aproveitado.

O trabalho aqui proposto descreve o modelo para utilização de informações de SGCW's em agentes conversacionais com auxílio da web semântica, denominado Eduardo. O principal diferencial está na automatização da atualização dos diálogos, pois os mesmos são extraídos do SGCW para uma base de dados de triplas, que são consultadas para geração dos diálogos, caso haja alguma atualização na base do SGCW, basta refazer os procedimentos que já foram automatizados, para que os diálogos sejam atualizados, evitando assim o trabalho manual de edição dos arquivos escritos em linguagem AIML (*Artificial Intelligence Markup Language*). Foi desenvolvido um protótipo deste modelo, que permite a integração dinâmica de conteúdo com um Agente Conversacional. O protótipo foi avaliado quanto à capacidade de gerar as informações necessárias para os diálogos, acessando as páginas Web e representando seu conteúdo nos formatos RDFa e AIML. Em um segundo momento, ensaios de interação foram realizados com usuários nos componentes do Agente Conversacional, para avaliar sua funcionalidade, aceitação e demais aspectos uso.

Palavras-chave: Agente Conversacional, Web Semântica, Sistema de Gerenciamento de Conteúdo Web.

ABSTRACT

A conversational agent is able to interact with users in natural language, imitating the dialogue held between humans. However, the need for manual labor by the authors content for the construction of dialogues base makes the not too attractive to be integrated into websites. It would be possible to facility the creation of new dialogues and automate the update them using as a basis the resources provided by the Semantic Web to integrate conversational agent to a Web Content Management System (WCMS) so that you content is used.

The work proposed here describes the model for use of WCMS's information in conversational agents with semantic web aid, called Eduardo. The main difference is in automating the update of dialogues, as they are extracted from WCMS for a triple database, which are referred to generation of dialogues, if there is any update on the basis WCMS, just redo the procedures that have been automated so that the dialogues are update, thus avoiding manual labor editing of files written in AIML (Artificial Intelligence Markup Language). It developed a prototype of this model which allows dynamic content integration with a conversational agent. The prototype was evaluated for ability to generate the information necessary for the dialogue, accessing the Web pages and their content representing in RDFa and AIML formats. In a second step, interaction assays were performed with users in the components of conversational agent to assess its functionality, acceptance and other aspects of use.

Keywords: Conversational Agent, Semantic Web, Web Content Management System.

LISTA DE FIGURAS

Figura 1 – Arquitetura de núcleo de um chatbot tradicional	29
Figura 2 – Arquitetura básica de um sistema de diálogo.....	30
Figura 3 – Exemplo de código em Linguagem AIML.....	32
Figura 4 – Exemplo básico de código com PyAIML	32
Figura 5 – Grafo com estrutura de uma declaração RDF.....	37
Figura 6 – Pilha de conceitos sobre RDFa API e RDF API.....	39
Figura 7 - VOX: <i>Multimodal Interface Architecture</i>	43
Figura 8 – Vox: módulo de acesso a DBPedia.....	44
Figura 9 - Arquitetura do sistema de agente guia de laboratório interativo	46
Figura 10 – Exemplo de Extensão AIML.....	47
Figura 11 - Fluxo de trabalho do processo de geração de diálogo de um Agente Conversacional.....	48
Figura 12 – Arquitetura do sistema RDFaCE	50
Figura 13 – As quatro formas do autor visualizar o conteúdo	51
Figura 14 – Trecho de diálogo com sensibilidade a troca de contexto.....	52
Figura 15 – Arquitetura proposta para Agente conversacional com base em Web Semântica	57
Figura 16 – Editor de texto Semântico <i>WYSYWIM</i>	60
Figura 17 - Metadata Extractor.....	61
Figura 18 - <i>Semantic AIML Interpreter</i>	62
Figura 19 - Expansão da Ontologia.....	63
Figura 20 - Casos de Uso	64
Figura 21 - Implementação do Modelo Eduardo no Plone SGCW	68
Figura 22 - Diagrama de Classes do <i>Semantic AIML Interpreter</i>	72
Figura 23 - Exemplo de uma consulta SPARQL em um arquivo AIML	73

Figura 24 - Classe responsável por servir de Proxy para NLP API's	74
Figura 25 – Parte da ontologia para o Mapeamentos dos Conselhos.....	77
Figura 26 - Atendimento a expectativa dos entrevistados.....	79
Figura 27 - Interface do componente WYSIWYM.....	82
Figura 28 - <i>Fact Browser</i>	83
Figura 29 - Código HTML5/RDFa gerado pelo componente WYSIWYM	83
Figura 30 – Código Turtle extraído em teste de validação no RDFaPlay.....	84
Figura 31 - Ambiente proposto para extensão da ontologia Schema.org.....	86
Figura 32 - Protégé contendo algumas entidades e propriedades utilizadas no protótipo	87
Figura 33 – Extensão da Ontologia Schema.org.....	88
Figura 34 - Primeira conversa com o Agente	89
Figura 35 - Interface de Chat do Eduardo	91
Figura 36 - Resumo da Avaliação de Aceitabilidade	94

LISTA DE TABELAS

Tabela 1 - Elementos básicos da AIML.....	31
Tabela 2 - Terminologia Básica RDF	36
Tabela 3 – Tabela comparativa dos Trabalhos Relacionados.....	54
Tabela 4 - RUP.....	70
Tabela 5 – Descrição das ferramentas do RDFaCE	82
Tabela 6 – Resultado da Avaliação para Facilidade de Uso	93
Tabela 7 – Resultado da Avaliação para Percepção de Utilidade dos Usuários.....	93

LISTA DE ABREVIATURAS E SIGLAS

AIML - Artificial Intelligence Markup Language
API - Application Programming Interface
DAM - Digital Asset Management
DMS - Document Management System
DOM - Document Object Model
ECA - Embodied Conversational Agent
ECM - Enterprise Content Management
e-Gov - Governo Eletrônico
GRDDL - Gleaning Resource Descriptions from Dialects of Languages
HTML - HyperText Markup Language
IFRR - Instituto Federal de Roraima
IHC - Interação Humano Computador
NLTK - Natural Language Toolkit
OWL - Web Ontology Language
PLN - Processamento de Linguagem Natural
RDF - Resource Description Framework
RDFa - Resource Description Framework in Attributes
RDFaCE - RDFa Content Editor
RDFS - Resource Description Framework Schema
SEO - Search Engine Optimization
SGCW – Sistema de Gerenciamento de Conteúdo Web
SOAP - Simple Object Access Protocol
SPARQL - SPARQL Protocol and RDF Query Language
TAM - Technical Architecture Modeling
MAT - Modelo de Aceitação de Tecnologia
URI - Uniform Resource Identifier
W3C - World Wide Web Consortium
WCM - Web Content Management⁵
WCMS - Web Content Management System
WYSIWYG - What You See Is What You Get
WYSIWYM - What You See Is What You Mean
XML - eXtensible Markup Language

XSL - Extensible Stylesheet Language

SUMÁRIO

1	INTRODUÇÃO	20
1.1	Motivação	22
1.2	Questão da Pesquisa	23
1.3	Objetivos	24
1.4	Metodologia	24
1.5	Organização do documento	25
2	EMBASAMENTO TEÓRICO	26
2.1	Sistemas de Gerenciamento de Conteúdo Web	26
2.2	Agentes Conversacionais	27
2.2.1	Comparação de <i>Chatbot</i> com Sistemas de Diálogo	28
2.2.2	AIML - <i>Artificial Intelligence Markup Language</i>	30
2.2.3	PyAIML um Interpretador Python para AIML	32
2.3	Processamento de Linguagem Natural	33
2.4	Web Semântica	36
2.4.1	RDF (<i>Resource Description Framework</i>)	36
2.4.2	<i>RDF Serialization</i> : RDF/XML Sintaxe	37
2.4.3	Microformatos, RDFa, GRDDL e RDFa API	37
2.4.4	RDFS	40
2.4.5	OWL	40
2.4.6	SPARQL	40
3	TRABALHOS RELACIONADOS	42
3.1	VOX System: a semantic embodied conversational agent exploiting linked data	42
3.2	Embodied Conversational Agent Based on Semantic Web	45
3.3	Building an AIML Chatter Bot Knowledge-Base Starting from a FAQ and a Glossary	48
3.4	The RDFa Content Editor - From WYSIWYG to WYSIWYM	49
3.5	Gerenciamento de Diálogo Baseado em Modelo Cognitivo para Sistemas de Interação Multimodal	52
3.6	Comparação entre os modelos estudados	53
4	MODELO EDUARDO	56

4.1 Modelo Conceitual	56
4.1.1 <i>WYSIWYM</i>	58
4.1.2 Metadata Extractor	61
4.1.3 Semantic AIML Interpreter	62
4.1.4 Expansão da Ontologia	63
4.2 Requisitos e Definição do Perfil dos Usuários	64
4.2.1 Casos de Uso	64
4.3 Demais considerações acerca do modelo	65
5 DESENVOLVIMENTO DO PROTÓTIPO	68
5.1 Implementação de componentes	69
5.1.1 Implementação do <i>Semantic AIML Interpreter</i>	71
5.1.2 Implementação do <i>WYSIWYM</i> no Plone SGCW	73
5.1.3 Implementação do <i>Script Metadata Extractor</i>	75
5.2 Ontologia	76
6 AVALIAÇÃO E TESTE	78
6.1 Teste Antecipado	78
6.2 Avaliação de Funcionalidade	80
6.2.1 Cenário 1 – Anotação do Conteúdo com RDFa	81
6.2.2 Cenário 2 – Extração dos Dados	83
6.2.3 Cenário 3 – Expansão da Ontologia e SPARQL <i>endpoint</i>	85
6.2.4 Cenário 4 – Conversando com o Agente	88
6.3 Avaliação de Aceitabilidade	92
7 CONCLUSÃO	96
7.1 Limitações e Trabalhos Futuros	97
REFERÊNCIAS	98
ANEXO A DADOS RECEBIDOS DOS PARTICIPANTES PARA ELABORAR OS EXPERIMENTOS	104
ANEXO B QUESTIONÁRIO PARA A AVALIAÇÃO DE USABILIDADE	106

1 INTRODUÇÃO

Há cerca de duas décadas atrás, Sistemas de Gerenciamento de Conteúdo Web (SGCW) eram considerados as ferramentas mais efetivas para comunicação na web, sendo utilizados como uma das principais formas de comunicação das corporações com seus clientes, investidores, mídia e outros. Porém com o advento da Web 2.0 (termo usado para designar a web interativa), os SGCW's passaram a dividir espaço com Blogs e Redes Sociais, nos quais os internautas são capazes de produzir seu próprio conteúdo. Além disso ampliaram as possibilidades de uso das ferramentas colaborativas, como é o caso da Wikipédia. Atualmente um usuário da Web pode gerir seu próprio espaço de conteúdo digital em tempo real com o mínimo de requisitos técnicos. Uma consequência desta situação é o fato do conteúdo online estar crescendo em uma velocidade sem precedentes (MAASS; KOWATSCH, 2012).

Um dos desafios encontrados no contexto da Web atualmente é justamente encontrar a informação desejada em meio a tanto conteúdo disponível. Como exemplo de soluções neste sentido, podem ser citados os motores de busca, tais como Google, Yahoo e Bing que têm oferecido apoio na busca de respostas para questões dos usuários.

Porém as buscas realizadas na Web, quando tratadas pelos algoritmos tradicionais voltados para uso de padrões textuais, podem apresentar problemas. Por exemplo: quando um usuário pesquisa em um motor de busca a palavra SOAP, com uma ideia de resposta em mente que seria ligada ao fato de SOAP ser um padrão mantido pelo W3C (*World Wide Web Consortium*) para Web Services, o resultado obtido poderá ser diferente do esperado. Provavelmente, o usuário receberá um grande número de resultados para a pesquisa, mas a composição dos resultados pode apresentar desde marcas de detergentes, receitas de sopa ou até novelas. Depois de navegar entre várias páginas o usuário deve encontrar a Especificação SOAP do W3C. A razão desse problema é que motores de busca utilizam palavras-chave para indexar os documentos da Web (YU, 2014).

Considerando a forma pela qual o conteúdo da web é indexada, em poucos anos essas informações poderão estar perdidas na *deep web*, ou seja, vão estar

ocultas aos internautas. Uma nova geração de tecnologias está tomando forma que irá estender o alcance dos motores de busca a cantos escondidos da web, Wright (2009) afirma que quando isso acontecer, não vai apenas melhorar a qualidade dos resultados da pesquisa – que pode remodelar a forma como muitas organizações fazem negócios online. A web atual não foi projetada para facilitar a compreensão humana, pois ela é ambígua, desestruturada e seus conteúdos estão distribuídos e não interligados. A Web precisa evoluir, para melhorar a experiência dos seus usuários e para facilitar tarefas rotineiras. Para que isso aconteça, as ferramentas atuais precisam evoluir também. Essa evolução já está acontecendo, pois universidades, governos e empresas estão investindo em pesquisas no uso da Web Semântica para tornar o conteúdo da Web mais facilmente utilizável por máquinas.

Agentes conversacionais, uma área de pesquisa que não é nova em se tratando de computação, será uma das áreas que poderá se beneficiar dessa evolução da web, pois os agentes conversacionais poderão atuar como assistentes virtuais, vasculhando a web de acordo com os atributos desejados pelos usuários e trazer resultados mais relevantes às suas pesquisas.

Um campo de aplicação comum para os agentes conversacionais é o suporte a usuários da Web, através da integração desses agentes a web sites, fornecendo meios adicionais de controlar e acessar informações e funcionalidades disponíveis que vão além das interfaces tradicionais. Ao possibilitar o usuário interagir com a máquina em linguagem natural, dá a ele o poder de informar entradas fora de padrão, ou seja, de maneira irrestrita ou inesperada. No entanto, dois requisitos são considerados os mais importantes para um agente conversacional, o primeiro é a robustez em tratar entradas inesperadas e o segunda é uma forma fácil de gerar novos diálogos, tarefa que presumivelmente deve ser simples para que os autores de conteúdo não necessitem de experiência em linguística (PEREZ-MARIN; PASCUAL-NIETO, 2011, p. 1).

Devido a estas duas exigências, a funcionalidade de diálogo de agentes conversacionais é muitas vezes baseada em padrões e/ou palavras-chave. Tais técnicas se popularizaram através do *chatbot* de Joseph Weizenbaum “ELIZA” da década de 1960 (PEREZ-MARIN; PASCUAL-NIETO, 2011, p. 1).

Infelizmente, somente o uso dessas técnicas não garante a efetiva compreensão do enunciado do usuário, como também gera uma grande dependência de enormes quantidades de dados de padrões de diálogos. Estas desvantagens resultam em sistemas inflexíveis. Para superar esses problemas várias otimizações são possíveis, a partir das quais a integração de conhecimentos e métodos provenientes da Linguística e também da Linguística Computacional (*CL - Computational Linguistics* no inglês) serão especialmente úteis (PEREZ-MARIN; PASCUAL-NIETO, 2011, p. 2).

Este trabalho explora as possibilidades existentes para que os SGCW's sejam estendidos para atender a demanda do gerenciamento de conteúdo semântico, com aplicação na gestão dos repertórios de diálogo dos agentes conversacionais possibilitando aproveitar o conteúdo já existente para facilitar a criação de novos diálogos e automatizar a atualização dos mesmos.

1.1 Motivação

Este trabalho se inspira nos avanços das pesquisas relacionadas à Web Semântica e Processamento de Linguagem Natural para melhorar a experiência de usuários no âmbito dos Agentes Conversacionais Inteligentes.

Observa-se que a cada dia mais pessoas, inclusive crianças, conseguem obter resultados satisfatórios ao interagir com dispositivos tais como um *tablet*, um computador ou até mesmo interfaces de gestos. Na década de 70, quando os primeiros computadores pessoais foram lançados, somente pessoas com alto nível de conhecimento técnico conseguiam utilizar de forma proveitosa tal interação (WALKER, 1990). A multimodalidade na comunicação, característica tão natural entre humanos, pode ainda ser bastante ampliada na Interação Humano Computador (IHC) (NETO *et al.*, 2009). Essa evolução das tecnologias no âmbito da interação com os dispositivos e a multimodalidade são áreas de pesquisa que ainda podem ser exploradas e que podem contribuir no gerenciamento de diálogos dos agentes conversacionais. Segundo Hartson e Hix (1989),

Gerenciamento de diálogo [...] busca proporcionar melhorias significativas para os sistemas de interfaces humano-computador. Desenvolvida a partir dos anos 70, essa área engloba a simulação,

manutenção e execução de diálogos em sistemas conversacionais. Alguns conceitos têm grande importância para a geração de discurso entre sistemas e humanos. Muitos sistemas utilizam uma abordagem linguística com processamento de linguagem natural para entendimento da entrada e geração das respostas para o usuário.

Já com relação aos avanços das pesquisas em Web Semântica, existem benefícios evidentes da Web Semântica na Otimização de Mecanismos de Busca (*Search Engine Optimization - SEO*), capazes de refletir no aumento de visualizações de páginas web (FONSECA, 2012) e por consequência na democratização do acesso à informação. O Governo Brasileiro, através da Lei nº 12.527/2011 regulamenta o direito constitucional de acesso às informações públicas. Essa norma entrou em vigor em 16 de maio de 2012 e criou mecanismos que possibilitam, a qualquer pessoa, física ou jurídica, sem necessidade de apresentar motivo, o recebimento de informações públicas dos órgãos e entidades (BRASIL, 2014).

Esta pesquisa investiga a geração de conteúdo semântico com o apoio de Sistemas de Gerenciamento de Conteúdo Web (SGCW) para que um agente conversacional utilize desses dados estruturados em seu repertório de diálogos. Foi desenvolvido um protótipo, com objetivo de viabilizar um produto de software no domínio do e-Gov para o auxílio na promoção e transparência, no âmbito de atividades como inscrições, acompanhamento dos processos seletivos, vestibulares e concursos públicos realizados no Instituto Federal de Educação, Ciência e Tecnologia do Estado de Roraima IFRR (Instituto Federal de Roraima).

1.2 Questão da Pesquisa

Um agente conversacional é capaz de interagir com usuários em linguagem natural, imitando o diálogo realizado entre seres humanos. Entretanto, a necessidade de trabalho manual por parte dos autores de conteúdo para a construção da sua base de diálogos não o torna muito atrativo para ser integrado a websites. Desta forma considera-se neste trabalho a questão de pesquisa descrita a seguir.

No âmbito dos agentes conversacionais seria possível facilitar a criação de novos diálogos e automatizar a atualização dos mesmos usando como base os recursos disponibilizados pela Web Semântica para integrar o agente conversacional a um SGCW de forma que seu conteúdo seja aproveitado.

1.3 Objetivos

O objetivo geral dessa pesquisa é definir um modelo de publicação de conteúdo para websites, levando em consideração o uso da Web Semântica e que possibilite o consumo do conteúdo por um agente conversacional.

Para chegar a esse resultado os seguintes objetivos específicos também foram elencados:

1. Identificar e comparar as formas já existentes de publicação de conteúdo semântico;
2. Criar um Agente Conversacional capaz de consumir os metadados produzidos tanto pelos autores de conteúdo do site como os já disponíveis na Web;
3. Criar cenários para ensaios de interação com foco na validação do modelo e na avaliação de aceitabilidade.

O diferencial dessa pesquisa está identificado na automatização do processo de atualização do repertório de diálogos do agente conversacional. Este aspecto é considerado uma das dificuldades para o acesso a amplos conjuntos de informações por este tipo de recurso, sendo desta forma, um incentivo no sentido de melhoria do processo de interação entre usuários e sistemas web.

1.4 Metodologia

Este trabalho consiste em uma pesquisa com objetivo exploratório, cuja finalidade é proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito e definir um modelo arquitetônico de software. Na maioria dos casos

essas pesquisas envolvem um levantamento bibliográfico, entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado (GIL, 2008, p. 41) e análise de exemplos que “estimulem a compreensão” (Selltiz et al, 1967, p. 63 apud GIL, 2008, p. 41).

Essa familiaridade com o problema será evidenciada com base nas pesquisas de referências bibliográficas, na experimentação proporcionada pelo protótipo a ser desenvolvido e nos testes a serem realizados.

Com relação ao método de trabalho (WAZLAWICK, 2009), foram definidas quatro etapas para realizar os objetivos propostos. São elas (I) Realizar estudos sobre a aplicação de Agentes Conversacionais com base em Web Semântica no domínio de sistemas de gerenciamento de conteúdo web; (II) Descrever a arquitetura de integração das diferentes tecnologias para nortear o desenvolvimento de um protótipo; (III) Desenvolver um protótipo funcional; (IV) Realizar experimentos para coleta de dados quantitativos e qualitativos.

1.5 Organização do documento

O documento está dividido da forma descrita a seguir. No capítulo 2 é apresentado o embasamento teórico do trabalho. No capítulo 3 são apresentados trabalhos relacionados e sua análise. No capítulo 4 é apresentado o modelo proposto. No capítulo 5 é apresentado o desenvolvimento do protótipo. No capítulo 6 são apresentadas as formas de Avaliação. A conclusão, limitações e os trabalhos futuros estão descritos no capítulo 7.

2 EMBASAMENTO TEÓRICO

Esse capítulo apresenta o embasamento teórico dos pontos abordados nesse trabalho de pesquisa. A seção seguinte descreve a forma pela qual o conteúdo web tem sido gerenciado dentro das corporações. Em seguida é feita uma comparação em ordem histórica sobre as tecnologias utilizadas para construção de Agentes Conversacionais e qual a contribuição da Linguística Computacional para o desenvolvimento das pesquisas nessa área. Também será abordada a temática da Web semântica, que será imprescindível para armazenamento, estruturação de páginas web e consulta de conteúdo semântico a ser consumido pelo Agente Conversacional desenvolvido.

2.1 Sistemas de Gerenciamento de Conteúdo Web

A publicação de conteúdo nos sites contemporâneos, em grande parte, se dá através de sistemas conhecidos como Sistemas Gerenciadores de Conteúdo Web (SGCW). Estes sistemas disponibilizam um *framework* geral, que geralmente serve para gerenciar dados heterogêneos, considerando a segurança e a privacidade em ambientes multiusuários. SGCW são muitas vezes utilizados para gerenciar sites que contêm blogs, notícias, ferramentas de compras, sites corporativos e de marketing. Um SGCW tipicamente visa evitar a necessidade de codificação manual para implementação do suporte para estes diferentes tipos de sites, além de possuir componentes de apoio para necessidades específicas (SAHA; SETUA, 2014).

Hoje as empresas e as instituições de ensino estão usando SGCW para ajudá-los na entrega de informações específicas para os seus usuários. Muitos pacotes de software estão disponíveis, no entanto a seleção de um destes sistemas para uso é por vezes difícil, porque tendências e dados de avaliação nem sempre convivem nos mesmos contextos. Desenvolver um novo SGCW representa grandes desafios, especialmente se ele lida com as necessidades da organização (SAHA; SETUA, 2014).

O termo "gestão de conteúdos" possui significados diferentes no mercado de hoje. Basicamente, um SGCW é aquele que armazena conteúdos digitais para a

busca, navegação, acesso e recuperação de usuários em um grupo de trabalho ou empresa (HUSAIN, 2012).

