Bruno Cesar Neves de Oliveira

Uma Abordagem Automática para Enriquecimento Semântico de Serviços de Dados Web

Florianópolis
2018

Bruno Cesar Neves de Oliveira

UMA ABORDAGEM AUTOMÁTICA PARA ENRIQUECIMENTO SEMÂNTICO DE SERVIÇOS DE DADOS WEB

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do Grau de Mestre em Ciência da Computação.
Orientador: Prof. Dr. Frank Augusto Siqueira

Florianópolis
2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Oliveira, Bruno Cesar Neves de
Uma Abordagem Automática para Enriquecimento Semântico de Serviços de Dados Web / Bruno Cesar Neves de Oliveira ; orientador, Frank Augusto Siqueira, 2018.
143 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Ciência da Computação, Florianópolis, 2018.

Inclui referências.

1. Ciência da Computação. 2. Web Services. 3. Web Semântica. 4. Construção e Matching de Ontologias. 5. Dados Conectados. I. Siqueira, Frank Augusto. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Bruno Cesar Neves de Oliveira

UMA ABORDAGEM AUTOMÁTICA PARA ENRIQUECIMENTO SEMÂNTICO DE SERVIÇOS DE DADOS WEB

Esta dissertação foi julgada adequada para obtenção do título de mestre e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis, ____de _____ de _____.

_____
Prof. José Luís Almada Güntzel, Dr.
Coordenador do Programa

Banca Examinadora:

_____
Prof. Frank Augusto Siqueira, Dr.
Universidade Federal de Santa Catarina
Orientador

_____
Prof. Alexandre Leopoldo Gonçalves, Dr.
Universidade Federal de Santa Catarina (Videoconferência)

_____
Prof. Renato Fileto, Dr.
Universidade Federal de Santa Catarina

_____
Prof. Roberto Willrich, Dr.
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Embora uma dissertação seja um trabalho individual, o resultado textual é fruto de muita persistência, esforço, desgaste e, sobretudo, de muitas descobertas – fatores que não são vividos individualmente. Posto que muitos colaboraram de alguma forma para a existência deste trabalho, redigir um agradecimento fiel é uma tarefa que exige cuidado. Tentarei aqui, portanto, ser o mais breve e justo.

Inicio agradecendo aos meus pais pela vida, pelo amor e pela valorização da minha educação. À minha esposa Vívian pela força emanada nos momentos adversos, pelo companheirismo e pela compreensão das minhas ausências e das noites em claro. À minha irmã Mariana, tio(a)s Hugo, Jussara, Isabel, avós Lourdes e Waldete, e avô Everaldo, agradeço pela sabedoria e incentivos imensuráveis. Penso que a presença da família é essencial para a sanidade de um ser. Portanto, agradeço a todos os demais familiares e amigos que, mesmo tendo dificuldades de captar a razão para tanto estudo, colaboraram com a minha mudança e apoiaram a decisão do mestrado.

Agradeço aos professores pelos ensinamentos e conhecimentos transmitidos e, em especial, ao orientador Prof. Frank Siqueira pelo seu tempo valioso, suas oportunas colaborações e correções extremamente importantes. Muito mais que isso, devo agradecê-lo pela confiança depositada e pelo aceite da orientação mesmo antes da minha mudança para Florianópolis. Ademais, sou grato aos membros da banca examinadora pelas sugestões que certamente contribuíram para a melhoria deste trabalho.

Aos colegas de mestrado e, em especial, aos colegas do Laboratório de Pesquisa em Sistemas Distribuídos que acompanharam os resultados do projeto, deixo aqui o meu sincero apreço