Segundo Saha e Setua (2014), os tipos mais comuns de sistemas de gerenciamento de conteúdo são:

- **Enterprise Content Management (ECM):** Sistemas que facilitam a gestão de documentos societários e outros tipos de informações, tanto para uso interno entre os setores e funcionários quanto para uso externo com os parceiros de negócios da empresa, os clientes, os reguladores e o público em geral.
- **Digital Asset Management (DAM):** Sistemas que gerenciam ativos de mídia avançada, muitas vezes, incluindo áudio digital e vídeo clips, para recuperação e reutilização em ambientes de produção de mídia. Estes sistemas são, por vezes, também chamados de *Media Asset Management (MAM)*.
- **Web Content Management (WCM):** Sistemas para fornecer projetos contendo o modelo de página, fluxo de trabalho editorial e ambientes de publicação especificamente para sites e outras formas de distribuição de conteúdo da Internet.
- **Document Management System (DMS):** Sistemas de arquivamento eletrônico para captura, armazenamento e rastreamento de imagens de documentos. O acesso ao conteúdo de um *DMS* deve ser rápida e segura. Retenção e rastreabilidade também ser partes de *DMS*

2.2 Agentes Conversacionais

Um agente de conversacional é um software capaz de interpretar e responder a declarações feitas por usuários em linguagem natural. Tem o potencial de integrar técnicas linguísticas computacionais com internet. O termo Agente Conversacional é usado principalmente em um contexto acadêmico por cientistas e engenheiros, é uma área de pesquisa que trabalha para garantir uma interação satisfatória e relevante com o usuário. Os cientistas também se concentram no desenvolvimento de Agentes Conversacionais Encorpados (*ECA – Embodied*

Conversational Agent no inglês), ou seja, não apenas uma interface de chat mas um personagem com expressões faciais ou corporais capaz de melhorar a interação humano-computador de forma verbal e não verbal (LUN, 2011), a criação de um personagem com tais características se distânciam do objetivo desta pesquisa.

2.2.1 Comparação de *Chatbot* com Sistemas de Diálogo

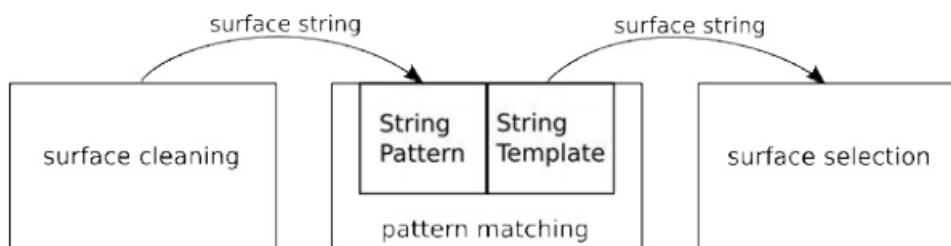
A terminologia *chatbot* origina-se do sistema correspondente Chatterbot, inventado como um personagem do jogo multiusuário (MAULDIN, 1994). Sua principal tarefa era responder perguntas do usuário sobre a navegação através da masmorra, outros objetos e jogadores disponíveis no mundo deste jogo.

O jogo obteve algum sucesso simulando a conversa com os jogadores, que por não saberem da possibilidade de haver um robô no jogo, pensaram que se tratava de outro usuário. Isso graças ao uso de regras simples e alguns truques inovadores inspirados em seu precursor ELIZA, um programa desenvolvido pelo cientista Joseph Weizenbaum (WEIZENBAUM, 1966).

ELIZA era um pequeno sistema que ganhou popularidade pela simulação de um psicoterapeuta virtual, o qual usuários se reportavam através de conversas digitadas em linguagem natural. Embora Weizenbaum não tenha chamado seu software de “chatbot”, é aceito que por suas características seja classificado como tal (PEREZ-MARIN; PASCUAL-NIETO, 2011, p. 2).

O software ELIZA apresenta as principais características dos chatbots tradicionais: a interface baseada em texto e um algoritmo de reconhecimento de padrão para prover respostas a estímulos que constitui a base para as funcionalidades do diálogo (PEREZ-MARIN; PASCUAL-NIETO, 2011). A Figura 1 mostra a arquitetura do protótipo de um sistema chatbot, contendo um componente pequeno para análise de entrada que faz alguma limpeza na cadeia de caracteres provenientes da interface com o utilizador, o núcleo de diálogo, que utiliza a correspondência de padrões para selecionar o próximo enunciado à dizer, e um componente saída o qual seleciona a resposta.

Figura 1 – Arquitetura de núcleo de um chatbot tradicional



Fonte: (PEREZ-MARIN; PASCUAL-NIETO, 2011)

Já um sistema de diálogo é um termo que deriva principalmente do campo de Processamento de Linguagem Natural e Inteligência Artificial que investiga e denota um sistema capaz de conduzir uma conversa com outro agente, normalmente um humano.

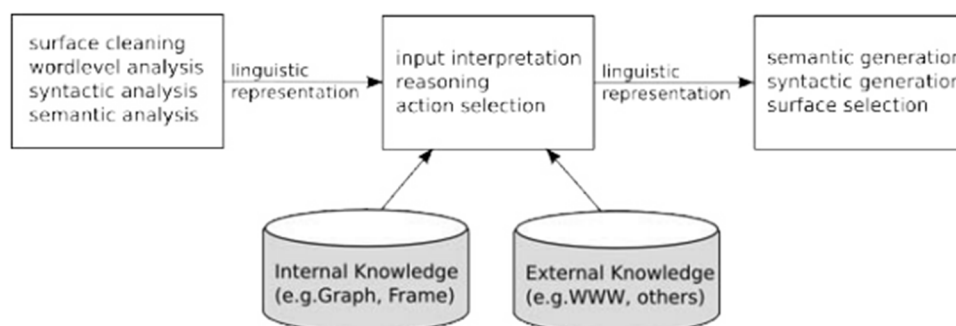
McTear (2004) observa as seguintes diferenças entre os sistemas de diálogo e os sistemas para “simulação de conversação” (*chatbots*): a) sistemas de diálogo são mais bem embasados teoricamente e b) sistemas de diálogo muitas vezes são desenvolvidos para um domínio específico, enquanto os sistemas de simulação de conversação são destinados a conversas em um domínio aberto.

O fato do sistema de diálogo ser aplicado a um domínio específico está amparado pelo primeiro ponto: a utilização de métodos mais sofisticados e teoricamente mais bem motivados implicam em trabalho elaborado em uma pesquisa mais aprofundada, sendo que seu uso só parece viável se aplicado em um domínio de aplicação específica. Esta característica é que servirá como base para definição de um sistema de diálogo e pode ser compreendida facilmente observando a sua arquitetura. Sistemas de diálogos podem ser categorizados por meio dos seus componentes, sendo que eles são equipados com pelo menos as seguintes funcionalidades: um processador de entrada e compressão de linguagem natural (NLU); um componente que orienta o fluxo do diálogo, comumente chamado “gerente de diálogo”; um componente de geração da saída.

A Figura 2 apresenta o exemplo de uma arquitetura protótipo do sistema de diálogo simplificada, mostrando as semelhanças e diferenças na arquitetura chatbot. A análise de entrada não se limita a limpeza de superfície, mas é executado através de uma cadeia de processamento linguística todo desde a superfície até uma representação linguística. Da mesma forma, a geração de saída usa uma

cadeia de linguagem de geração para derivar uma sequência de superfície a partir da descrição linguística da resposta. Assim, é capaz de fornecer várias cordas diferentes de superfície originários de uma única descrição semântica para o utilizador.

Figura 2 – Arquitetura básica de um sistema de diálogo



Fonte: (PEREZ-MARIN; PASCUAL-NIETO, 2011)

Apesar de possuírem arquiteturas muito similares, o que muitas vezes faz com que chatbots sejam referidos como “sistemas de diálogo” por muitas bibliografias existentes, existem diferenças cruciais em um sistema de diálogo, as quais dão relevância a essa diferenciação. Sistemas de diálogo incluem recursos de inteligência artificial para o módulo gerente de diálogo, os quais podem ser usados para recuperar a próxima ação possível ou resposta via raciocínio real sobre os fatos conhecidos do domínio ou o contexto do diálogo (PEREZ-MARIN; PASCUAL-NIETO, 2011, p. 4).

2.2.2 AIML - Artificial Intelligence Markup Language

Semelhante a ELIZA, ALICE é um robô de estímulo-resposta escrito em AIML, que funciona com um motor de correspondência de padrões, que mapeia padrões com ou sem expressões regulares, para uma entrada textual e fornece uma resposta modelo correspondente previamente salva em um banco de dados. Esses modelos da mesma forma podem usar pedaços combinados de entrada como parte de sua saída. Ao contrário de ELIZA, a instância mais comum do ALICE Bot é um produto de código aberto com cerca de 500 desenvolvedores e autores de conteúdo e um banco de dados igualmente espaçoso de mais de 40.000 pares padrão-modelo.

Devido à sua falta de inteligência ou conhecimento explicitamente codificado, esse tipo de *chatbot* é tão inteligente quanto à extensão do seu banco de dados. Esta é a razão pela qual é possível existir uma conversa engraçada com ALICE no início de uma interação, mas esta mesma conversa pode ficar confusa depois de um curto período de tempo (PEREZ-MARIN; PASCUAL-NIETO, 2011, p. 6).

AIML, *Artificial Intelligence Markup Language*, é uma linguagem derivada de XML bem conhecida para construir bases de conhecimento de Agentes Conversacionais, em um raciocínio de contexto baseado em casos e algoritmo de correspondência de padrão textual (DE GASPERIS, 2010). Os elementos mais importantes do AIML são apresentados na Tabela 1.

Tabela 1 - Elementos básicos da AIML

Tag	Descrição
<aiml>	Define início e fim do documento AIML.
<category>	Define uma unidade de conhecimento que representa uma possível interação com o usuário.
<pattern>	Contém um padrão que deve ser confrontado com a entrada do usuário. Cada entrada do usuário será avaliada conforme os nodos <pattern>.
<template>	Contém a resposta para a entrada do usuário.

Fonte: (ALICE AI FOUNDATION, 1999)

O funcionamento do AIML consiste em localizar o padrão apropriado de maneira simplificada. Na Figura 3 é possível observar a estrutura básica de um arquivo AIML. Nessa imagem o padrão é “OI, EU SOU *”, ou seja, assim que o usuário digitar “OI, EU SOU FÁBIO”, qualquer palavra pode substituir o (*) asterisco que é um caractere utilizado com esta função de substituição na notação AIML. Desta forma é possível que os termos capturados nesses padrões possam ser utilizados na resposta.

Figura 3 – Exemplo de código em Linguagem AIML

```

1 |<?xml version="1.0" encoding="UTF-8"?>
2 | <aiml version="2.0">
3 |     <category>
4 |         <pattern>OI EU SOU *</pattern>
5 |         <template>Oi <star index="1" /></template>
6 |     </category>
7 | </aiml>

```

Fonte: (PRATES, 2014, p. 44)

2.2.3 PyAIML um Interpretador Python para AIML

Um software chamado Interpretador AIML é usado para ler arquivos escritos com a notação AIML e fornecer funcionalidades em nível de aplicação com base em sua infraestrutura. Um interpretador AIML pode recorrer aos serviços de um processador XML. Supõe-se que um interpretador AIML é um módulo de uma aplicação maior, genericamente denominado de *bot*, que carrega o conjunto funcional maior de interação baseada em AIML (BUSH, 2011). O PyAIML¹ é um exemplo de interpretador AIML, implementado puramente com pacotes na linguagem Python.

Figura 4 – Exemplo básico de código com PyAIML

```

1 | import aiml
2 |
3 | k = aiml.Kernel()
4 | k.learn("std-startup.xml")
5 | k.respond("load aiml b")
6 |
7 | while True: print k.respond (raw_input(">"))

```

Fonte: (STRATTON, 2003)

Na Figura 4 é possível observar que para trabalhar com PyAIML, basta simplesmente realizar a importação dos pacotes, cujo código está presente na linha 1, em seguida instanciar um objeto *Kernel*, conforme o código da linha 3. Então o próximo passo é carregar alguns arquivos AIML. Isto é feito através do método *learn*, cujo exemplo está presente na linha 4. Neste exemplo foi carregado o arquivo de inicialização do conjunto AIML padrão que é disponibilizado junto com o instalador

¹ PyAMIL – Interpretador AIML: <http://pyaiml.sourceforge.net/>

do PyAIML. Ele define um padrão AIML único, indicado como “load aiml b”, que por sua vez faz com que o resto do conjunto seja carregado. Para desencadear o processo de interpretação, deve ser passado o valor “load aiml b” como parâmetro para o método *respond*, tal como exemplificado na linha 5. O método *respond* retorna uma *string* contendo a resposta de *Kernel* para a entrada, que pode ser processada caso necessário. Por fim na linha 7 é possível inicial o processo de interação com o usuário, tendo como base os dados descritos nos arquivos importados (STRATTON, 2003).

2.3 Processamento de Linguagem Natural

Processamento de Linguagem Natural (PLN) utiliza modelos matemáticos e linguística computacional para desenvolver sistemas capazes de, entre outras atividades, compreender instruções em linguagem natural, ou seja, instruções feitas por um ser humano.

Tecnologias de PLN vêm apresentando um impacto dramático a respeito da forma como as pessoas interagem com os computadores, na forma que as pessoas interagem umas com as outras através do uso da linguagem e na disponibilização de uma grande quantidade de dados linguísticos agora em formato eletrônico. Do ponto de vista científico, o PLN envolve questões fundamentais sobre como estruturar modelos formais (por exemplo, modelos estatísticos) dos fenômenos de linguagem natural, e de como projetar algoritmos que implementam esses modelos (COLLINS, MICHAEL, 2013).

PLN é usado por diversos sistemas, motores de busca como Google ou Weotta¹, interfaces de voz, como Siri², Google Now³ ou Dragon NaturallySpeaking⁴. Devido ao seu crescente uso, alguns recursos específicos foram desenvolvidos, em formas de pacotes auxiliares. Alguns deles, de interesse para este trabalho, são descritos a seguir.

¹ <http://www.weotta.com/>

² <http://www.apple.com/ios/siri/>

³ <https://www.google.com/landing/now/>

⁴ <http://www.nuance.com/dragon/index.htm>

O *NLTK – Natural Language Toolkit* - é um pacote para criação de sistemas de PLN inicialmente criado no meio acadêmico, porém com ampla utilização na indústria de software devido sua ampla gama de funcionalidades (PERKINS, 2014). *NLTK* foi criado originalmente em 2001 como parte de um curso de linguística computacional no Departamento de Computação e Ciência da Informação da Universidade da Pensilvânia. Desde então, tem sido desenvolvido e expandido com a ajuda de dezenas de contribuidores (BIRD; KLEIN; LOPER, 2009). É um conjunto de programas, bibliotecas e utilitários, escritos em Linguagem Python, para o processamento de linguagem natural. Disponibiliza as seguintes funcionalidades essenciais para PLN (LOPER; BIRD, 2002):

- *sentence delimiters*: procuram definir o escopo de cada sentença através de padrões utilizados em cada idioma, como pontuação de final de frase, letras maiúsculas e demais caracteres possíveis;
- *tokenizes*: são responsáveis por dividir o texto em partes, onde cada fatia representa uma palavra. Basicamente, eles são responsáveis por localizar uma sequência de caracteres, separadas por um delimitador, geralmente um espaço ou um ponto;
- *stemmers*: fazem a análise morfológica das palavras, associando-as com suas variantes, de acordo com seu radical;
- *taggers*: baseiam-se nos *tokenizes* e *sentence delimiters* a fim de achar a função de cada palavra na frase. Nesse ponto cada um dos *tokens* da sentença são classificados como substantivos, adjetivos, verbos e etc;
- *stop words*: são listas de palavras consideradas irrelevantes em cada idioma. Essas palavras geralmente aparecem repetidas vezes e prejudicam o processamento da sentença;
- *corpora*: é uma grande coleção de documentos (*corpus*) previamente processados, utilizados como base para análise estatística das sentenças.

Outro exemplo de recurso auxiliar relevante, a AlchemyAPI¹ usa tecnologia de processamento de linguagem natural e algoritmos de aprendizado de máquina para extrair metadados semânticos de conteúdo, tais como informações sobre pessoas, lugares, companhias, fatos, relações, autores e idiomas. Uma API *endpoint* é provida para realização de análise de conteúdo em páginas web, acessíveis pela internet e publicadas no formato HTML ou texto puro (ALCHEMYAPI, 2015).

Aplicativos codificados em Python, PHP, Ruby ou Node.js podem fazer uso da AlchemyAPI. Suas funções são acessadas por meio de uma API REST, e os dados são retornados como uma estrutura fácil de ser analisada por qualquer aplicação. Para projetos que necessitam extrair informações de textos não estruturados, as seguintes funções para análise de texto estão disponíveis (ALCHEMYAPI, 2015):

- Extração de entidade: identificar os substantivos próprios, ou seja, pessoas, empresas, locais, etc;
- Análise de sentimento: determinar o sentimento geral ou para uma palavra-chave ou entidade específica;
- Extração de palavra-chave: extrair os termos importantes;
- Extração de marcação: identificação da relação sujeito-predicado-objeto;
- Classificação baseada em Taxonomia: categoriza automaticamente o texto ou conteúdo baseado na web em uma taxonomia hierárquica;
- Extração do Autor: identifica o autor de um determinado texto;
- Detecção do Idioma: detecta 97 idiomas;
- Extração de resumos de texto: identifica apenas o conteúdo importante a partir de uma página web;
- Detecção de *Feed*: extração de ATOM ou RSS *feeds* de uma página web.

¹ <http://www.alchemyapi.com/>

2.4 Web Semântica

A Web Semântica tem como objetivo a resolução de deficiências observadas na web como a conhecemos e o desenvolvimento de uma série de melhorias em relação às possibilidades da Internet atual (RIGO, 2008, p. 29). A Web Semântica é construída sobre padrões W3C, como o modelo dados RDF, a linguagem de consulta SPARQL, RDF *Schema* e OWL para armazenar vocabulários e ontologias. Um produto ou projeto pode lidar com a semântica, mas se ele não emprega padrões compartilhados, ele não pode utilizar o potencial de compartilhamento e integração da web semântica (DUCHARME, 2013, p. 20).

2.4.1 RDF (*Resource Description Framework*)

RDF é um modelo padrão para o intercâmbio de dados na Web. RDF tem características que facilitam a fusão de dados, mesmo se os esquemas subjacentes diferem. O RDF especificamente suporta a evolução dos esquemas ao longo do tempo sem a necessidade de modificar a estrutura dos documentos nos sistemas consumidores de dados (RDF WORKING GROUP, 2014).

A terminologia básica do RDF está resumida na tabela 2, a seguir.

Tabela 2 - Terminologia Básica RDF

Sujeito	Utilizado para denotar um recurso no mundo, tem de ser identificado por uma Identificador Uniforme de Recurso (URI) - <i>Uniform Resource Identifier</i> (em inglês), é também chamado de nó, ou nó inicial num grafo RDF
Predicado	Utilizada para designar a relação entre sujeito e objeto, também chamados de aresta em um grafo de RDF
Objeto	Utilizado para denotar recurso no mundo, tem de ser identificado por uma URI ou literal, é também chamado de nó, ou nó final num grafo RDF

Fonte: (YU, 2014, p. 41)

A representação da estrutura de uma declaração RDF pode ser vista na Figura 5.

Figura 5 – Grafo com estrutura de uma declaração RDF



Fonte: (YU, 2014, p. 30)

2.4.2 *RDF Serialization*: RDF/XML Sintaxe

RDF/XML é uma sintaxe XML definida por uma especificação W3C para representar um grafo RDF. Nota-se que essa não é a única sintaxe de serialização que pode ser utilizada. Existem outras como: Notation-3, Turtle e N-Triples como formato de serialização não XML, também introduzidos pelo W3C e amplamente utilizados por desenvolvedores da Web Semântica.

2.4.3 Microformatos, RDFa, GRDDL e RDFa API

Um uso de um microformato é uma forma de incorporar dados semânticos específicos em documentos HTML. Assim, quando um aplicativo acessa seu conteúdo, será capaz de identificar mais precisamente qual o conteúdo em questão. Existe uma familiarização do uso, em páginas HTML, de dados que representam pessoas, empresas e lugares. Para cada um destes tópicos, vêm sendo desenvolvidos padrões de microformatos. Por exemplo, para adicionar dados semântico sobre pessoas em documentos é possível utilizar um microformato chamado hCard, que oferece um conjunto de elementos que podem ser usados para especificar este tipo de conteúdo (YU, 2014, p. 98).

Outra maneira muito popular para armazenar triplas é o padrão W3C RDFa. Ele permite que sujeitos, predicados e objetos sejam armazenados em um documento XML que não foi projetado para acomodar RDF, bem como os documentos HTML. RDFa define alguns novos atributos e especifica os vários atributos preexistentes no HTML como atributos locais para armazenar e ou identificar sujeitos, predicados e objetos misturados com dados XHTML. Desta forma, consiste em uma forma interessante e otimizada para descrever meta-dados sobre o conteúdo de documentos HTML. Cada vez mais utilitários estão disponíveis

para identificar e utilizar as triplas de atributos RDFa em um formato que permite que se possa consulta-los com outras linguagens, tal como a linguagem de consulta SPARQL (DUCHARME, 2013, p. 29).

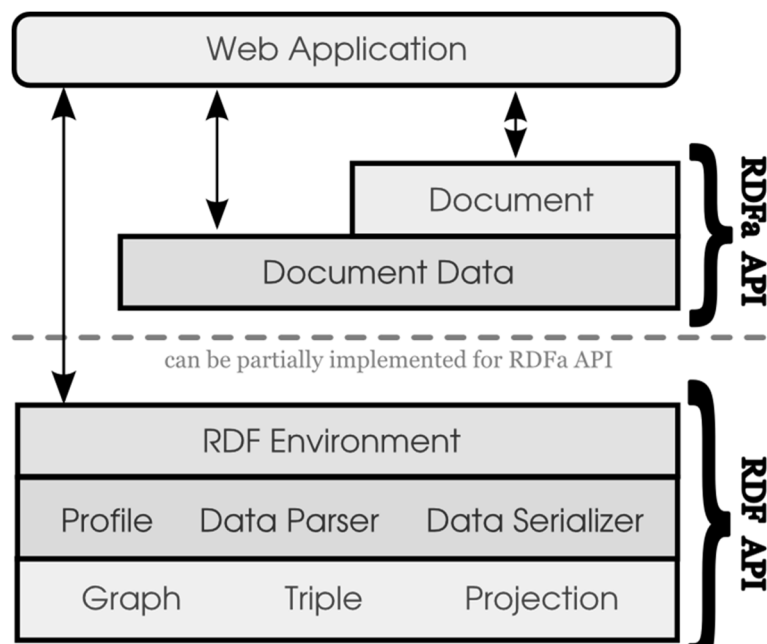
O formato Recolhendo Descrições de recursos de Dialectos de Idiomas (GRDDL) - *Gleaning Resource Descriptions from Dialects of Languages* (em inglês) é um formato de marcação que permite a seus usuários obter triplas RDF de documentos XML (chamados dialetos XML), como por exemplo documentos XHTML. Para compreender o GRDDL devem ser destacados os seguintes conceitos (YU, 2014, p. 116):

- **Agente GRDDL-aware:** um agente de software que é capaz de reconhecer as transformações GRDDL e executar essas transformações para extrair RDF;
- **Transformação GRDDL:** um algoritmo para obter RDF a partir de um documento origem.

É possível utilizar GRDDL para extrair RDF tanto com Microformatos quanto com RDFa, naturalmente.

Já RDFa API é um padrão W3C ainda não finalizado que possui como intuito detalhar um conjunto de interfaces para simplificar a extração e utilização de informações estruturadas de páginas Web. A API possui implementação para as linguagens Javascript, Python, Ruby e Perl. A Figura 6 descreve a relação entre todos os conceitos relacionados a RDFa API (HERMAN; BIRBECK; RIXHAM, 2012).

Figura 6 – Pilha de conceitos sobre RDFa API e RDF API



Fonte: (HERMAN; BIRBECK; RIXHAM, 2012)

A *API RDFa* é formada por camadas em cima de conceitos definidos pela *API RDF*. Os conceitos de nível mais baixo são triplas, grafos e projeções. Estas são estruturas de dados utilizadas por desenvolvedores da web para trabalhar com dados expressos em Linguagem *RDF*. Trabalhar diretamente com as estruturas de dados pode ser complicado às vezes, como a exigência de utilização de URIs absolutas quando se acessa as propriedades. Sendo assim, *Profiles* são fornecidas para suportar o acesso aos dados usando URIs compactos (*curies*). Analisadores de dados e *Serializes* de dados fornecem mecanismos para ler e gravar dados *RDF* em uma variedade de línguas *RDF*. Todos esses conceitos são fornecidos através de uma interface de alto nível chamada *RDF Environment*. Implementações da *API* devem observar que apenas as partes da *API RDF* que são usadas diretamente por esta *API* precisam ser implementadas para se obter uma implementação conforme a *API RDFa*. Ou seja, apenas projeções são obrigadas a ser implementadas a partir da *API RDF* para uma implementação plenamente em conformidade com a *API RDFa*. O nível mais alto da *API RDFa* fornece duas interfaces de acesso aos dados no documento. Consultas simples para os dados de *DOM* podem ser realizadas utilizando a interface do documento, enquanto consultas mais complexas para dados estruturados podem ser realizadas utilizando a interface de dados de documentos (HERMAN; BIRBECK; RIXHAM, 2012).

2.4.4 RDFS

RDFS ou *RDF Schema* é um padrão W3C para representação de conhecimento extensível que pode ser usado para criar um vocabulário para descrever classes, subclasses e propriedades de recursos RDF. Com esta definição, RDFS pode ser entendido como uma linguagem de descrição de vocabulário RDF. Como padrão, RDFS pode fornecer construções de linguagens que podem ser usadas para descrever as classes e propriedades dentro de um domínio de aplicação específico (YU, 2014, p. 123).

2.4.5 OWL

OWL é atualmente a linguagem mais popular para usar ao criar ontologias. O objetivo da OWL é exatamente o mesmo de *RDF Schema*: para definir ontologias que incluem classes, propriedades e suas relações para o domínio específico da aplicação. No entanto, em comparação com *RDF Schema*, OWL nos proporciona a capacidade de expressar relações muito mais complexas e mais ricas. Portanto é possível se desenvolver aplicativos com uma maior capacidade de raciocínio (YU, 2014). Uma vez que OWL é construída sobre *RDF Schema*, todos os termos contidos no vocabulário RDFS ainda podem ser usados durante a criação de documentos OWL.

2.4.6 SPARQL

SPARQL é uma linguagem de consulta capaz de consultar o conteúdo de dados em RDF e também de fornecer um protocolo que deve ser seguido para consultar um conjunto de dados RDF remoto. Os benefícios de uma linguagem de consulta como RDF são destacados a seguir (YU, 2014, p. 267):

- Consultar gráficos RDF para obter informações específicas;
- Consultar um servidor remoto e RDF para obter um conjunto de resultados;

- Executar consultas regulares de forma automatizada para novos conjuntos de dados RDF para gerar relatórios;
- Permitir o desenvolvimento de aplicativos em um nível superior, ou seja, as aplicações pode trabalhar com os resultados da consulta SPARQL, não diretamente com declarações RDF.

Um SPARQL *endpoint* pode ser entendido como uma interface com a qual os usuários podem acessar a consulta a um conjunto de dados RDF usando a linguagem de consulta SPARQL. Um SPARQL *endpoint* pode ser configurado para retornar resultados em diferentes formatos. Por exemplo, se for acessado por usuários humanos através de uma aplicação web, pode apresentar seus resultados na forma de uma tabela HTML(HyperText Markup Language), que é frequentemente construída por XSL (Extensible Stylesheet Language). Quando acessado por aplicativos, os resultados são serializados em formatos processáveis por máquina, como RDF/XML ou Turtle. Os SPARQL *endpoints* podem ser caracterizados como *endpoints* genéricos ou específicos. Um *endpoint* genérico funciona contra qualquer conjunto de dados RDF, que poderia ser armazenado localmente ou ser acessível a partir da Web. Um *endpoint* específico está ligado a um conjunto de dados em particular e este conjunto de dados não pode ser transferido para outro *endpoint* (YU, 2014, p. 268).

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados com o tema proposto dessa dissertação. Foram usadas duas abordagens para seleção dos trabalhos relacionados. Em primeiro lugar buscou-se identificar trabalhos de pesquisa, publicados em revistas científicas qualificadas e em eventos da área de inteligência artificial, IHC e processamento de linguagem natural, que abordassem o tema de Agentes Conversacionais com Base em Web Semântica, a fim de identificar aspectos gerais de arquitetura e detalhes de implementação de seus componentes. Em um segundo momento, o aspecto estudado está relacionado as características de integração das consultas SPARQL a linguagem AIML, a anotação dos metadados com apoio de RDFa e a extração dos dados anotados para compor uma base de dados *Triple Storage*.

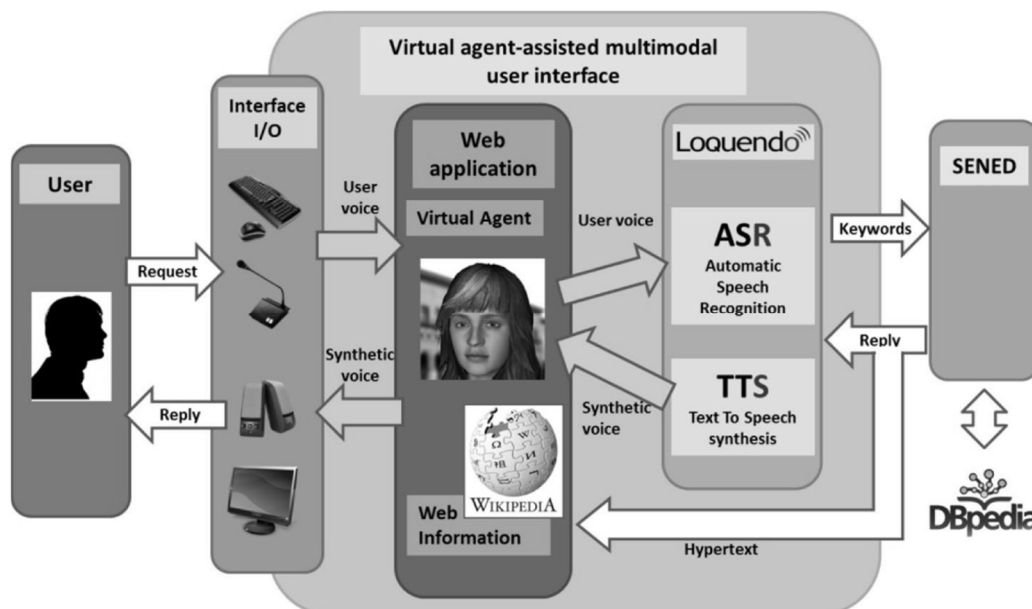
Nos tópicos a seguir são apresentados os trabalhos que destacam diferentes formas de aproveitar o conteúdo da web para compor o repertório de diálogos de um Agente Conversacional, quais as abordagens utilizadas, suas arquiteturas, principais diferenciais e, por fim, é apresentada uma tabela comparativa contendo uma visão geral sobre os mesmos.

3.1 VOX System: a semantic embodied conversational agent exploiting linked data

Segundo Seron e Bobed (2014), VOX é uma plataforma de ECAs que explora as fontes de informação estruturadas na *World Wide Web* sob a forma de Dados Ligados (*Linked Data*). O VOX baseia-se em um repositório externo de dados ligados (fora do controle da equipe do VOX) e toma como entrada uma ontologia que tem dois papéis no sistema: 1) definir a taxonomia do domínio de pesquisa; e 2) definir a estrutura dos objetos no domínio de pesquisa, refinando e sugerindo novos resultados da pesquisa. Com essa abordagem, é possível fornecer diferentes visões sobre um repositório externo de dados gerais apenas por adaptar a ontologia fornecida. Além disso, a abordagem pode ser associada a qualquer terminal SPARQL público sem sobrecarregá-lo, o que é importante em cenários abertos, tais

como aquele representado por *Linked Data*. A Figura 7 ilustra a arquitetura geral deste sistema.

Figura 7 - VOX: *Multimodal Interface Architecture*



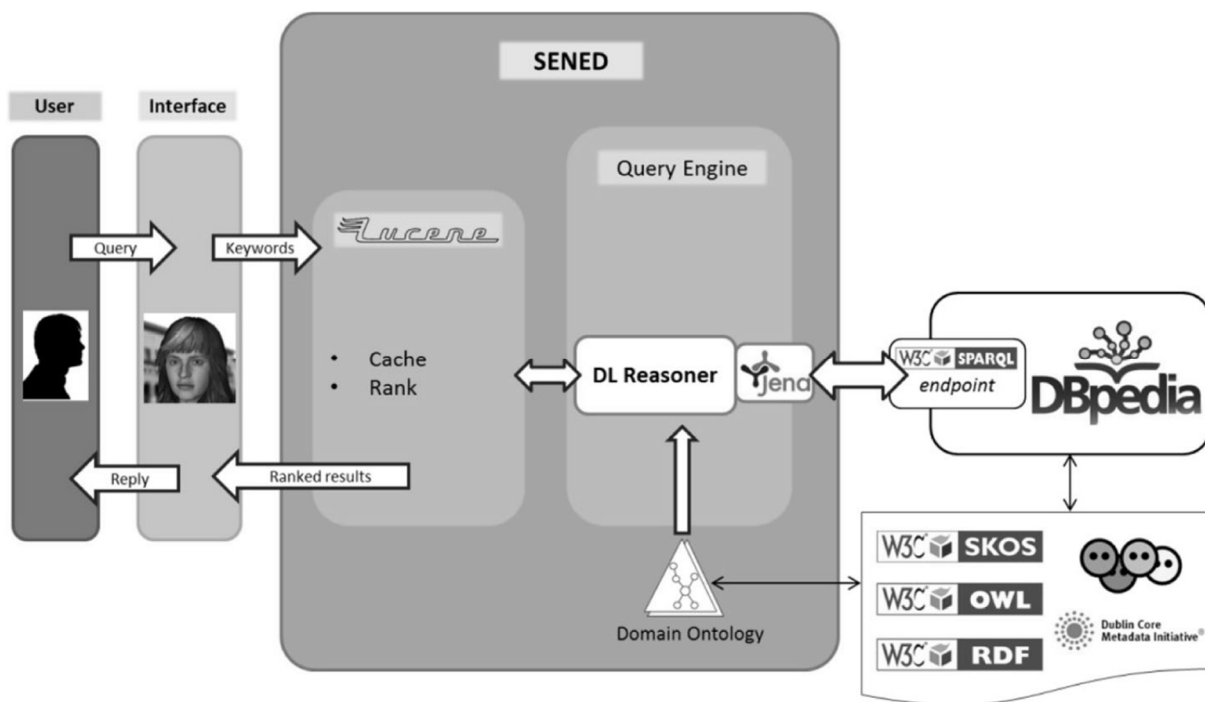
Fonte: (SERÓN; BOBED, 2014).

O modelo proposto suporta interação com o usuário através de diferentes canais: texto, mouse e voz. No entanto, os aspectos mais importantes são a sua representação visual e a interação por voz. A representação visual do *ECA* é definida por seu modelo tridimensional e texturas, juntamente com animações básicas e expressões. Para a interação por voz o sistema utiliza o Loquendo ASR¹, amplamente usado para o reconhecimento de voz. Ele usa o *W3C Speech Recognition Grammar Specification*.

O módulo SENE permite o acesso em tempo real as informações disponíveis na DBpedia, conforme descrito no resumo de funcionalidade do seu modelo, conforme a Figura 8.

¹ <http://www.loquendo.com/en/products/speech-recognition/>

Figura 8 – Vox: módulo de acesso a DBPedia



Fonte: (SERÓN; BOBED, 2014).

O módulo SENE é composto pelos seguintes componentes de apoio (SERÓN; BOBED, 2014):

- **O repositório Lucene:** possui dois benefícios, por um lado, ele fornece uma classificação dos resultados obtidos aplicando várias técnicas bem conhecidas no campo de *Information Retrieval*; por outro lado, atua como um cache (memória intermediária), aliviando a carga de trabalho imposta aos repositórios de dados externos;
- **O mecanismo de consulta:** Ele está encarregado de criar as consultas e armazená-las no repositório de dados. É implementado usando a OWL API¹, que permite ligar facilmente diferentes *DL Reasoned* e Jena² para acessar os repositórios de dados.

¹ <http://owlapi.sourceforge.net/>

² <http://jena.apache.org/>

- Para a ontologia do domínio, que foi implementada em OWL, foi utilizado o Protege ¹, um editor de ontologias e framework para base de conhecimento;
- A DBPedia é armazenada em um repositório Virtuoso, que é na realidade um repositório RDF que dá acesso aos dados armazenados por meio de um SPARQL *endpoint*. SPARQL é a linguagem usada para consultar gráficos RDF, através de métodos fornecidos pelo Jena para, uma vez que as consultas estejam definidas adequadamente em SPARQL, sejam passadas ao *endpoint* para acessar os dados reais.

3.2 Embodied Conversational Agent Based on Semantic Web

Um ECA é parecido com um personagem de desenho animado capaz de interagir com usuários através de conversa e gestos em uma tela de computador. ECA faz as interações humanas com computador mais amigáveis porque incorpora a maioria das habilidades de comunicação que já são empregadas cotidianamente, como conversa em linguagem natural. Esses Agentes são úteis na navegação web, ao serem incorporados em páginas web (KIMURA; KITAMURA, 2006).

Para construir esses agentes, é necessário descrever cenários para explicar a navegação entre páginas web. Convencionalmente tais cenários são escritos manualmente por desenvolvedores que usam uma linguagem de descrição de diálogo como o AIML, por isso é difícil de atualizá-los quando as páginas web sofrem alterações. Kimura e Kitamura (2006) propõem um modelo capaz de gerar declarações de diálogo automaticamente para navegação entre páginas web, baseando-se no conteúdo das próprias páginas web.

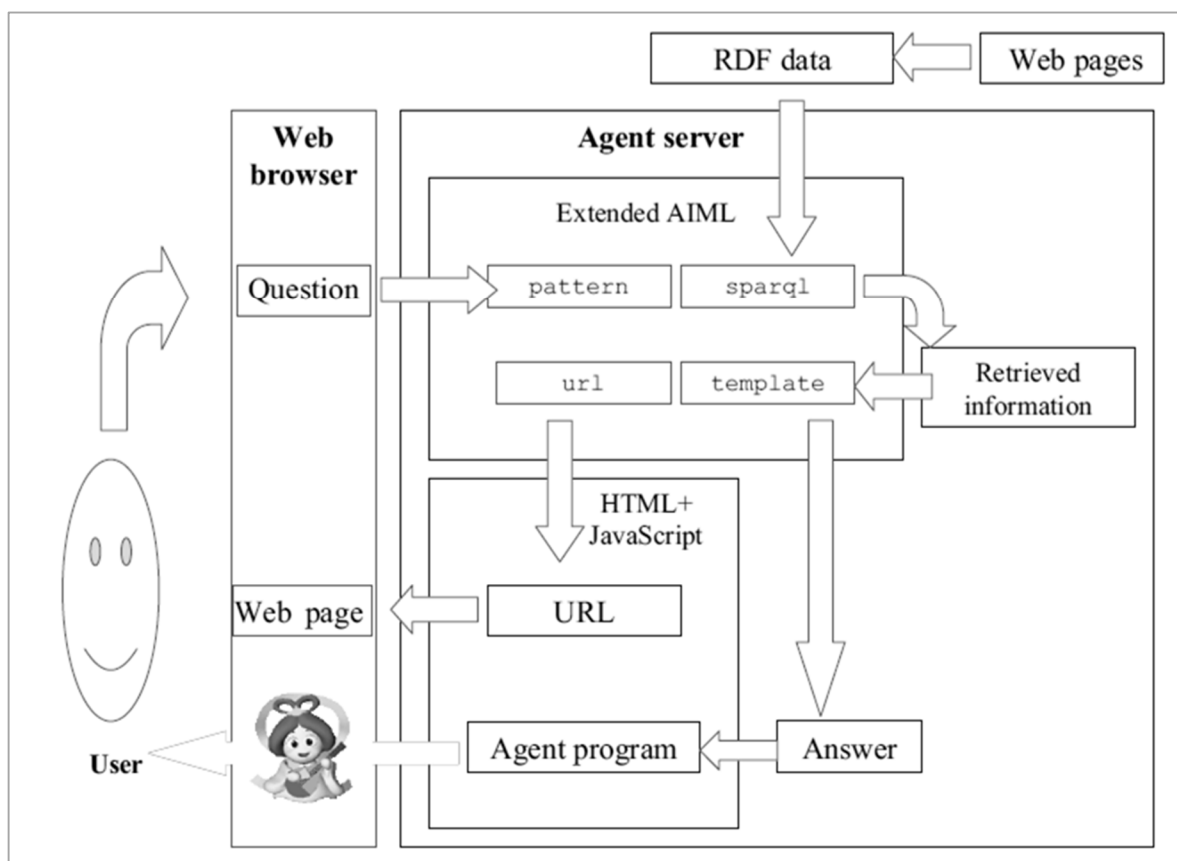
Para esta finalidade é necessário que o agente identifique o conteúdo das páginas, o que torna necessário a representação do conteúdo no formato RDF. Para a realização de consultas foi utilizada a linguagem SPARQL e também foi necessário estender a linguagem AIML para incorporar as consultas SPARQL.

¹ <http://protege.stanford.edu/>

Como protótipo, foi desenvolvido um sistema de navegação entre páginas web empregando um ECA (KIMURA; KITAMURA, 2006).

Este modelo, cujos elementos principais estão descritos na figura 9, é proposto com a finalidade de gerar declarações de diálogo automaticamente com base na tecnologias da Web Semântica. Para atingir esse objetivo, a agente precisa primeiramente entender o conteúdo das páginas web para em seguida conversar com usuários de acordo com esse conteúdo. RDF é utilizado para representar o conteúdo das páginas Web tornando-as reconhecíveis por máquina e pela linguagem de consulta SPARQL. Para o segundo ponto foi realizada a extensão da linguagem AIML, para conter uma consulta SPARQL (KIMURA; KITAMURA, 2006). A Figura 9 apresenta a arquitetura do sistema proposto em Kimura e Kitamura (2006) de um agente guia de laboratório interativo.

Figura 9 - Arquitetura do sistema de agente guia de laboratório interativo



Fonte: (KIMURA; KITAMURA, 2006)

O trabalho estudado apresenta a abordagem de Extensão AIML para incorporar as consultas SPARQL. AIML suporta apenas regras fixas de conversação entre um agente e um usuário, por isso foi necessário estendê-lo para incorporar

uma consulta SPARQL. Foi adicionada a marcação `<sparql>` e `<url>` na especificação AIML. Também foi necessário modificar as descrições nas tags `<pattern>` e `<template>` para poderem lidar com variáveis. Por exemplo, se o usuário envia a consulta “Qual o projeto atual do Kimura?”, a consulta corresponde ao padrão na declaração AIML e a variável `$x` é definida como “Kimura”. Em seguida o sistema executa uma consulta SPARQL definida na tag `<sparql>`. Como o elemento identificado pela variável `$x` é definido como “Kimura”, a consulta recupera o título do projeto atual de Kimura de um arquivo RDF chamado “member.rdf” e o resultado definido na variável `?ans`. Finalmente o sistema responde ao usuário dizendo: “O projeto atual do Kimura é Integração da Web usando Personagens Agentes”. A tag `<url>` é utilizada apenas para mostrar uma página web ao usuário. A figura 10 apresenta um trecho de código em formato AIML contendo o suporte necessário para o exemplo descrito anteriormente.

Figura 10 – Exemplo de Extensão AIML

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <aiml>
3 ....
4   <category>
5     <pattern>What is $x's current project?</pattern>
6     <template>$x's current project is ?ans.</template>
7     <sparql>
8       PREFIX rdf: &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#
9       &gt;;
10      PREFIX foaf: &lt;http://xmlns.com/foaf/0.1/&gt;;
11      PREFIX dc: &lt;http://purl.org/dc/elements/1.1/&gt;;
12      SELECT ?ans
13      FROM NAMED <member.rdf>
14      WHERE {
15        ?a foaf:family_name $x .
16        ?a foaf:currentProject ?y .
17        ?y dc:title ?ans .
18      }
19    </sparql>
20    <URL>http://ist.ksc.kwansei.ac.jp/~kitamura/lab/projects-j.htm</URL>
21  </category>
22 ....
23 </aiml>

```

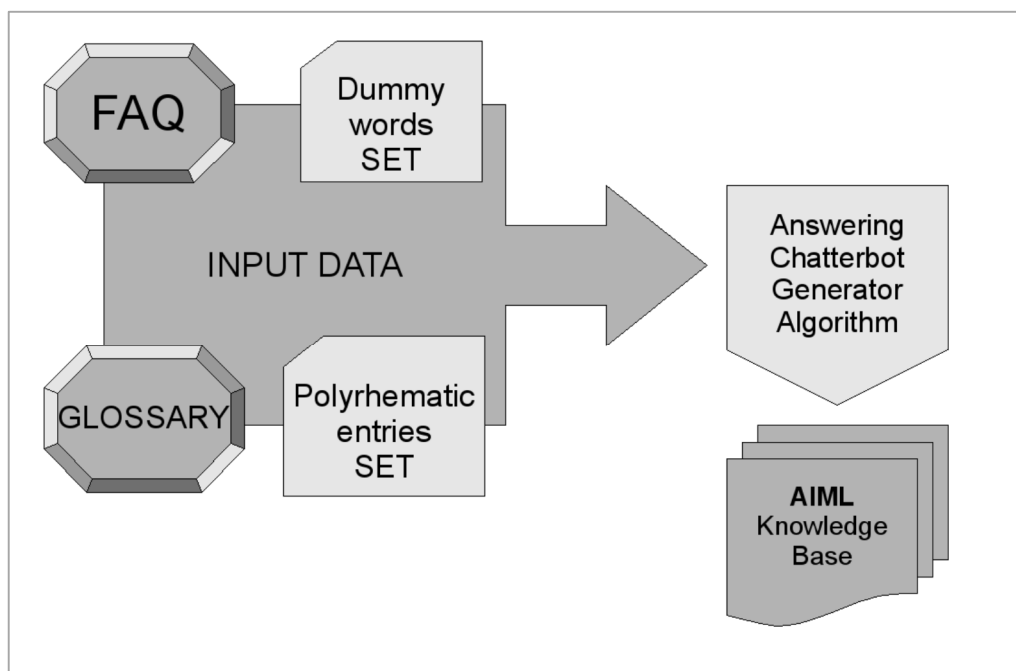
Fonte: (KIMURA; KITAMURA, 2006)

3.3 Building an AIML Chatter Bot Knowledge-Base Starting from a FAQ and a Glossary

De Gasperis (2010) descreve um *Chatter Bot* que utiliza uma metodologia inovadora baseada em um algoritmo para gerar automaticamente bases de conhecimento em formato AIML a partir de FAQs, texto puro e glossário de termos. Para a validação do projeto foi gerado um Agente Conversacional em língua italiana.

Todo o conhecimento relacionado com o domínio de interesse é incluído em um arquivo FAQ (*Frequently Asked Questions*) e em arquivos de Glossário. Assim, a disponibilização de um método para extrair todo o possível conhecimento de FAQ's e Glossários é o ponto de partida para o processo de gerar um Agente de software que responda a perguntas de um domínio de conhecimento, tal como mostrado na Figura 11.

Figura 11 - Fluxo de trabalho do processo de geração de diálogo de um Agente Conversacional



Fonte: (DE GASPERIS, 2010)

Dado um arquivo de texto contendo uma lista de FAQ, num formato onde uma questão está associada a sua respectiva resposta e um arquivo de Glossário onde o termo está associado a sua respectiva definição, a fim de obter a resposta adequada, o agente precisa ao término do processo de geração, as seguintes etapas devem ser cumpridas: a) Definição do conjunto de palavras falsas

dependendo do domínio do conhecimento; b) Definição do conjunto de entradas específicas do domínio; d) Definição do algoritmo de geração da conversa.

Portanto, o algoritmo trabalha com dois conjuntos de fontes de entrada, ou seja, com FAQs e com glossários. Os dois conjuntos são utilizados em combinação para filtrar as palavras não significativas antes de construir o padrão estabelecido para a geração das declarações AIML. Técnicas linguísticas computacionais podem ser utilizadas a fim de cortar o conjunto de palavras; elas são úteis para identificar palavras binárias, que geralmente correspondem às palavras mais frequentes que podem ser calculadas dado um conjunto de corpora de linguagem falada, a fim de não inseri-los na base de dados AIML.

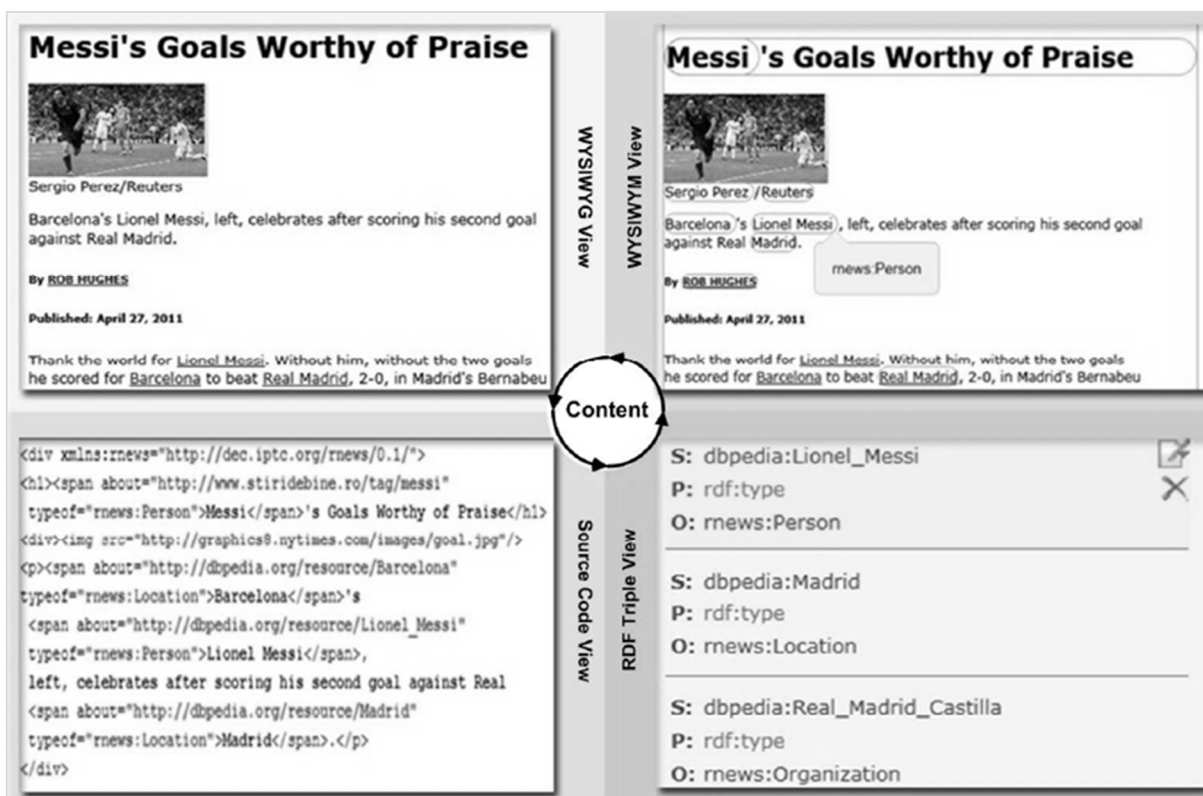
3.4 The RDFa Content Editor - From WYSIWYG to WYSIWYM

Recentemente observam-se progressos no gerenciamento do ciclo de vida do conteúdo semântico na web. No entanto, o aspecto menos desenvolvido é a facilitação da criação de conteúdo semântico de forma manual ou semiautomática. Khalili *et al* (2012) apresenta a abordagem RDFa *Content Editor* (RDFaCE) para combinar uma ferramenta WYSIWYG (acrônimo para “o que você vê é o que você quer recebe”, em inglês), para autoria de textos com a criação de anotações ricas para web semântica. É apresentada a visão WYSIWYM (acrônimo para “o que você vê é o que você quer dizer”, em inglês) foi realizada uma avaliação empírica com base nos resultados do uso da ferramenta (KHALILI; AUER; HLADKY, 2012).

A arquitetura do RDFaCE pode ser visualizada na Figura 12. Ela consiste em três camadas. A camada de base sobre a qual é disponibilizado o RDFaCE como *plugin* é o Editor de Texto Rico TinyMCE¹. É um editor HTML *open source* e foi escolhido por ser flexível, extensível e suportado por uma ampla gama de SGCW's, *blogs*, *wikis* e fóruns de discussão e etc (KHALILI; AUER; HLADKY, 2012).

¹ <http://tinymce.moxiecode.com>

Figura 13 – As quatro formas do autor visualizar o conteúdo



Fonte: (KHALILI; AUER; HLADKY, 2012).

A seguir são comentadas cada uma destas quatro visões:

- **WYSIWYG View:** “O que você vê é o que você tem” é a tradução literal para a sigla *WYSIWYG*, é uma visão clássica para interfaces de edição de texto-rico e muito utilizada por autores e jornalistas. É o tipo de ferramenta para autoria de conteúdo que está onipresente na Web e é parte integrante de SGCW's, Weblogs, Wikis e também de Lojas virtuais;
- **WYSIWYM View:** “O que você vê é o que você quer dizer” é a tradução literal para a sigla *WYSIWYM* e também é a proposta modelo RDFaCE. *WYSIWYM View* é uma extensão ao *WYSIWYG* tradicional que permite destacar nomes de entidades dentre outras informações semânticas. O destaque é realizado com seletores CSS3 especiais para anotações RDFa;
- **RDF Triple View:** Essa *View* resume todos os fatos que podem ser extraídos do texto anotado semanticamente com RDFa;

- *Source Code View*: Visão da interface onde todo o código fonte pode ser visualizado e editado.

3.5 Gerenciamento de Diálogo Baseado em Modelo Cognitivo para Sistemas de Interação Multimodal

A principal contribuição Prates (2014) está na utilização da memória de trabalho como forma de tornar os diálogos mais naturais, fazendo o agente demonstrar traços de consciência, quando consegue travar uma conversa com sensibilidade de contexto. Essas características são identificadas em duas situações propostas para avaliação do trabalho: a) identificar um contexto prévio referido, de modo que a percepção dos usuários seja adequada; b) identificar uma troca de contexto, evitando uma continuidade de diálogo de forma inadequada.

Figura 14 – Trecho de diálogo com sensibilidade a troca de contexto

```
Usuário: Olá!  
Sistema: Olá!  
Usuário: O que você gosta de comer?  
Sistema: Churrasco  
Usuário: E o que é isso?  
Sistema: Churrasco é o nome do prato feito a base de carne in natura ou processada, assada sobre o fogo ou brasas, com a utilização de estacas de madeira ou metal – chamados de espetos – ou de grelhas  
Usuário: O que vai no churrasco?  
Sistema: Comida típica brasileira: fraldinha, picanha, galinha.  
Usuário: E quem foi o diretor do filme Titanic?  
Sistema: James Cameron  
Usuário: E qual a sinopse do filme?  
Sistema: Titanic é um filme estadunidense de 1997 escrito e dirigido por James Cameron. É uma história ficcionalizada do naufrágio do RMS Titanic, estrelado por Leonardo DiCaprio e Kate Winslet.
```

Fonte: (PRATES, 2014).

A figura 14 ilustra um trecho de conversa gerado com o sistema, no qual podem ser observados os dois aspectos destacados. A situação descrita como identificação de um contexto prévio referido é considerada satisfeita quando o sistema permitir que um assunto previamente citado seja utilizado ao longo das interações futuras, sem necessidade de menções explícitas. A identificação de uma troca de contexto é considerada satisfatória quando o sistema responder adequadamente a uma mudança no assunto principal de conversação (PRATES, 2014).

A Figura 14 ilustra um diálogo gerado pelo sistema e demonstra as possibilidades do modelo desenvolvido para o tratamento dos itens de avaliação elencados. Deve ser destacado que as perguntas como “E o que é isso?” estão associadas corretamente ao contexto da pergunta anterior, bem como as perguntas sobre novos assuntos são corretamente tratadas. Durante a execução deste diálogo, os conteúdos foram recuperados na DBPedia após a análise sintática e o reconhecimento do tópico principal das mensagens enviadas pelo usuário.

3.6 Comparação entre os modelos estudados

A Tabela 3 apresenta um comparativo entre as características arquiteturais dos modelos estudados. Esta tabela foi construída para permitir uma visualização de aspectos que são temas importantes no desenvolvimento desta área e também para auxiliar na identificação de lacunas de interesse para a pesquisa atual.

A característica *Domínio* presente na primeira linha da Tabela 3 descreve o campo de aplicação dos trabalhos relacionados, o que proporciona possibilidades de análise quanto à requisitos e restrições segundo cada campo. O item *WYSIWYM* analisa o quanto os trabalhos estudados incorporam o relacionamento entre o significado dos conteúdos e sua visualização nos documentos (onde a autoria de conteúdo semântico seria apoiada por um SGCW por exemplo). No item *Representação de dados Estruturados* é verificado a utilização de padrões de representação estruturados e compartilhados, esse ponto é analisado identificando se existe uma ontologia específica para o domínio de conhecimento ao qual o modelo foi proposto. No item *Extração de metadados semânticos* verifica-se a apropriação das possibilidades de uso de marcações semânticas existentes no padrão HTML5 (o uso de microformatos, RDFa ou JSON-LD). Outra característica comparada foi o *Método de Integração*, ou seja, o método utilizado para integração do agente conversacional com as ferramentas de web semântica. Também foi verificado o *Mecanismo para tornar o diálogo mais natural*, identificado por recursos específicos para fomentar a melhoria da percepção de aceitabilidade dos usuários durante o diálogo, como é o caso da memória de trabalho proposta por Prates (2014) por exemplo.

Tabela 3 – Tabela comparativa dos Trabalhos Relacionados

Características / Trabalhos Relacionados	SERÓN; BOBED, 2014	KIMURA; KITAMURA, 2006	DE GASPERIS, 2010b	KHALILI et al, 2012	PRATES, 2014
Domínio	Interfaces Multimodais	Assistente de Navegação	Geração de Diálogo	Publicação de Conteúdo	Modelo Genérico
WYSIWYM	Não	Não	Não	Sim	Não
Representação de dados Estruturados	RDF/XML	RDF/XML	RDF/XML	RDFa e Micro-formatos	RDF/XML e JSON-LD
Extração de metadados semânticos	Não	Não	Sim	Sim	Sim
Método de Integração	Módulo DL Reasoner	Não possui	Extensão do AIML	Não possui	Não está explícito
Mecanismo para tornar o diálogo mais natural.	Não possui	Não possui	Não possui	Não possui	Memória de trabalho

Fonte: Elaborado pelo autor.

Os trabalhos analisados foram selecionados com foco no uso de aspectos de web semântica para composição do repertório de diálogos de agentes conversacionais. É possível identificar, com o exposto na tabela 3, que nenhum dos trabalhos estudados leva em consideração a integração das três categorias elencadas como prioritárias nesta pesquisa, ou seja, a integração do conteúdo gerado no SGCW, o uso da web semântica e a aplicação do agente conversacional. Como diz Khalili *et al* (2012), a maioria dos trabalhos considera a exportação de bases de dados relacionais, dados em arquivo (no formato texto/csv) ou outras fontes de informações para dados estruturados no formato RDF.

Conforme também pode ser evidenciado na análise da Tabela 3, nenhum destes trabalhos leva em consideração o consumo de dados semânticos publicados por autores de conteúdo em um SGCW de um Site web. Tais conteúdos já vem sendo publicados como pré-requisitos de outras área e contém alguma estruturação em termos de meta-dados e dados relacionados com o conhecimento do domínio. Sendo que assim apresentam uma oportunidade de pesquisa na sua utilização para composição de diálogos em linguagem natural.

Voltamos a salientar que não foi identificado nos trabalhos estudados a integração de conteúdo gerado por um SGCW com um agente conversacional

através do uso de conceitos e ferramentas de Web Semântica, sendo esta a lacuna que esta pesquisa busca dar sua contribuição.

4 MODELO EDUARDO

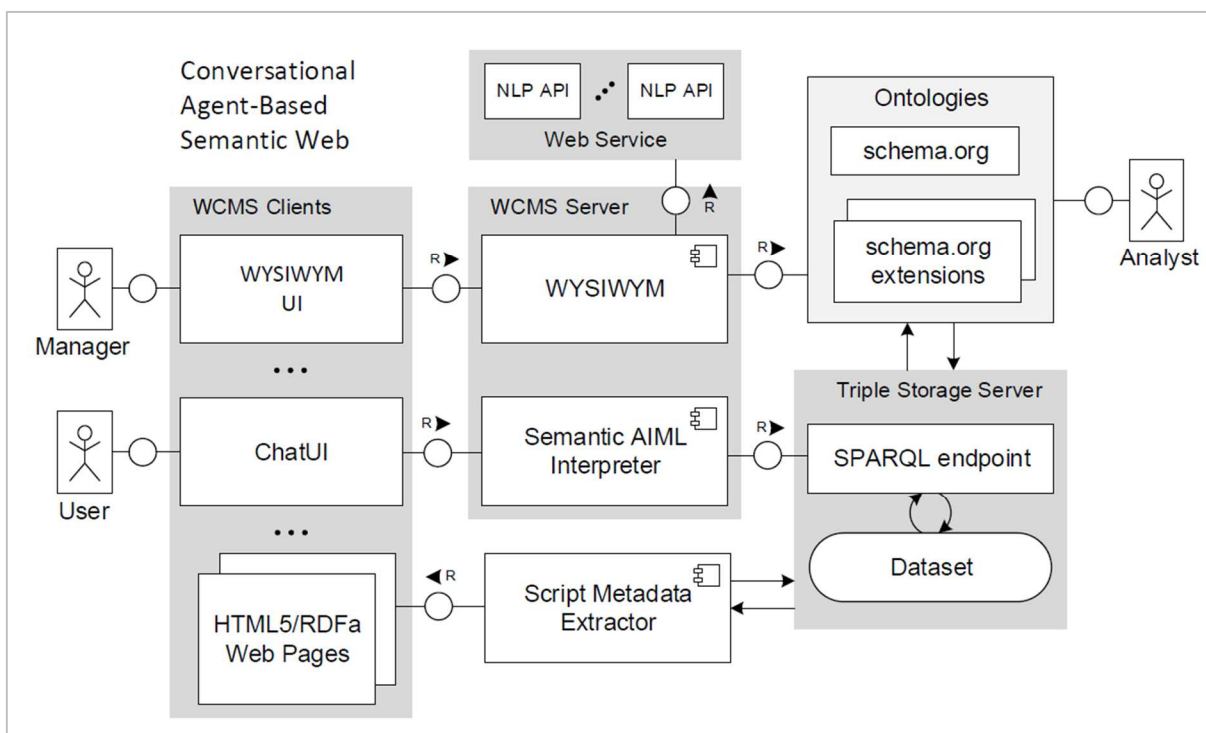
Neste capítulo é apresentado o modelo proposto neste trabalho, denominado *Eduardo*, a origem do nome é pois, seu apelido *Edu* corresponde ao diminutivo da palavra Educação área da instituição onde foi realizado seu primeiro estudo de caso, também foi escolhido um nome masculino para sair dos habituais femininos como ALICE e ELIZA por exemplo. A principal contribuição destacada neste modelo é o uso de Web Semântica como estratégia de expansão dos diálogos do agente conversacional. O Modelo está subdividido em três componentes, denominados *Semantic AIML Interpreter*, *Script Metadata Extractor* e *WYSIWYM*. Estes componentes e seu relacionamento serão descritos nos itens a seguir.

4.1 Modelo Conceitual

Uma visão geral da arquitetura está ilustrada na Figura 15. A notação TAM (*Technical Architecture Modeling*) definida pela SAP¹ com o propósito de cobrir tanto o nível conceitual como o nível de design de aplicações de software foi utilizada na sua construção. A notação TAM combina a descrição FMC (*Fundamental Modeling Concepts*) para o nível conceitual e a notação UML (*Unified Modeling Language*) para o nível de design (KNÖPFEL, 2007).

¹ <http://go.sap.com/brazil/index.html>

Figura 15 – Arquitetura proposta para Agente conversacional com base em Web Semântica



Fonte: Elaborado pelo autor.

O modelo proposto é baseado na arquitetura cliente/servidor. A camada servidor está representada na Figura 15 pelo componente *WCMS¹ Server*. Esta camada utiliza um SGCW como plataforma para proporcionar o suporte e a integração aos componentes do modelo denominados de *WYSIWYM* e *Semantic AIML Interpreter*. Os detalhes desses componentes são descritos nos tópicos 4.1.1 e 4.1.3 respectivamente. A camada servidor do modelo foi concebida de forma a proporcionar flexibilidade de integração dos seus componentes com SGCW distintos, de modo a ampliar o escopo de utilização. Também é nesta camada que se concentram as atividades principais de análise e de geração de conteúdo anotado semanticamente, bem como é a partir desta camada que elementos adicionais necessários serão integrados, por exemplo, serviços de apoio para PLN, ontologias ou *triple-stores*, como pode ser observado na Figura 15.

Já a camada cliente do modelo possui como objetivo implementar os componentes dedicados à interação direta com o usuário. Ela está representada na figura 15 pelo componente *WCMS Clients*, onde os componentes *WYSIWYM UI* e

¹ Web Content Management System – Sistema de Gerenciamento de Conteúdo Web (em inglês)

Chat UI servem como interface aos usuários do portal que implementar o modelo. Nesta camada também está considerada a localização de fontes de dados que podem ser de interesse para a geração de conteúdo anotado com base em extração de dados com base em elementos semânticos.

Outro componente é o servidor de repositório de triplas representado no modelo pelo *Triple Storage Server*, que deve ser capaz de realizar consultas Sparql e possibilitar que um procedimento automatizado submeta metadados em sua base de dados. Já o componente denominado *Script Metadata Extractor* é o responsável por realizar tal procedimento ao extrair os metadados das páginas HTML/RDFa disponibilizadas pelo *WCMS Server* e enviá-las ao *Triple Storage Server*.

O modelo também prevê uma organização de papéis de usuários, associados com funções específicas. Desta forma, são identificados na Figura 15 os seguintes papéis: *Manager*, *User*, *Analyst*. O papel denominado como *User* é o papel exercido pelos usuários que buscam informações diversas junto do web site. Este usuário não possui nenhuma interação com o conteúdo do web site do ponto de vista de sua manutenção ou atualização. Já os papéis denominados *Manager* e *Analyst* possuem esta atribuição, sendo que o primeiro está associado com a geração de conteúdo textual e sua associação em relação à base de conhecimento, enquanto que o segundo é um usuário especializado em representação de conhecimento e atua na geração de adaptações nas ontologias utilizadas de acordo com necessidades de cada domínio.

Maiores detalhes sobre cada um dos componentes do modelo Eduardo estão descritos nos tópicos a seguir. Junto com esta descrição estão também ilustrados e analisados os principais processos que envolvem o uso dos componentes do modelo.

4.1.1 WYSIWYM

O componente denominado de WYSIWYM no modelo Eduardo possui similaridades com os elementos conhecidos como *WYSIWYG*, que em geral consistem em um editor de texto comum, pré-instalado na maioria dos *SGCW's* e que disponibiliza um conjunto de ferramentas para edição de texto, possui uma parte

escrita em *Javascript* para a interface com usuário, com código que será interpretado pelo navegador e portanto seu processamento se dá do lado do cliente.

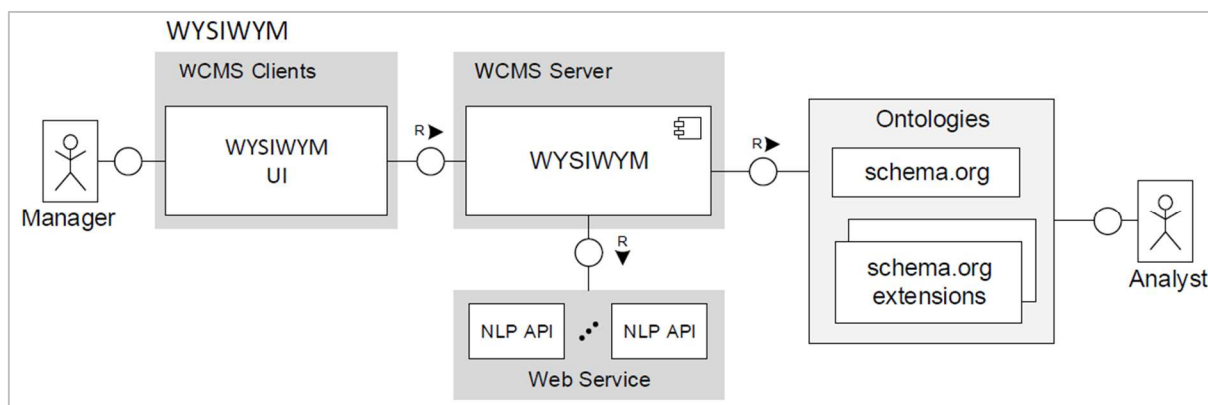
O diferencial que o modelo Eduardo oferece em relação ao editor *WYSIWIG* tradicional está na capacidade de anotar o conteúdo que está sendo editado através das marcações em *RDFa*. Para descrever esta situação, o termo cunhado por Khalili *et al* (2012) é *WYSIWIM*, um elemento de interface com o usuário que associa a visualização dos elementos com o seu significado, usualmente representado em uma base de conhecimento. Neste caso, adotou-se a utilização de uma ontologia para representação do conhecimento. No modelo é proposto o uso da ontologia *Schema.org*¹ e também extensões da mesma que são utilizadas para torná-la capaz de abranger as especificidades do site web no qual o agente será disponibilizado.

Outra característica do *WYSIWYM* é a capacidade de acessar bibliotecas de apoio para o processamento de linguagem natural, para realizar a anotação automatizada das entidades identificadas no texto por essas bibliotecas de apoio.

Aproveitando o fato dos *SGCW's* serem extensíveis, pretende-se disponibilizar aos seus autores uma variação da ferramenta proposta por Khalili *et al* (2012), para que o conteúdo semântico seja produzido pelos gestores de conteúdo.

Na Figura 16 estão ilustrados apenas os elementos do modelo Eduardo que correspondem ao componente *WYSIWIM*. Este elemento deve ser implementado como componente do *SGCW*, a finalidade é tirar proveito da infraestrutura que este já oferece. O componente *WYSIWYM* disponibiliza uma interface aos elementos do módulo *WCMS Clients*, que pode ser por exemplo, um cliente de páginas web, geralmente um navegador de internet. Representado no modelo como *WYSIWYM UI*, ele é escrito em *Javascript* e transforma a caixa de texto de edição de conteúdo em um elemento de edição de texto rico, possibilitando a interação dinâmica dos usuários para anotação de dados, utilizando de microformatos e *RDFa*.

¹ <http://schema.org>

Figura 16 – Editor de texto Semântico *WYSIWIM*

Fonte: Elaborado pelo autor.

Uma funcionalidade prevista no *WYSIWIM* é a capacidade de identificar as entidades automaticamente, sendo que para isso ele realiza requisições Ajax (*Asynchronous Javascript and XML*) às NLP API previamente configuradas em seu ambiente. Requisições Ajax também são necessárias para que o *WYSIWIM* ofereça aos seus usuário as entidades, relações e propriedades da representação de conhecimento adotada no modelo, a partir do Schema.org e suas extensões.

Todo o SGCW disponibiliza aos seus gestores de conteúdo uma ferramenta escrita em Javascript para edição de texto e/ou gerenciamento de mídias. Alguns exemplos dessas ferramentas são TinyMCE¹, CKEditor², Kupu³. O aspecto relevante não é saber qual ferramenta adotar, o importante é que essas ferramentas são comumente extensíveis, porém seus componentes são geralmente dependentes dos SGCW's onde foram pré-instalados, ou seja, tais editores tem distribuições diferentes para cada SGCW.

Então pode-se afirmar que o TinyMCE disponível como componente no Wordpress⁴ não é o mesmo que é distribuído junto com o Joomla⁵ ou mesmo o Plone⁶. O que é proposto neste modelo conceitual é estender um *WYSIWYG* através da sua própria arquitetura de componentes para disponibilizar uma interface aceitável para anotação de dados em RDFa e também que consulte uma variedade

¹ <https://www.tinymce.com>

² <http://ckeditor.com/>

³ <https://plone.org/documentation/manual/plone-4-user-manual/using-kupu-as-visual-editor>

⁴ <https://wordpress.org/>

⁵ <https://www.joomla.org/>

⁶ <https://plone.org/>

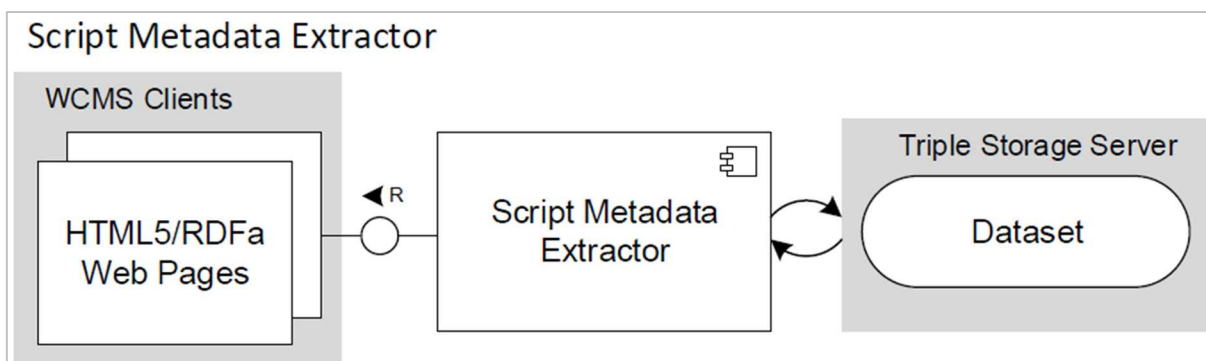
de ontologias para servir ao responsável pelo conteúdo do website, para que ele defina quais entidades e propriedades deseja utilizar em suas anotações. Sendo assim, o componente *WYSIWYM* é dividido em *WYSIWYM* e *WYSIWYM UI*, foi adotada sigla UI (*User Interface*) por se tratar da camada de software que servirá de interface para o usuário com o papel de *Manager*.

Outro agente humano importante para tornar esse modelo possível é o *Analyst*, pois a ele cabe a responsabilidade de gerenciar os arquivos da ontologia estendida da *schema.org*. O diferencial deste componente do modelo está justamente na funcionalidade de consumir as ontologias estendidas.

4.1.2 Metadata Extractor

O modelo prevê um script para extração dos metadados disponibilizados pelos usuários gerenciadores de conteúdo através do componente *WYSIWYM*. Sendo assim, para facilitar a sua descrição, na Figura 17 foram extraídos do modelo Eduardo apenas os elementos relacionados com o componente denominado *Script Metadata Extractor*.

Figura 17 - Metadata Extractor



Fonte: Elaborado pelo autor.

Este componente descrito na Figura 17 foi definido com uma dinâmica na qual periodicamente, ou quando requisitado, realiza uma requisição para as páginas HTML5 previamente anotadas em RDFa. Em sua implementação deve realizar o procedimento de extração do conteúdo RDF destas páginas e convertê-lo para um formato compatível com o Triple Storage, tal como o formato RDF/XML ou

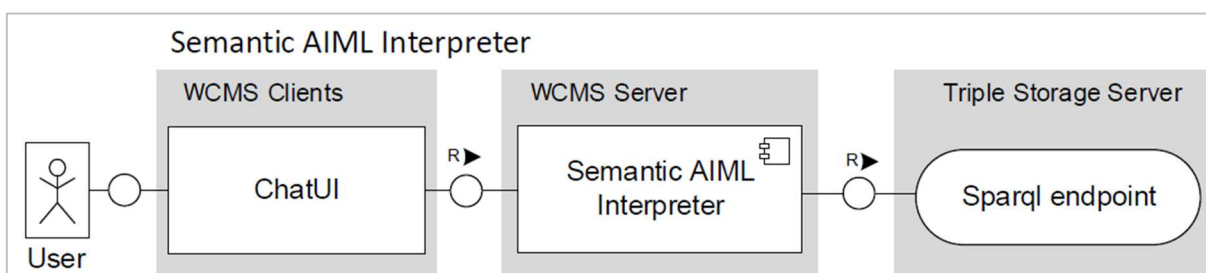
Turtle por exemplo. Em seguida ele deve submeter essas dados ao Triple Storage Server.

4.1.3 Semantic AIML Interpreter

O modelo prevê que para o usuário do site web será disponibilizada uma tela de chat no qual este poderá interagir com o Agente Conversacional. A comunicação deste componente com o conteúdo anotado semanticamente será possível através de requisições Ajax para o componente *Semantic AIML Interpreter* disponibilizado como um componente do SGCW.

Nenhuma alteração no código fonte do SGCW, nem mesmo em sua Base de Dados deverá ser realizada, sendo que a implantação do Agente Conversacional deverá ser o menos intrusiva possível e sua infraestrutura servirá como base para o modelo proposto. A Figura 18 descreve estes componentes.

Figura 18 - *Semantic AIML Interpreter*



Fonte: Elaborado pelo autor.

O módulo *Semantic AIML Interpreter* disponibiliza um interface de *chat* escrita em Javascript, além de outros arquivos estáticos necessários a sua renderização, tais como CSS (*Cascading Style Sheets*) e arquivos de imagens.

Conforme esse modelo conceitual da figura 18, é possível identificar que o usuário interage com uma interface de *chat*, a qual faz uma requisição *HTTP* (*Hypertext Transfer Protocol*) ao módulo disponibilizado pelo SGCW, que por sua vez processa as requisições dos usuários, podendo consultar um *SPARQL endpoint* e em seguida retornar a resposta ao usuário.

4.1.4 Expansão da Ontologia

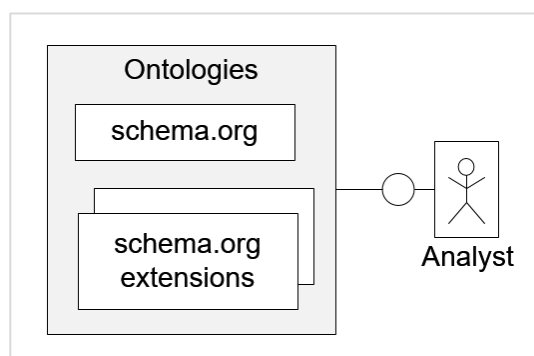
Para manter a possibilidade de atualização de conteúdo utilizado pelo agente conversacional, é adotado um mecanismo de anotação semântica que implica em expansões sucessivas da ontologia utilizada.

O ponto de expansão da ontologia não é um procedimento simples, pois é necessário conhecimento de Web Semântica para criação desse componente. Este procedimento não pode ser automatizado completamente, porém algumas ferramentas auxiliam na realização dessa tarefa.

Para aumentar a confiabilidade dos dados da base de conhecimento que irá alimentar o diálogo com os usuários, foi proposto, no modelo Eduardo, que usuários qualificados assumam tal responsabilidade. A expansão dos diálogos depende tanto das ontologias quanto das consultas em formato *sparql*, que também são responsabilidade deste usuário.

A Figura 19 ilustra esta concepção para o funcionamento deste componente do modelo, tendo como base a interação eventual com um especialista capaz de realizar a ampliação da ontologia usada de acordo com o domínio do web site que irá utilizar o modelo.

Figura 19 - Expansão da Ontologia



Fonte: Elaborado pelo autor.

A interação prevista é necessária em poucas situações, basicamente de acordo com a modificação no domínio no qual o sistema é empregado.

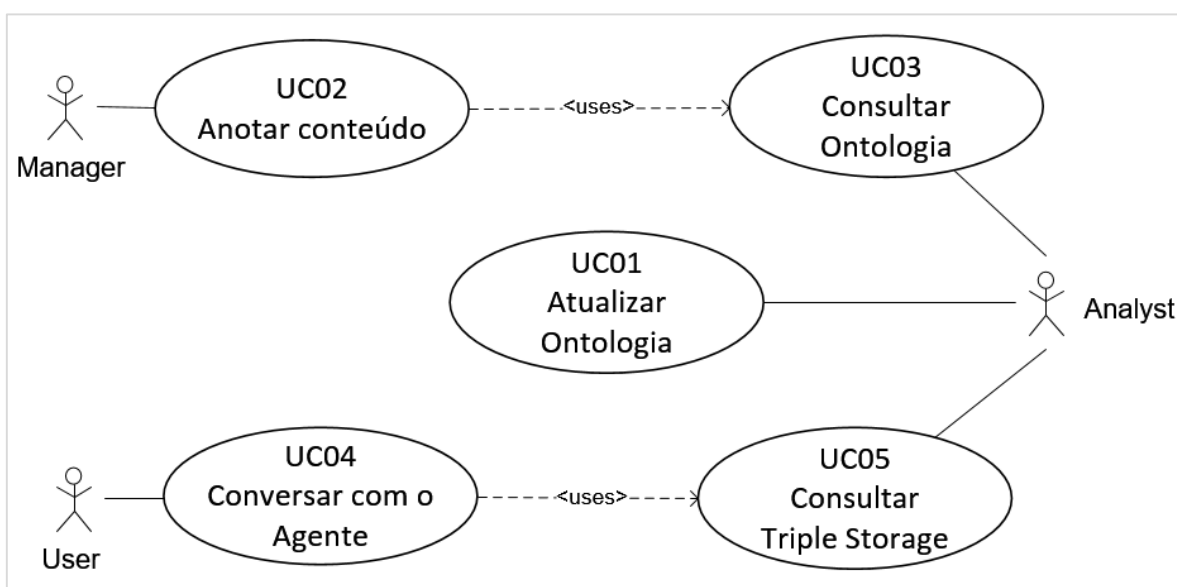
4.2 Requisitos e Definição do Perfil dos Usuários

Neste item são apresentados os principais requisitos levados em consideração pelas características do modelo Eduardo.

4.2.1 Casos de Uso

Esta subseção tem por objetivo apresentar e descrever os principais casos de uso identificados como requisitos do modelo Eduardo. Na Figura 20 é apresentado o Diagrama de Casos de Uso. Neste diagrama é possível identificar a interação com o modelo por três níveis de usuários: o *User*, o *Manager* e o *Analyst*. O *User* é tido como um visitante e portanto não precisa se identificar para conversar com o Agente. O usuário *Manager* precisa ter acesso como editor do SGCW e o usuário *Analyst*, além de perfil idêntico ao *Manager* ainda tem acesso físico aos servidores que disponibilizam os serviços.

Figura 20 - Casos de Uso



Fonte: Elaborado pelo autor.

O caso de uso descrito no diagrama pelo código UC01 e descrito como Atualizar Ontologia é o requisito responsável por permitir a ampliação dos diálogos do agente. A ontologia possibilita determinar o domínio e o escopo dos diálogos,

sendo que foi considerado o uso de uma ontologia já existente e um usuário responsável por usar e manter a ontologia proposta.

O caso de uso de código UC02 e descrito como Anotar Conteúdo é responsável por permitir que o usuário *Manager* seja capaz de anotar o conteúdo usando a linguagem RDFa. Ele prevê a disponibilização de um componente de interface com o usuário e a necessidade de consultar a ontologia que permite definir entidades, propriedades, relações e tipos de dados. O caso de uso UC03 então torna-se pré-requisito para que essa tarefa seja concretizada.

Já o caso de uso UC03, descrito como Consultar Ontologia, estabelece a necessidade de um ambiente que disponibilize as ontologias para serem consultadas pelos usuários com perfil *Analyst* e que sejam disponibilizadas para que os usuários *Manager* possam anotar o conteúdo.

O caso de uso UC04, descrito como Conversar com o Agente, é o requisito responsável por permitir que usuários que estejam visitando o site possam interagir com o agente conversacional através de uma interface de chat, onde ele possa realizar perguntas e receber respostas. Para realização de tais consultas o caso de uso UC05 se torna um pré-requisito.

E o caso de uso UC05 descrito como Consultar Triple Storage é um requisito fundamental para o agente conversacional proposto, esse caso de uso atende essa necessidade e disponibiliza uma interface para que o usuário de perfil *Analyst* possa testar as consultas que pretende disponibilizar para o Agente conversacional.

4.3 Demais considerações acerca do modelo

O modelo proposto se destaca-se também dos trabalhos analisados até o momento pelo fato de disponibilizar conteúdo semântico através de marcação RDFa (que pode ser embutida nas página HTML e divulgadas diretamente pelos autores), torna esse conteúdo legível tanto por máquina quanto por humanos o que ajuda nos processos de automatização da atualização dos diálogos. Esse conteúdo poderá ser consumido diretamente por outros sistemas e dar origem a aplicações dos mais diversos domínios e esta estrutura ainda pode proporcionar a otimização nos resultados em motores de buscas.

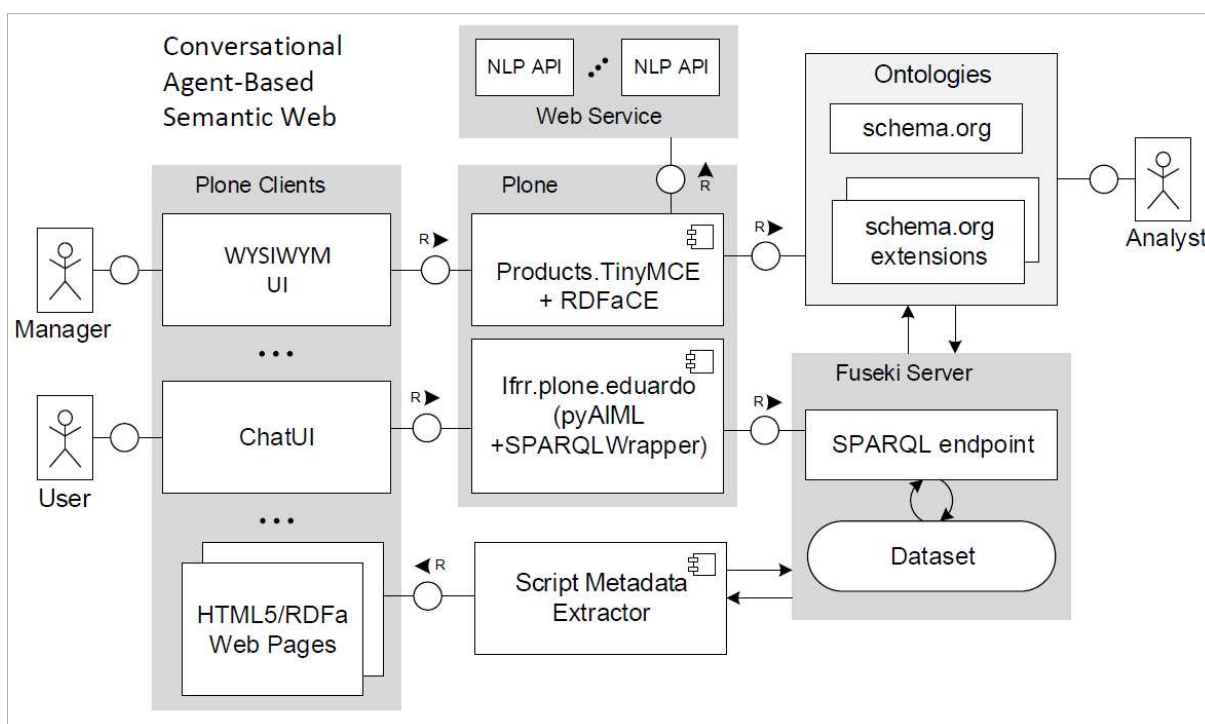
O principal diferencial está na automatização da atualização dos diálogos, pois os mesmos são extraídos do SGCW para uma base de dados de triplas, que são consultadas para geração dos diálogos, caso haja alguma atualização na base do SGCW, basta refazer os procedimentos que já foram automatizados, para que os diálogos sejam atualizados, evitando assim o trabalho manual de edição dos arquivos escritos em linguagem AIML (Artificial Intelligence Markup Language).

5 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são apresentadas as etapas realizadas para o desenvolvimento de um protótipo que aplica o Modelo Eduardo. O objetivo desta implementação foi proporcionar o ambiente necessário para a avaliação de aspectos propostos pelo modelo.

No capítulo anterior foi apresentado o modelo Eduardo, sendo que na Figura 15 foi apresentado o modelo genérico de sua arquitetura. A utilização do modelo pressupõe a integração com outros sistemas. Para demonstrar onde os detalhes da implementação são percebidos, deve ser observada a composição final da implementação com os componentes escolhidos, de acordo com a Figura 20.

Figura 21 - Implementação do Modelo Eduardo no Plone SGCW



Fonte: Elaborado pelo autor.

A Figura 21 apresenta a abordagem utilizada para implementação dos componentes no SGCW Plone, sendo que:

- o componente *Products.TyneMCE* junto com o componente *RDFaCE* proporcionam a instância do componente *WYSIWYM*;

- o componente denominado *ifrr.plone.eduardo* (composto pelo interpretador *pyAIML* e pelo *SPARQLWrapper*) constituem a instancia do *Semantic AIML Interpreter*;
- o *Fuseki Server* foi utilizado como a instância do *Triple Storage Server*.

A Secretaria de Comunicação com base no projeto de Identidade Padrão de Comunicação Digital do Poder Executivo Federal, desenvolveu a implementação do modelo utilizando o SGCW Plone (SECOM, 2014). Este gerenciador também é adotado no âmbito do IFRR e portanto foi escolhido para servir de base para o desenvolvimento dos componentes propostos no modelo Eduardo.

O Plone é escrito em Python, e as bibliotecas disponíveis para essa linguagem são *pyAIML* que serve para interpretar a linguagem *aiml*, *SparqlWrapper* que realizar consultas *Sparql* em um *Sparql endpoint*, além da biblioteca *RDFlib* para analisar o *RDF* e extrair *RDFa* das páginas web anotadas no site.

5.1 Implementação de componentes

Neste item é apresentada a descrição dos aspectos de implementação dos componentes do protótipo para o modelo Eduardo, onde são melhor abordadas as tecnologias, frameworks e bibliotecas utilizadas, incluídas também as interfaces gráficas.

Para o desenvolvimento do protótipo foi utilizado o Processo Unificado da Rational conhecido como RUP (*Rational Unified Process*), que é um processo de engenharia de software criado para apoiar o desenvolvimento orientado a objetos, proporcionando uma forma sistemática para se obter vantagens com a padronização dos processos de desenvolvimento e com base no uso da UML (KRUCHTEN, 2004).

Segundo COSTA, 2013, p. 65, as fases do RUP são as seguintes:

- Fase de Concepção - a fase de iniciação ou concepção contém os workflows necessários à concordância das partes interessadas - com os objetivos, a arquitetura e o planejamento do projeto. Nesta fase, os requisitos essenciais do sistema são transformados em casos de uso.

- Fase de Elaboração - A fase de elaboração será apenas para o projeto do sistema, buscando complementar o levantamento e a documentação dos casos de uso, voltado para a arquitetura do sistema, revisar a modelagem do negócio para os projetos e inicia a versão do manual do usuário.
- Fase de Construção - na fase de construção, começa o desenvolvimento físico do software, produção de códigos, e testes.
- Fase de Transição - nesta fase ocorre a entrega do software, onde é realizado o plano de implantação, acompanhamento e qualidade do software.

Tabela 4 - RUP

Fase	Descrição	Artefatos	Ferramentas
Concepção	Levantamento dos requisitos e definição dos cenários de uso	Modelo conceitual e Diagrama de Casos de Uso dos cenários elencados	Microsoft Visio
Elaboração	Documentação	Diagrama de classes dos principais componentes	
Construção	Construção do protótipo	Codificação do Agente, Edição da Ontologia e Versionamento e preparação do código para <i>deploy</i>	Protégé, PyCharm 4 Community, Gitlab, RDFa, schema.org, TinyMCE, RDFaCE
Transição	Disponibilização do software	Instalação dos componentes nos servidores de produção	Fuseki, Google appengine, Plone

Fonte: Elaborado pelo autor.

A

Tabela 4 apresenta as respectivas fases, bem como descreve as ferramentas utilizadas.

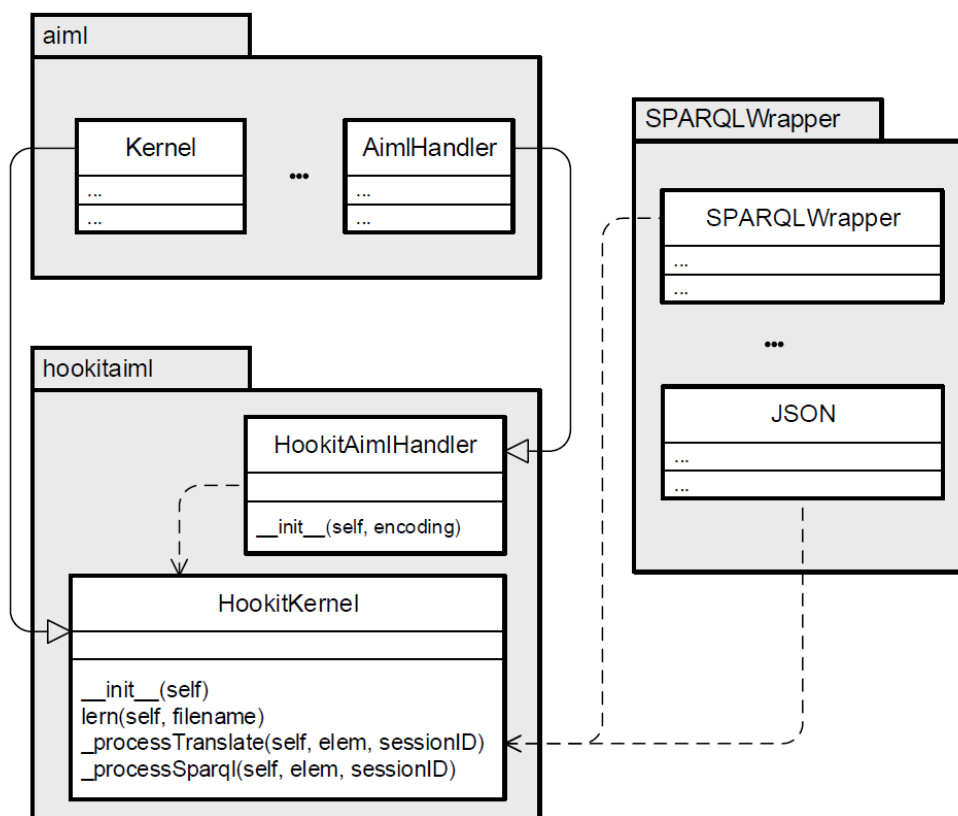
5.1.1 Implementação do *Semantic AIML Interpreter*

O componente *Semantic AIML Interpreter* proposto no Modelo Eduardo foi implementado ao estender a biblioteca pyAIML, retratada na Figura 22 apenas como *aiml*.

Por padrão esta biblioteca não interpreta consultas *sparql*, então como pode ser visto na classe *HookitKernel* do pacote *hookitaiml* que estende da classe *Kernel* do pacote *aiml*, foram sobrescritos os métodos `__init__` (construtor) e *lern*. Além disso, outros dois métodos foram propostos, o `_processSparql` responsável por interpretar a tag *sparql* proposta para a linguagem *aiml*, e o `_processTranslate` que tem o propósito de traduzir as entidades e os parâmetros para a ontologia escolhida. Como o agente utiliza a linguagem a partir do português e o núcleo da ontologia Schema.org encontra-se em inglês esse método se fez necessário.

Outras adaptações necessárias no código foram feitas por herança de modo que não interferissem no código original da biblioteca pyAIML. Na Figura 22 também é possível observar que a biblioteca *SparqlWrapper* foi utilizada para interpretar as consultas *sparql*, sendo que a resposta a essa consulta foi serializada em formato JSON¹.

¹ <http://www.json.org/>

Figura 22 - Diagrama de Classes do *Semantic AIML Interpreter*

Fonte: Elaborado pelo autor.

Um exemplo de consulta SPARQL embutida em um documento AIML é apresentado na Figura 23. Destaca-se que na linha 4 são capturados dois parâmetros através da tag `<star>`, sendo que na linha seguinte o atributo *endpoint* possui como valor o endereço do *endpoint* onde a consulta deve ser realizada. Esse atributo é opcional e caso não seja definido a consulta será realizada por padrão em `http://pt.dbpedia.org/sparql`. O código responsável pela consulta é definido como conteúdo do elemento `sparql`, onde os parâmetros são definidos através da tag `<set>`. Esta abordagem foi inspirada em (KIMURA; KITAMURA, 2006).

Figura 23 - Exemplo de uma consulta SPARQL em um arquivo AIML

```

1 <category>
2   <pattern>QUAL 0 * DO *</pattern>
3   <template>
4     0 <star index="1"/> do <star index="2" /> é:
5     <sparql endpoint="http://localhost:3030/ifrr/sparql">
6       PREFIX schema:&lt;http://schema.org/&gt;
7       SELECT ?s ?o
8       WHERE {
9         ?s ?property ?x FILTER regex(?x, '^<star index="2"/>$', 'i')
10        ?s schema:<set name="prop"><translate><star index="1"/></translate></set>
11          <condition name="prop">
12            <li value="address"><set name="aux">?oo</set></li>
13            <li><set name="aux">?o</set></li>
14          </condition> .
15          <condition name="prop" value="address">
16            ?oo schema:streetAddress ?o
17          </condition>
18        }
19        LIMIT 1
20      </sparql>
21      . Para mais informações acesse a página de &lt;a href='/IFRR/@@uo-view'&gt;
22        contatos da instituição&lt;/a&gt;.
23    </template>
24 </category>

```

Fonte: Elaborado pelo autor.

5.1.2 Implementação do WYS/WYM no Plone SGCW

O Plone é extremamente extensível, ele utiliza o *TinyMCE* como editor de texto WYSIWYG padrão na versão 3.2.6, que é compatível com o RDFaCE citado anteriormente (KHALILI; AUER; HLADKY, 2012).

O editor TinyMCE é disponibilizado no Plone através do componente Products.TinyMCE. Para seu uso foi necessário configurar no Products.TinyMCE o plugin RDFaCE e realizar algumas adaptações no mesmo para que funcionasse no Plone, já que sua distribuição padrão é para a linguagem PHP.

Voltando a analisar a Figura 12, nota-se que existe um componente Proxy para acessar APIs de processamento de linguagem natural, o *RDFa Proxy for Enricher APIs*. Este componente possibilita um canal único para o uso das APIs de enriquecimento semântico de texto descritas no modelo como *Semantic Text Enricher*, sendo que as APIs suportadas são evriE, Ontos¹, AlchemyAPI e etc. A

¹ <http://www.ontos.com/>

considerado depreciado por usar uma técnica para identificar automaticamente os dados disponibilizados pelas página HTML do Schema.org (CYGANIAK; HAUSENBLAS, 2015) e também pelo fato do Schema.org já disponibilizar seu vocabulário no formato JSON-LD.

Porém como RDFaCE é anterior a essa iniciativa, foi codificado apenas para ler os dados em JSON. Então optou-se por continuar a usar a abordagem considerada depreciada e posteriormente contribuir para o projeto editando o RDFaCE para aceitar os arquivos no formato JSON-LD, não comprometer assim o cronograma estabelecido para este trabalho. Mesmo usando a abordagem indicada, os scripts precisaram ser atualizados para o novo formato em que o Schema.org disponibiliza seu vocabulário, o que acarretou menos trabalho do que reescrever o RDFaCE para suportar os dados em JSON-LD.

Esse trabalho foi necessário pois o RDFaCE não acessa diretamente o vocabulário Schema.org para sugerir as entidades, propriedades e dados primitivos aos seus usuários. Como em javascript a manipulação de dados em JSON é trivial, acredita-se que por este motivo o autor optou por essa abordagem, porém, agora que os colaboradores do vocabulário Schema.org já disponibilizam o vocabulário em JSON-LD, isso tornará mais aceitável sua utilização pelo RDFaCE em novas versões. Porém o principal motivo é que deseja-se disponibilizar aos usuários do protótipo desenvolvido o vocabulário estendido, ou seja, necessita-se do arquivo em JSON com essas informações para integrar o RDFaCE com o vocabulário entendido e assim demonstrar como os diálogos poderão também ser ampliados.

5.1.3 Implementação do *Script Metadata Extractor*

O *Script Metadata Extractor* é o elemento responsável por extrair os Metadados de páginas HTML5 anotadas semanticamente com o RDFaCE descrito no tópico anterior. O mesmo script pode ser utilizado para extrair dados em RDF/XML, JSON-LD e Turtle de qualquer site, bastando configurar as url's em seu dicionários de dados.

Este componente utiliza as bibliotecas *rdflib* para acessar os recursos e extrair os metadados e gravá-los em arquivos de texto, em seguida utiliza a biblioteca *subprocess* para executar os binários do *Fuseki* e enviá-los para um dataset.

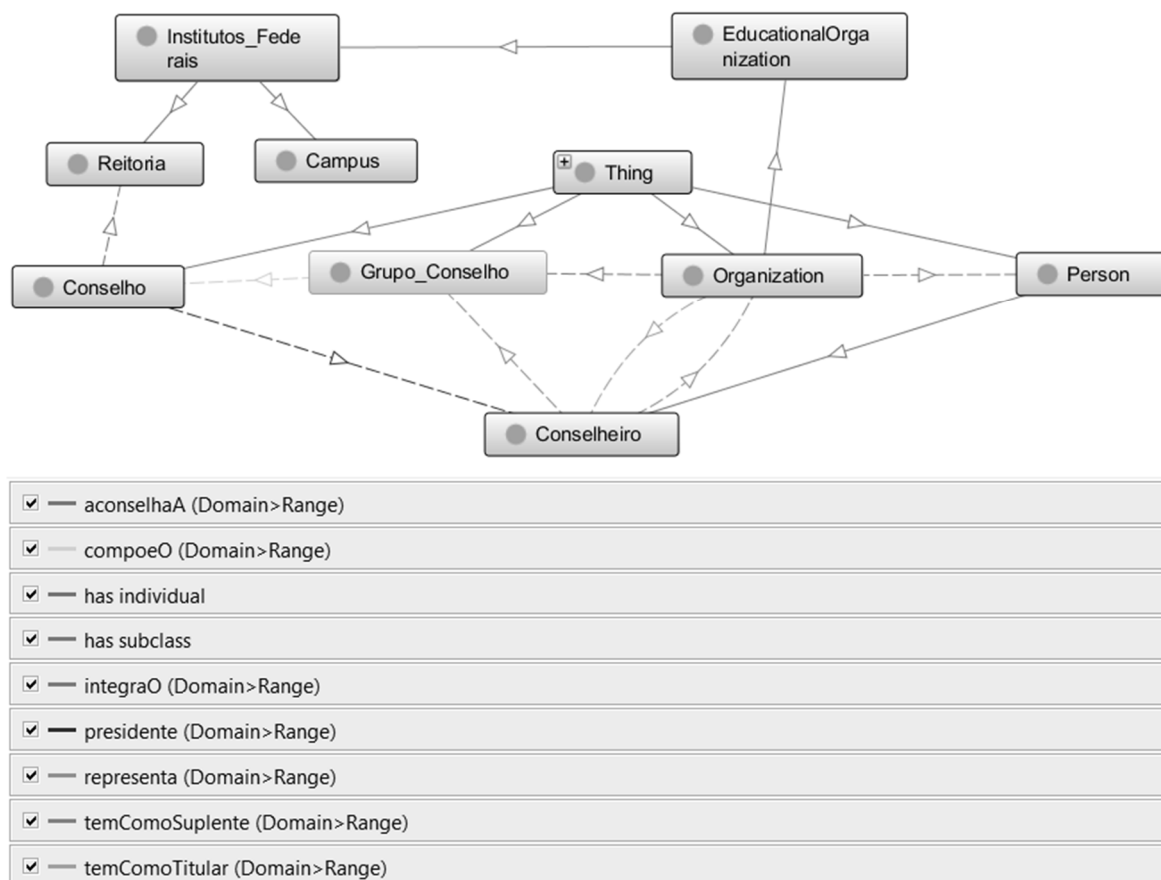
5.2 Ontologia

A ontologia *Schema.org* é amplamente utilizada para SEO, apesar do uso dos microformatos do *Schema.org* não melhorar o índice de *SERP* (*Search Engine Results Page*) do site no Google por não influenciar diretamente na classificação, o fato de facilitar a leitura dos *crawlers* (robôs de busca) e conseguir criar os *Rich Snippet* (resultados do Google com *ranking* de estrelas e *review*) (MARCOS, 2013). Utilizar os microformatos ainda é uma grande vantagem, pois irá afetar a taxa de cliques (*Click Through Rate* - CTR) considerando que o conteúdo irá aparecer com maior destaque e atrair a atenção de quem estiver buscando (FONSECA, 2012), então é um fator que influencia a decisão de pessoas

Porém o *Schema.org* não abrange as especificidades dos Institutos Federais, foco desta implementação, elementos como conselhos, conselheiros, campus, editais e outros termos presentes na taxonomia de instituições de ensino não são contemplados. Então a *Schema.org* foi adotada no protótipo como ontologia de base, para demonstrar a possibilidade de expansão da ontologia *Schema.org*. Pode-se visualizar na Figura 25 um trecho da ontologia utilizada pelo agente conversacional para anotar dados referentes aos conselhos e conselheiros existentes em instituições de ensino superior.

O *Schema.org* utiliza RDFS para definição de seu vocabulário e em maio de 2015 apresentou um novo mecanismo de extensão baseado em RDFa e JSON-LD. *Schema.org* fornece um vocabulário básico para descrever o tipo de entidades mais comum e que as aplicações web mais precisam. Muitas vezes existe a necessidade de vocabulários mais especializados e/ou mais profundos, que podem ser construídos usando *schema.org* como núcleo. Os mecanismos de extensão facilitam a criação de tais vocabulários adicionais (SCHEMA.ORG, 2015).

Figura 25 – Parte da ontologia para o Mapeamentos dos Conselhos



Fonte: Elaborado pelo autor.

A Figura 25 representa parte da ontologia que será utilizada para mapear os conselhos e conselheiros em uma instituição de ensino superior, esta ontologia especificamente foi criada para atender as expectativas dos usuários coletada na Teste Antecipado. Na figura, as entidades *Thing*, *Person*, *Organization* e *EducationalOrganization* já são fornecidas pelo Schema.org, as setas não tracejadas indicam uma relação de herança (*has subclass*), as demais entidades (*Institutos_Federais*, *Reitoria*, *Campus*, *Conselheiro*, *Grupo_Conselho* e *Conselho*) foram criadas para expandir o vocabulário, mais especificamente os Institutos Federais, que possuem uma composição multicampi.

6 AVALIAÇÃO E TESTE

Três experimentos foram realizados com o protótipo implementado. O primeiro experimento foi o *Teste Antecipado* e teve como base um questionário com perguntas abertas, cujo objetivo foi verificar as expectativas dos usuários com relação ao agente conversacional. O segundo experimento foi uma *Avaliação de Funcionalidade* baseada em cenários, projetada para verificar problemas críticos que poderiam prejudicar a verificação dos objetivos do modelo. O terceiro experimento avaliou a *Aceitabilidade* do modelo, verificando sua facilidade de uso e a percepção de sua utilidade por parte dos usuários.

6.1 Teste Antecipado

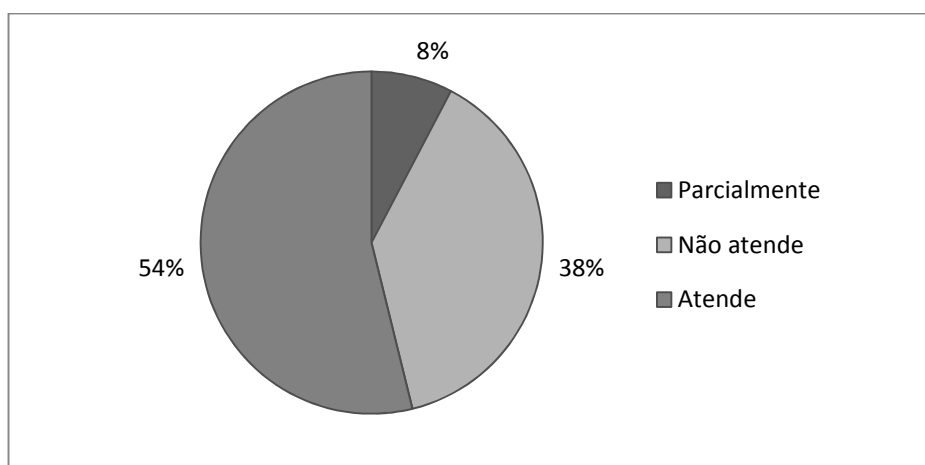
Nesta etapa foi solicitado a um grupo de usuários que escrevessem questões sobre o IFRR para que o sistema desenvolvido fosse utilizado para gerar a resposta. Além da pergunta, solicitou-se aos usuários indicar a resposta que esperavam para cada pergunta. Esta abordagem pretendia identificar junto do público usuário do web site qual o tipo de questão que seria enviada para o agente conversacional e quais as respostas e padrões esperados e considerados adequados pelos usuários. Além disso, este experimento permitiu identificar o tipo de informação que necessitaria estar representado na base de conhecimento do protótipo.

O grupo de entrevistados foi composto por dez pessoas. O perfil dos entrevistados é composto de alunos, professores, técnicos administrativos e membros da comunidade, com idade variando entre 17 e 37 anos, atuantes nas áreas de Análise e Desenvolvimento de Sistemas, Pedagogia, Comunicação, Ciências Biológicas e outras.

As perguntas e respostas estão disponíveis no Anexo A desse trabalho. Para exemplificar seguem algumas perguntas sugeridas pelos entrevistados: “Qual o nome do Reitor?”, sendo que para responder a essa pergunta é necessário que informações relacionadas a pessoas e cargos estejam disponíveis. Outro exemplo obtido no experimento foi a pergunta “Qual o endereço do Campus Amajari?”, sendo que para responder a essa pergunta é necessário que a ontologia permita anotação de informações de localização geográfica.

Estas perguntas foram utilizadas em interações com o protótipo e em seguida a resposta do agente foi comparada com a resposta esperada por cada entrevistado, para identificar se atendia ou não as suas expectativas. Este processo foi realizado para coletar os tipos de informações desejadas pelos 10 entrevistados e assim caso os dados representados no Schema.org não atendessem as perguntas, seria necessário ampliar a ontologia para representação das informações necessárias para a resposta. Este foi um experimento relevante para identificar como os usuários elaboram suas perguntas e também identificação dos dados.

Figura 26 - Atendimento a expectativa dos entrevistados



Fonte: Elaborado pelo autor.

Uma forma efetiva de utilização dos testes para obter informações úteis para o desempenho dos sistemas pode ser justamente realizar o mesmo no início do processo. Testes antecipados podem pegar quaisquer erros ou falhas antes que se tornem mais caras para solucionar (UNGER; CHANDLER, 2009, p. 226). A etapa aqui descrita foi realizada antes mesmo de existir uma versão funcional do protótipo e ajudou a identificar os principais assuntos relacionados ao IFRR que deveriam ser considerados para elaboração dos diálogos do agente. A Figura 26 permite observar que embora o protótipo ainda estivesse em fase inicial de desenvolvimento, o mesmo conseguiu responder a 54% das perguntas realizadas pelos entrevistados, sendo que 38% das perguntas não foram respondidas e 8% das perguntas foram respondidas parcialmente.

O fato de 54% das perguntas terem sido respondidas, indica que o Schema.org disponibiliza uma ampla gama de entidades, propriedades e relações e para responder a perguntas relacionadas a pessoas e cargos. Por exemplo, foi

criado um webservice para o sistema administrativo do IFRR e um componente cliente para o portal, de forma que disponibilizasse automaticamente em páginas HTML5/RDFa as informações dos servidores, seus cargos e informações de contato, bem como das unidades administrativas.

Esse procedimento foi considerado necessário em virtude do grande fluxo de servidores ocupando cargos e a troca da administração esporádica, o que acarretaria um fluxo inviável de atualização manual nas páginas que foi evitado pelo fato das informações já estarem atualizadas em outra base de dados. Esse webservice ainda será ampliado, e futuramente poderá disponibilizar metadados em RDFa de cursos, docentes e alunos. Para este trabalho foi utilizado apenas para gerar um volume inicial de dados anotados e também para validar os conceitos empregados pelo modelo Eduardo.

Os 38% das perguntas que não foram respondidas, ocorrem devido ao fato da ontologia Schema.org não dispor das entidades, propriedades e/ou relações necessárias para anotação dos dados, ou pelo fato dos dados não estarem anotados e disponibilizados no site. Esse percentual pode diminuir consideravelmente quando os trabalhos de anotação com o componente *WYS/WYM* forem amplamente utilizados pelos editores do site web.

O percentual de 8% de questões que foram descritas no gráfico como parcialmente atendidos é explicado pelos testes que estavam sendo realizados na anotação dos dados com o *WYS/WYM* para anotar os dados dos conselhos e conselheiros, mais detalhes sobre essa validação de ampliação da ontologia está disposta no tópico 6.2.3.

6.2 Avaliação de Funcionalidade

Segundo Dey (2001), a avaliação por cenários é utilizada pela comunidade acadêmico-científica para avaliar projetos sensíveis ao contexto e de acordo com Satyanarayanan (2001) é também utilizada em ambientes ubíquos.

Neste item é descrita uma abordagem de avaliação na qual definiram-se cenários representativos de futuras situações a serem atendidas pelo sistema. Com base na especificação dos cenários e na análise do uso do protótipo no intuito de

atender às demandas descritas, é possível avaliar fragilidades ou pontos positivos do protótipo.

6.2.1 Cenário 1 – Anotação do Conteúdo com RDFa

Este cenário pretende demonstrar a utilização do componente WYSIWYM proposto como componente do modelo Eduardo para anotar o conteúdo de uma página web por um usuário com o perfil *Manager* que tenha acesso a área administrativa do Plone SGCW.

Descrição do cenário:

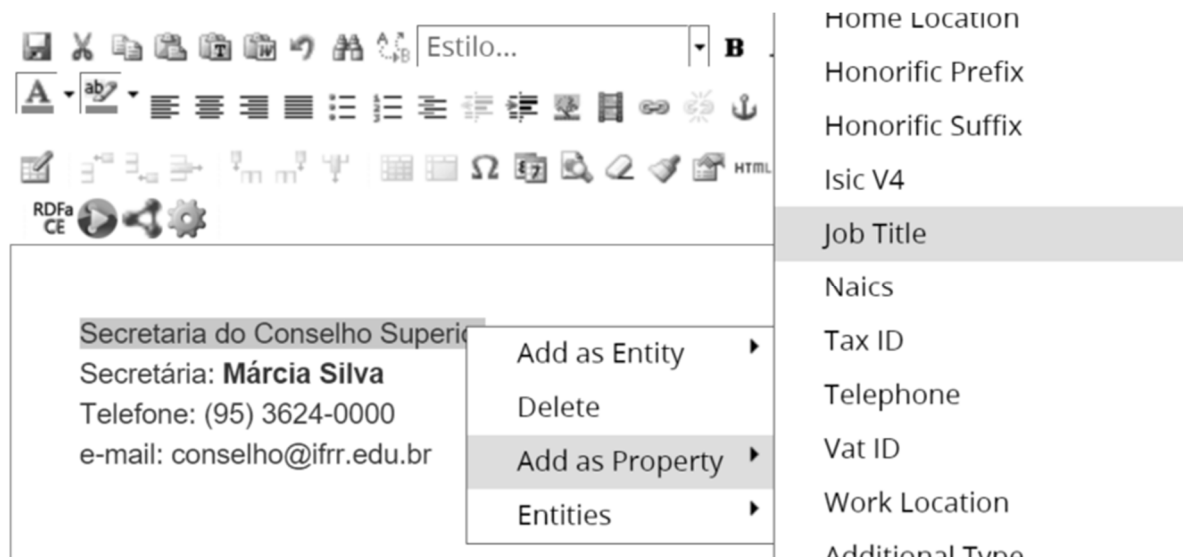
Márcia é chefe de gabinete na Reitoria do IFRR e lhe foi solicitado que publique no portal institucional as informações pertinentes ao conselho superior. O Conselho Superior é um órgão consultivo e de deliberação, com estrutura, atribuições e funcionamento definidos na lei 11.892/08, no estatuto e em regimento Interno. Sabendo disso, Márcia então coleta e publica informações como: Legislação, Composição e Estrutura, Local e Calendário de Reuniões e etc. Márcia também participa do Conselho como secretária e ela é o principal meio de contato com os membros do conselho. Para publicar tais informações e otimizá-las para que o Agente Conversacional e demais mecanismos de busca. Márcia verifica no conteúdo das páginas Entidades tais como, Pessoas, Organizações e Lugares e procura marcá-los, selecionando cada um e definindo as respectivas entidades através de um painel suspenso que aparece sempre que pressiona o botão direito do mouse sobre um texto pré-selecionado. O texto recebe um destaque visual para identificar que já foi marcado, sendo que após marcar uma entidade Márcia pode clicar novamente com o botão direito do mouse sobre ela e adicionar propriedades e relações. Caso Márcia opte por maior comodidade, ela pode usar o botão de anotação automática de conteúdo, que identifica as entidades mais comuns.

Detalhes de execução do cenário:

O componente WYSIWYM captura os eventos de clique de Mouse efetuados pelo usuário e adiciona uma tag `` na seleção contendo o atributo RDFa

definido na marcação feita pelo próprio usuário. Como pode ser visto na Figura 27 onde Márcia estaria anotando suas informações de contato como secretária.





Figura 27 - Interface do componente WYSIWYM



Fonte: Elaborado pelo autor.

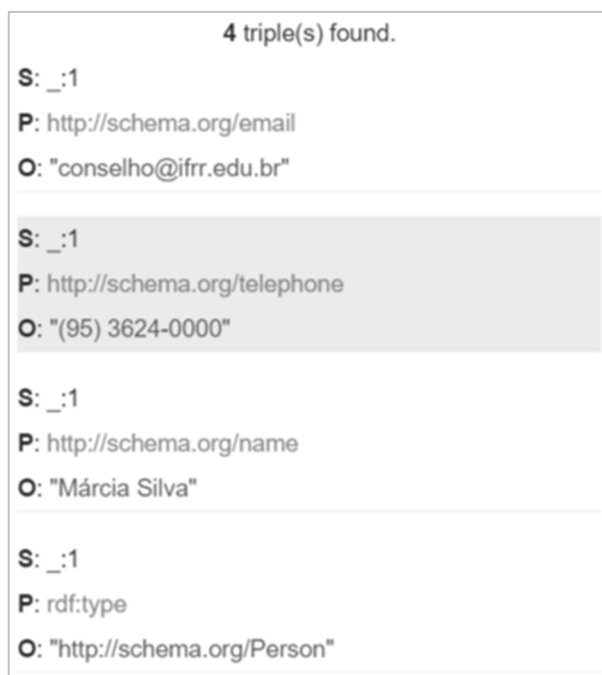
As funcionalidades dos quatro botões vistos na parte inferior do painel de ferramentas do componente WYSIWYM são descritas na Tabela 5.

Tabela 5 – Descrição das ferramentas do RDFaCE

Ícone	Nome	Descrição
	<i>RDFaCE About</i>	Janela que busca informações sobre os desenvolvedores e licenciamento do RDFaCE
	<i>Automatic Content Annotation</i>	Anotação automática de conteúdo, utiliza das NLP API's para sugerir entidades.
	Fact Browser	Navegador de Fatos identificados no texto.
	<i>Settings</i>	Configuração das NLP API's que serão consultadas e quais as Entidades Padrão, para torná-las mais acessíveis.

Fonte: Elaborado pelo autor.

Quando o usuário clicar no botão *Fact Browser*, uma janela modal abre contendo informações sobre a quantidade de triplas marcadas e a listagem das mesmas, a Figura 28 apresenta a tela do *Fact Browser* com as informações da Márcia anotadas por ela.

Figura 28 - *Fact Browser*

Fonte: Elaborado pelo autor.

O resultado pode ser visualizado no código HTML com anotação RDFa gerado, observe na Figura 29 os atributos *typeof* e *property* além é claro do alinhamento dos elementos.

Figura 29 - Código HTML5/RDFa gerado pelo componente WYSIWYM

```
<div id="content-core">
  <div class="" id="parent-fieldname-text-aa3f486a77e347ee988e8817e288f1b2">
    <div id="namespaces">
    <div class="visualClear"></div>
    <div class="r_entity r_person" typeof="schema:Person">
    <div class="r_prop r_jobtitle" property="schema:jobTitle">
      Secretaria do Conselho Superior</div>
    <div class="visualClear">Secretária: <strong><span>
      <span class="r_prop r_name" property="schema:name">Márcia Silva</span>
      </span></strong></div>
    <div class="visualClear">Telefone:
      <span class="r_prop r_telephone" property="schema:telephone">(95) 3624-0000</span></div>
    <div class="visualClear">e-mail:
      <span class="r_prop r_email" property="schema:email">conselho@ifrr.edu.br</span></div>
  </div>
```

Fonte: Elaborado pelo autor.

6.2.2 Cenário 2 – Extração dos Dados

Este cenário pretende demonstrar a configuração do componente *Script Metadata Extractor* proposto como componente do modelo Eduardo para extrair os

metadados das páginas veiculadas no site do IFRR e em seguida submetê-los ao Fuseki server como alternativa proposta para automatização do processo.

Descrição do cenário:

Diogo é analista de TI do IFRR, atua na área de infraestrutura e lhe foi solicitado a instalação de um servidor para disponibilizar o serviço de *triple storage*. O software selecionado foi o Fuseki server e a plataforma adotada foi o Gnu Linux, mais especificamente a distribuição Debian 8 de 64 bits, em um ambiente virtualizado. Como a maioria das distribuições do Gnu Linux já oferece a linguagem python pré-instalada, após instalar o Fuseki server o Diogo tem apenas que baixar o *Script Metadata Extractor* do sistema de controle de versão para a pasta do Fuseki server e em seguida o executar para testar a conectividade com a rede e a internet. Em seguida ele pode optar por executar o script apenas quando lhe for solicitado ou configurar o mesmo no agendador de tarefas do sistema para que seja executado periodicamente.

Detalhes do cenário

Ao executar o script ele acessa através da biblioteca urllib, as páginas que estão cadastradas em seu dicionário de dados uma a uma, a resposta fica gravada em memória, para cada página o script utiliza da biblioteca RDFlib para extrair os metadados e gravá-los em arquivo, o próximo passo é utilizar os scripts disponíveis no fuseki server para submeter os arquivos a sua base de dados e assim disponibilizá-los online.

O Resultado do script é o código estruturado extraído do HTML na Figura 30 é apresentado o código Turtle gerado pelo *Script Metadata Extractor*.

Figura 30 – Código Turtle extraído em teste de validação no RDFaPlay

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix schema: <http://schema.org/> .

_:1
  rdf:type schema:Person;
  schema:jobTitle "Secretaria do Conselho Superior";
  schema:name "Márcia Silva";
  schema:telephone "(95) 3624-0000";
  schema:email "conselho@ifrr.edu.br" .
```

Fonte: Elaborado pelo autor.

6.2.3 Cenário 3 – Expansão da Ontologia e SPARQL *endpoint*

Este cenário pretende demonstrar os procedimentos executados pelo usuário *Analyst*. A expansão da ontologia utilizada pelo modelo Eduardo e a criação de consultas são atividades que têm o propósito de ampliar o diálogo do Agente Conversacional.

Descrição do cenário:

Tiago é Analista de TI do IFRR e foi encarregado de cumprir com a responsabilidade de manutenção da ontologia e criação das consultas SPARQL na implantação do protótipo do modelo Eduardo. Ele possui bom conhecimento das tecnologias de web semântica para realizar essas duas atividades.

Para a primeira atividade o Tiago deve manter os arquivos que ampliam o vocabulário Schema.org. Caso ele queira projetar uma nova ontologia para um domínio específico do site, ele pode usar o Protégé, porém se quiser continuar hospedando a ontologia em sua instância do Schema.org deverá convertê-la para RDFa ou JSON-LD, sendo que até o momento esse procedimento não é suportado pelo Protégé então deve ser realizado manualmente. A dificuldade deste procedimento vai depender do grau de complexidade da ontologia.

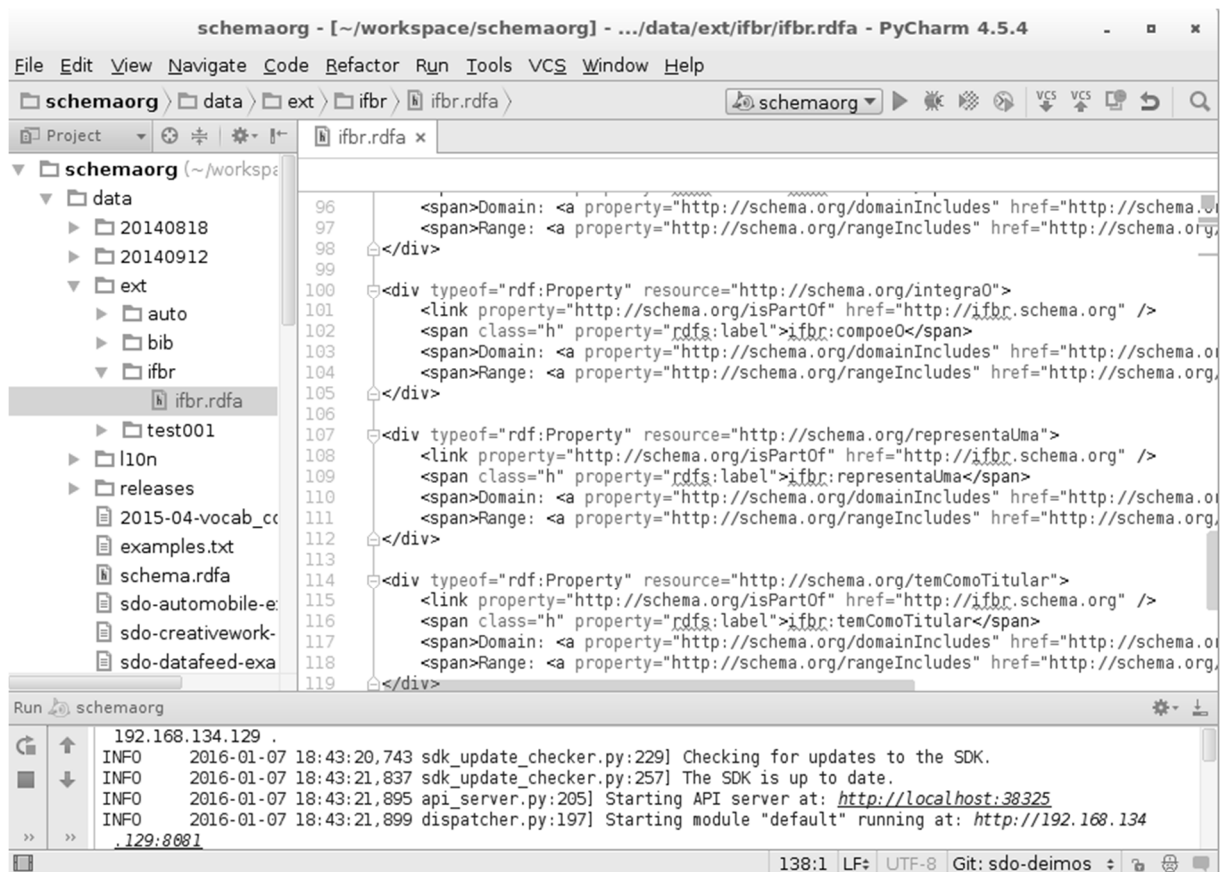
Para criar, testar e executar novas consultas SPARQL que lhe tenham sido solicitadas para o agente ou qualquer outro propósito, o Tiago pode utilizar a interface de Sparql endpoint do que o fuseki server disponibiliza para este fim.

Detalhes do cenário

O ambiente onde o Tiago poderá editar a ontologia é basicamente qualquer IDE ou editor de texto de sua preferência, tendo sido escolhido o uso do PyCharm Community por se tratar de um aplicação em Python cujo *deploy* pode ser feito no Google appengine e esta ferramenta apresenta algumas facilidades para se desenvolver com essas tecnologias. O mais importante é o código para extensão e implantação de instância do Schema.org. Na Figura 31 uma parte da estrutura de

diretórios do projeto Schema.org pode ser visto em `schemaorg->ext->ifbr` como um exemplo de extensão do seu vocabulário.

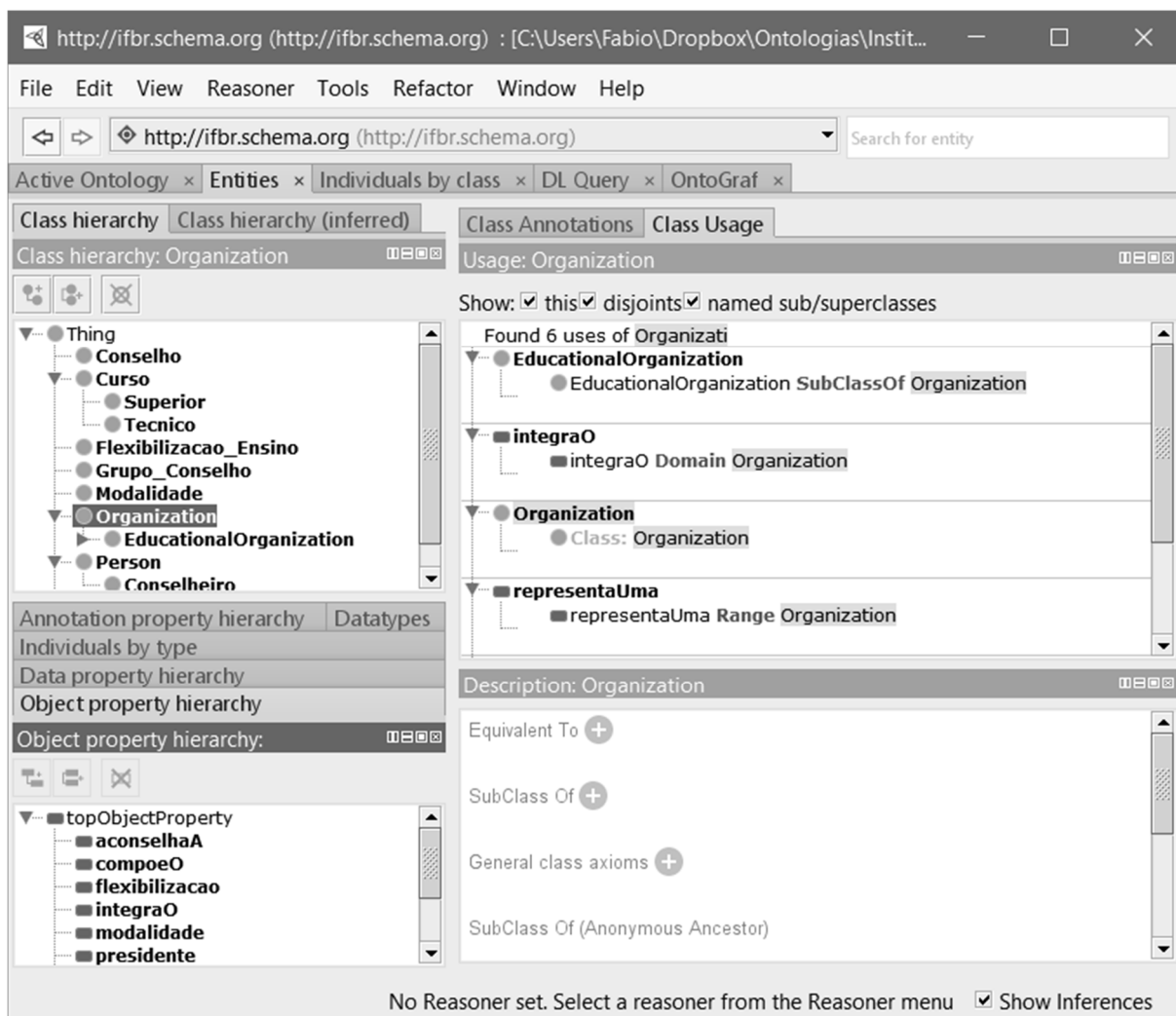
Figura 31 - Ambiente proposto para extensão da ontologia Schema.org



Fonte: Elaborado pelo autor.

Para obter uma visão conceitual da ontologia, realizar testes e propor inferências, ou até mesmo criar uma ontologia nova para qualquer domínio de aplicação que desejar, Tiago pode fazer uso do Protégé, sendo que esta ferramenta pode ser visualizada na Figura 32 juntamente com algumas entidades e propriedades utilizadas no modelo protótipo do modelo Eduardo.

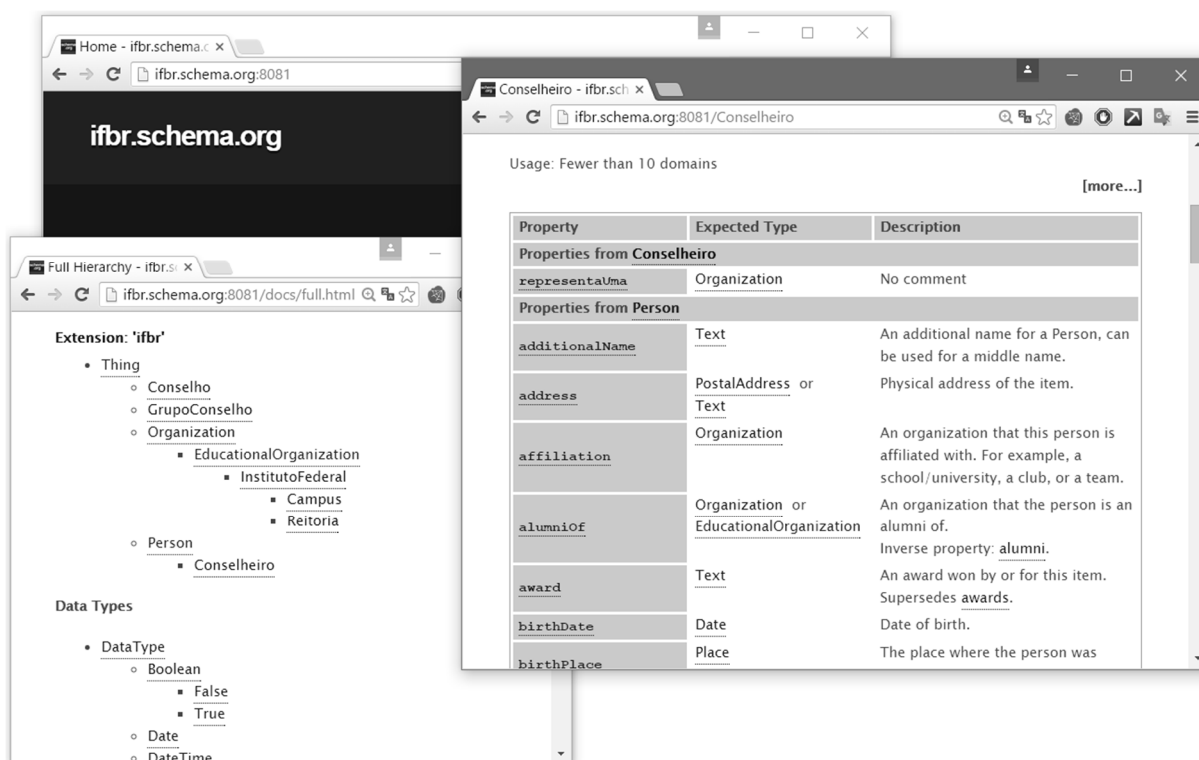
Figura 32 - Protégé contendo algumas entidades e propriedades utilizadas no protótipo



Fonte: Elaborado pelo autor.

O resultado das atividades do Tiago pode ser visualizado na Figura 35 que contem um figura da extensão da ontologia Schema.org.

Figura 33 – Extensão da Ontologia Schema.org



Fonte: Elaborado pelo autor.

6.2.4 Cenário 4 – Conversando com o Agente

O objetivo desse cenário é demonstrar a expansão do diálogo do agente conforme os usuários forem interagindo com ele.

Descrição do cenário:

Maria é aluna do IFRR e acessa o site para perguntar ao Agente sobre o termo “consup” do qual tem ouvido falar pelos corredores da instituição. Então ela inicia uma conversa com o agente em busca de respostas aos seus questionamentos.

O diálogo da Maria com o Agente pode ser visualizado na Figura 34, ela realiza algumas perguntas simples como uma saudação “Oi” e uma pergunta sobre o agente “Quem é você?”, quando ela pergunta “O que é o consup?”, o agente responde “Consup ou Conselho Superior é o órgão de deliberação máxima do IFRR.”, estas respostas estão representadas em AIML e o agente não realizou nenhuma consulta sparql para formulá-las. Porém quando a Maria pergunta “Quem são os membros do consup?” o

agente não consegue responder pois não possui a resposta corresponde representada em AIML, então apresenta uma resposta padrão para quando não tem em sua base de conhecimento a resposta relacionada a pergunta do Usuário.

Figura 34 - Primeira conversa com o Agente

Você: Oi
Edu: Olá! Seja bem vindo! Em que posso ajudar?
Você: Quem é você?
Edu: Eu me chamo Eduardo e sou um agente conversacional inteligente, mas você pode me chamar de Edu, em que posso ajudar?
Você: O que é o consup?
Edu: Consup ou Conselho Superior é órgão de deliberação máxima do IFRR.
Você: Quem são os membros do consup?
Edu: Desculpe não entendi o que disse, talvez eu não saiba do que se trata, mas vou pesquisar e tentar responde-la em outra oportunidade.
Você: Ok
Edu: Ok! Posso ajudar respondendo a outras perguntas?

Fonte: Elaborado pelo autor.

No tópico seguinte é apresentada a solução desta solução, com a expansão e manutenção dos diálogos do agente que o ajudam a responder esse tipo de pergunta.

Detalhes do cenário

Tiago que o analista responsável pelos diálogos do agente, lê os logs de perguntas realizadas ao protótipo do modelo Eduardo e identifica que a pergunta da Maria não foi atendida, então Tiago amplia a ontologia Schema.org para oferecer as entidades, propriedades e relações necessárias para anotar as informações dos membros do consup (este procedimento já foi detalhado no tópico 6.2.3). Para isso ele utiliza a abordagem *scrape* para extrair o JSON do Schema.org com o *schema* ampliado e disponibiliza para o componente *WYSIWYM*. Em seguida ele solicita a Márcia que é a secretária do consup, que anote a página do consup que dá publicidade aos seus membros. Por fim Tiago cria a consulta Sparql necessária para que o Agente consiga responder a pergunta de Maria.

Na Figura 35 é apresentado um trecho da utilização da interface de chat do Eduardo onde este cenário foi testado para consultar informações de contato de servidores disponíveis na página de servidores e que estão ligadas a página do consup.

Figura 35 - Interface de Chat do Eduardo



Fonte: Elaborado pelo autor.

No caso da pergunta “Quais os contatos dos representantes dos docentes no consup?”, o agente ainda não suporta perguntas com sujeitos ocultos ou implícitos como por exemplo “Quais os contatos dos docentes?” ou no caso da pergunta “Quem é o presidente do consup?” ele não aceitaria a pergunta “Quem é o presidente?”, pois o protótipo foi desenvolvido apenas para atender a lacuna da automatização da atualização dos diálogos e não o gerenciamentos das conversas, isso será implementado em um trabalho futuro para não comprometer o foco da pesquisa.

6.3 Avaliação de Aceitabilidade

Para a avaliação de aceitabilidade foi utilizado o Modelo de Aceitação de Tecnologia (MAT) proposto por Davis (1989) e ampliado por Yoon e KIM (2007), usando uma escala de Likert (1932).

A avaliação envolveu 8 voluntários, que avaliaram o protótipo com o objetivo de investigar a percepção de utilidade e facilidade de uso do modelo Eduardo. Os participantes receberam um tutorial com o passo a passo das atividades que deveriam realizar para anotar o conteúdo. As atividades deste tutorial foram realizadas na forma presencial, as informações de contato pessoal deveriam ser identificadas em um texto, a entidade *Person* do Schema.org, e propriedades como *name*, *email*, *telephone* e *jobTitle* eram marcadas com o auxílio do componente *WYSIWYM*. Eles também foram instruídos sobre conceitos básicos de metadados e também foram informados sobre o objetivo do modelo Eduardo. Em seguida, os voluntários foram convidados a usar o protótipo, executando as seguintes tarefas que foram previamente demonstradas: (i) anotar o conteúdo com o componente *WYSIWYM* (uma página de contato contendo informações de pessoas, organizações e suas relações) e (ii) conversar com o Agente. Depois de testar o protótipo, estes voluntários responderam a um questionário contendo 10 questões.

Os participantes do experimento foram selecionadas por estarem ligados, direta ou indiretamente, no processo de publicação de conteúdo no portal do IFRR. A faixa etária varia entre 25 e 60 anos. As profissões dos entrevistados são predominantemente das áreas de tecnologia da informação, administração, comunicação e docência. Deve ser reforçado que os usuários já publicavam no site e utilizavam o TinyMCE, sendo que eles apenas não haviam trabalhado com o componente de anotação RDFaCE.

A Tabela 6 apresenta a síntese das respostas em relação à percepção de facilidade de uso. A primeira coluna da tabela apresenta a declaração colocada. As seguintes cinco colunas mostram os resultados obtidos em cada escala, em graus absolutos e relativos.

Analisando os resultados, podemos ver que os usuários concordam que a interface *WYWISYM* facilita marcação dos dados semânticos (declaração 1), porém a quantidade de entidades deve ser observada (declaração 2), ainda com relação a

necessidade de anotação dos dados, os entrevistados não questionaram que tal atividade compromete sua rotina de trabalho diária (declaração 3). Com relação a interação com o agente conversacional, os entrevistados ficaram parcialmente satisfeitos, sendo que isso pode estar relacionado ao fato da base de conhecimento do agente ainda ser pequena, afinal ele ainda não está em plena utilização (declaração 4), contudo um percentual maior concordou que o agente facilitou a sua busca por informações no site.

Tabela 6 – Resultado da Avaliação para Facilidade de Uso

Declaração / Resultados	Discordo totalmente	Discordo parcialmente	Não concordo nem discordo	Concordo parcialmente	Concordo totalmente
1. Foi fácil marcar as entidades, definir as relações e as propriedades	0% (0)	0% (0)	0% (0)	37,5% (3)	62,5% (5)
2. As entidades disponíveis para anotação foram suficientes	0% (0)	0% (0)	0% (0)	62,5% (5)	37,5% (3)
3. A necessidade de anotar dos dados não chegou a comprometer as suas atividades de publicação	0% (0)	0% (0)	12,5% (1)	25% (2)	62,5% (5)
4. A experiência ao conversar com o agente foi satisfatória	0% (0)	0% (0)	0% (0)	62,5% (5)	37,5% (3)
5. O Agente facilitou a sua busca por informações	0% (0)	0% (0)	0% (0)	50% (4)	50% (4)
Média	0%	0%	2,5%	47,5%	50%

Fonte: Elaborado pelo autor.

As declarações relacionadas com a percepção de utilidade são apresentados na Tabela 7.

Tabela 7 – Resultado da Avaliação para Percepção de Utilidade dos Usuários

Declaração / Resultados	Discordo totalmente	Discordo parcialmente	Não concordo nem discordo	Concordo parcialmente	Concordo totalmente
6. É útil a distribuição de metadados	0% (0)	0% (0)	0% (0)	0% (0)	100% (8)
7. A componente de edição o ajudou anotar o conteúdo	0% (0)	0% (0)	12,5% (1)	37,5% (3)	50% (4)
8. O agente esclareceu as suas dúvidas	0% (0)	0% (0)	0% (0)	87,5% (7)	12,5% (1)
9. A resposta do agente foi útil pra você	0% (0)	0% (0)	0% (0)	12,5% (1)	87,5% (7)
10. Você utilizaria o Agente novamente	0% (0)	0% (0)	0% (0)	12,5% (1)	87,5% (7)
Média	0%	0%	2,5%	30%	67,5%

Fonte: Elaborado pelo autor.

Analisando os resultados da tabela 7, observa-se que os usuários concordam totalmente com a utilidade de se distribuir metadados no contexto do portal do IFRR (declaração 6). Com relação ao componente WYSIWYM observa-se que algumas melhorias em sua interface são necessárias em virtude da não conformidade nas respostas (declaração 7). A opinião dos entrevistados a respeito da declaração 8 diverge pouco, sendo o mesmo observado em resposta às declarações 9, onde os entrevistados consideram a resposta do agente útil, e 10, onde eles demonstram interesse em utilizar o agente novamente. Apesar das declarações estarem interligadas de certa forma, considera-se que melhorias no gerenciamento de diálogo do agente deveriam ser implementadas tanto no modelo quanto no protótipo.

A visão geral desse experimento foi revelada através da Figura 36 levando também em consideração os comentários levantados pelos participantes durante os testes. Os resultados em relação a facilidade de uso se mostraram positivos e revelam o potencial do modelo no apoio a transparência na disponibilização dos dados da instituição e também na orientação a navegação dos usuários que acessam o portal do IFRR, já que seus os principais interessadas consideraram os componentes do protótipo fáceis de se manusear. Ainda na Figura 36, no gráfico que se refere a percepção de utilidade dos usuários, apresenta forte identificação de utilidade no fato de se divulgar os metadados para apoiar as buscas nos mesmos.

Figura 36 - Resumo da Avaliação de Aceitabilidade



Fonte: Elaborado pelo autor.

O experimento foi realizado apenas uma vez com um número reduzido de pessoas e ações, o que poderia tornar o resultado tendencioso caso o viés da

pesquisa fosse de natureza quantitativa. Como se trata de um experimento de natureza qualitativa e universo de apenas 15 pessoas, o número de 8 participantes é considerado expressivo por tratar-se de metade dos envolvidos no processo publicação.

7 CONCLUSÃO

Este trabalho apresentou um modelo para integração automática de conteúdo anotado semanticamente com um agente conversacional. A descrição deste modelo proporcionou a implementação de um protótipo, cujo objetivo foi a publicação de conteúdo anotado semanticamente para websites e o consumo deste conteúdo por um agente conversacional. Este protótipo foi implementado tendo como domínio um caso de divulgação de informações institucionais e foram realizadas avaliações que apresentaram resultados promissores.

Para elaboração do modelo proposto foram avaliados trabalhos relacionados tendo como base os principais temas do modelo. Estes trabalhos estudados foram resumidos e analisados no capítulo de trabalhos relacionados e auxiliaram na identificação de possibilidades de pesquisas e também na identificação de soluções já existentes nesta área (SERÓN; BOBED, 2014), (KIMURA; KITAMURA, 2006), (DE GASPERIS, 2010), (KHALILI; AUER; HLADKY, 2012), (PRATES, 2014). Dentre as características do modelo proposto, podem ser citadas o fato de ser um modelo aplicável à um domínio genérico, por possuir o suporte a um editor *WYSIWYM* e permitir a representação de dados estruturados em RDFa, Microformatos e RDF/XML, além da capacidade de extrair os metadados semânticos de páginas HTML5 e contar com a extensão de um interpretador AIML para realizar as consultas SPARQL. Estes elementos foram definidos com base nas lacunas observadas e com o objetivo de possibilitar a utilização flexível do modelo.

Tendo em vista a questão de pesquisa definida, foram tomadas decisões de projeto ao longo da definição do modelo e foram realizados procedimentos que buscaram evidenciar como o uso de recursos da Web Semântica pode apoiar a melhor utilização de conteúdo web, a partir de procedimentos de anotação semântica tanto automáticos como manual. Os resultados dos conteúdos anotados representaram a possibilidade de uma integração com o contexto de um agente conversacional com base na utilização do AIML, o que permite superar uma das limitações conhecidas no caso destes sistemas, que é a dificuldade de geração de conteúdo que sirva de base para os diálogos. O processo automatizado de geração de diálogos em formato AIML e os mecanismos de integração com os conteúdos

anotados através de consultas SPARQL apoiam a funcionalidade e aceitação de agentes conversacionais por parte dos usuários.

Uma das contribuições a destacar em relação ao modelo apresentado é a possibilidade de ampliação automatizada do repertório do agente conversacional conforme novos conteúdos são publicados nos sites vinculados a ele. Este é um fator importante para apoiar a consolidação da utilização de recursos da web semântica para a anotação das informações de modo automático e disponibilização destas para uso por componentes como os agentes conversacionais. A melhoria na aceitabilidade percebida por usuários em ambientes de interação por linguagem natural é dependente da qualidade dos recursos que os sistemas conversacionais possuem à sua disposição. Deste modo, esta contribuição observada se localiza como mais um elemento de melhoria na interface de interação com os usuários.

Os resultados das avaliações se mostraram promissores ao usar um agente conversacional com base em web semântica como elemento de interface com usuários em web sites, pois foi possível avaliar os conceitos presentes no modelo, bem como observar a indicação obtida de facilidade de uso e utilidade perceptível.

7.1 Limitações e Trabalhos Futuros

Nesse trabalho, procurou-se tratar o mapeamento do conteúdo considerando a língua portuguesa, e uma das funcionalidades do componente *WYSIWYM* está relacionada ao processamento de linguagem natural. Segundo (PRATES, 2014), estudos da língua portuguesa ainda estão muito aquém do desejado e compõem assim uma limitação a superar. Em trabalhos futuros espera-se implementar as funcionalidades de (i) memória de trabalho e sensibilidade de contexto, (ii) o reconhecimento dos usuários por algoritmos de reconhecimento facial para incorporar estes elementos como aspectos do diálogo, (iii) melhoria na interface do *WYSIWYM* e (iv) atualização na forma como o *WYSIWYM* consome o vocabulário Schema.org para que suporte o formato JSON-LD divulgado oficialmente.

REFERÊNCIAS

- ALCHEMYAPI. **Getting Started with AlchemyAPI**. Disponível em: <<http://www.alchemyapi.com/developers/getting-started-guide>>. Acesso em: 9 maio. 2015.
- ALICE AI FOUNDATION. **AIML - The Artificial Intelligence Markup Language**. Disponível em: <<http://www.alicebot.org/aiml.html>>. Acesso em: 2 maio. 2015.
- BIRD, S.; KLEIN, E.; LOPER, E. **Natural language processing with Python**. 1st ed ed. Beijing ; Cambridge [Mass.]: O'Reilly, 2009.
- BRASIL. **LAI: A Lei de Acesso à Informação**. Governamental. Disponível em: <<http://www.acessoainformacao.gov.br/assuntos/conheca-seu-direito/a-lei-de-acesso-a-informacao/a-lei-de-acesso-a-informacao>>. Acesso em: 29 dez. 2015.
- BUSH, N. **AIML 1.0.1 (A.L.I.C.E. AI Foundation)**. Disponível em: <<http://www.alicebot.org/TR/2011/>>. Acesso em: 2 maio. 2015.
- COLLINS, MICHAEL. **Processamento da Linguagem Natural**. E-lern. Disponível em: <<https://www.coursera.org/course/nlangp>>. Acesso em: 18 fev. 2015.
- COSTA, H. **Um Modelo de Arquitetura para o Turismo Ubíquo Utilizando Dispositivos Móveis**. São Leopoldo: UNISINOS, 2013.
- CYGANIAK, R.; HAUSENBLAS, M. **What is Schema.RDFS.org?** Disponível em: <<http://schema.rdfs.org/>>. Acesso em: 5 jan. 2016.
- DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS quarterly**, p. 319–340, 1989.
- DE GASPERIS, G. Building an AIML Chatter Bot Knowledge-Base Starting from a FAQ and a Glossary. **Journal of e-Learning and Knowledge Society**, v. 6, n. 2, 2010.
- DEY, A. K. Understanding and using context. **Personal and ubiquitous computing**, v. 5, n. 1, p. 4–7, 2001.
- DUCHARME, B. **Learning SPARQL: querying and updating with SPARQL 1.1**. 2. ed ed. Beijing: O'Reilly, 2013.

FONSECA, J. **Como Aumentar seu CTR com Schema.org - SEO Master.**

Disponível em: <<http://www.seomaster.com.br/blog/aumente-seu-ctr-com-schema.org>>. Acesso em: 18 jan. 2016.

GIL, A. C. **Como elaborar projetos de pesquisa.** São Paulo: Atlas, 2008.

HARTSON, H. R.; HIX, D. Human-computer interface development: concepts and systems for its management. **ACM Computing Surveys (CSUR)**, v. 21, n. 1, p. 5–92, 1989.

HERMAN, I.; BIRBECK, M.; RIXHAM, N. **RDFa API.** Disponível em: <<http://www.w3.org/TR/rdfa-api/>>. Acesso em: 2 maio. 2015.

HUSAIN, K. S. Performance Evaluation Process of the Content Management System: Case Study of Kau Marz System. **Middle-East Journal of Scientific Research**, v. 11, n. 1, p. 117–127, 2012.

KHALILI, A.; AUER, S.; HLADKY, D. **The RDFa Content Editor - From WYSIWYG to WYSIWYM.** IEEE, jul. 2012Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6340208>>. Acesso em: 27 abr. 2015

KIMURA, M.; KITAMURA, Y. Embodied conversational agent based on semantic web. In: **Agent Computing and Multi-Agent Systems.** [s.l.] Springer, 2006. p. 734–741.

KNÖPFEL, A. **FMC Quick Introduction**FMC Consortium, , jun. 2007.

KRUCHTEN, P. **The rational unified process: an introduction.** [s.l.] Addison-Wesley Professional, 2004.

LIKERT, R. A Technique for the Measurement of Attitudes. **Archives of Psychology**, v. 22, n. 140, p. 1–55, 1932.

LOPER, E.; BIRD, S. **NLTK: The natural language toolkit.** Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1. **Anais...**Association for Computational Linguistics, 2002Disponível em: <<http://dl.acm.org/citation.cfm?id=1118117>>. Acesso em: 26 mar. 2016

LUN, E. VAN. **Conversational Agent - a program that acts on behalf of humans and chats**. Disponível em: <https://www.chatbots.org/conversational_agent/>.

Acesso em: 18 fev. 2015.

MAASS, W.; KOWATSCH, T. (EDS.). **Semantic Technologies in Content Management Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

MARCOS. **Como schema pode te ajudar em SEO**. Disponível em:

<<http://tableless.com.br/como-schema-pode-te-ajudar-em-seo/>>. Acesso em: 18 jan. 2016.

MAULDIN, M. L. ChatterBots, TinyMuds, and the Turing Test Entering the Loebner Prize Competition. p. 6, 1994.

MCTEAR, M. F. **Spoken Dialogue Technology**. London: Springer London, 2004.

NETO, A. T. et al. **Developing and evaluating web multimodal interfaces-a case study with usability principles**. Proceedings of the 2009 ACM symposium on Applied Computing. **Anais...ACM**, 2009Disponível em:

<<http://dl.acm.org/citation.cfm?id=1529306>>. Acesso em: 16 mar. 2016

<<http://dl.acm.org/citation.cfm?id=1529306>>. Acesso em: 16 mar. 2016

PEREZ-MARIN, D.; PASCUAL-NIETO, I. (EDS.). **Conversational agents and natural language interaction: techniques and effective practices**. Hershey, PA: Information Science Reference, 2011.

PERKINS, J. **Python 3 text processing with NLTK 3 cookbook: over 80 practical recipes on natural language processing techniques using Python's NLTK 3.0**.

2. ed ed. Birmingham: Packt Publ, 2014.

PRATES, J. S. **Gerenciamento de Diálogo Baseado em Modelo Cognitivo para Sistemas de Interação Multimodal**. Mestrado—São Leopoldo: UNISINOS, 2014.

RDF WORKING GROUP. **RDF - Semantic Web Standards**. Disponível em:

<<http://www.w3.org/RDF/>>. Acesso em: 18 fev. 2015.

RIGO, S. J. **Integração de recursos da web semântica e mineração de uso para personalização de sites**. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2008.

SAHA, A.; SETUA, S. K. **Architectural model for next generation content management system**. Electrical and Computer Engineering (ICECE), 2014

International Conference on. **Anais...IEEE**, 2014Disponível em:
<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7026859>. Acesso em: 18 fev. 2015

SATYANARAYANAN, M. Pervasive Computing: Vision and Challenges. **Personal Communications, IEEE**, v. 8, n. 4, p. 10–17, 2001.

SCHEMA.ORG. **Extending Schemas**. Disponível em:
<<http://schema.org/docs/extension.html>>. Acesso em: 5 jan. 2016.

SECOM. **Manual de Diretrizes: Identidade padrão de comunicação digital do poder executivo federal**, dez. 2014.

SERÓN, F. J.; BOBED, C. VOX system: a semantic embodied conversational agent exploiting linked data. **Multimedia Tools and Applications**, 9 out. 2014.

STRATTON, C. **PyAIML (a.k.a. Program Y) - A Python AIML Interpreter**. Disponível em: <<http://pyaiml.sourceforge.net/>>. Acesso em: 2 maio. 2015.

UNGER, R.; CHANDLER, C. **O Guia Para Projetar UX: A Experiência do Usuário (UX) para Projetistas de Conteúdo Digital, Aplicações e Web Sites**. Rio de Janeiro: Alta Books, 2009.

WALKER, J. Through the looking glass. In: LAUREL, B. (Ed.). . **The art of human-computer interface design**. 1. ed. Massachusetts: Addison-Wesley Publishing, 1990. p. 439–447.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro: Elsevier, 2009.

WEIZENBAUM, J. ELIZA—a computer program for the study of natural language communication between man and machine. **Communications of the ACM**, v. 9, n. 1, p. 36–45, 1966.

WRIGHT, A. New Search Technologies Mine the Web More Deeply. **The New York Times**, 22 fev. 2009.

YOON, C.; KIM, S. Convenience and TAM in a ubiquitous computing environment: The case of wireless LAN. **Electronic Commerce Research and Applications**, v. 6, n. 1, p. 102–112, mar. 2007.

YU, L. **A Developer's Guide to the Semantic Web**. Berlin, Heidelberg: Springer
Berlin Heidelberg, 2014.

ANEXO A DADOS RECEBIDOS DOS PARTICIPANTES PARA ELABORAR OS EXPERIMENTOS

Pergunta	Respostas esperadas Respostas
1 Qual é o nome do Reitor?	O nome dele é Ademar de Araújo Filho
2 Qual o e-mail do Reitor?	O e-mail dele é ademar@ifrr.edu.br
3 Onde encontro os cursos ofertados pelo IFRR?	<ul style="list-style-type: none"> • Informar a url ao usuário para uma página que contém a relação de curso do campus ou • Perguntar se tem interesse em algum campus, se sim, filtrar os cursos deste campus e informar ao usuário.
4 Onde encontro as portarias dos servidores?	Informar o endereço da página que contém as portarias da DGP.
5 Como posso consultar um processo?	<ul style="list-style-type: none"> • Informar que a consulta pública é feita através do sistema unificado de administração pública (SUAP) e informar o link para o usuário acessar ou • Perguntar se é um servidor do IFRR e informar que ele pode acessar o SUAP para filtrar os processos
6 Como posso agendar uma visita técnica no IFRR?	<ul style="list-style-type: none"> • Perguntar sobre qual área técnica ele tem interesse e em seguida informar os dados de contato do coordenador do curso.
7 Quando serão abertas as inscrições do vestibular?	<ul style="list-style-type: none"> • No momento não temos nenhum vestibular em aberto ou • As inscrições iniciam dia DD/MM/AAAA e finalizam no dia DD/MM/AAAA e os cursos ofertados são: (listar os curso) • As inscrições já iniciaram e finalizam no dia DD/MM/AAAA
8 Onde fica localizado o campus Amajari?	O Campus Amajari está localizado na Vicinal de acesso que liga a balsa de Aparecida à Vila Brasil, Km 03, Amajari – RR / CEP 69343-000
9 Onde encontro minhas Portarias?	Você poderá encontrar todas as portarias nesse link: http://www.ifrr.edu.br/portarias
10 O que é o Consup?	É o órgão de deliberação máxima do IFRR.
11 Quem são os membros do Consup?	A comissão é formada pelos seguintes membros: (listar os membros)
12 Quando será realizada a próxima reunião do Consup?	Ainda não foi publicada no calendário.
13 Quem é o presidente do Consup?	O Reitor.

ANEXO B QUESTIONÁRIO PARA A AVALIAÇÃO DE USABILIDADE

Preencha o questionário abaixo com a sua opinião sobre as afirmativas a respeito o Protótipo Eduardo.

Sobre a facilidade de uso

1. Foi fácil marcar as entidades, definir as relações e as propriedades				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente
2. As entidades disponíveis para anotação foram suficientes				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente
3. A necessidade de anotar os dados não chega a comprometer as suas atividades				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente
4. A experiência ao conversar com o agente foi satisfatória				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente
5. O Agente facilitou a sua busca por informações				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente

Sobre a sua utilidade

6. É útil a distribuição de metadados				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente
7. O componente de edição o ajudou anotar o conteúdo				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente
8. O agente esclareceu as suas dúvidas				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente
9. A resposta do agente foi útil pra você				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente
10. Você utilizaria o Agente novamente				
<input type="checkbox"/> Discordo totalmente	<input type="checkbox"/> Discordo parcialmente	<input type="checkbox"/> Não concordo nem discordo	<input type="checkbox"/> Concordo parcialmente	<input type="checkbox"/> Concordo totalmente

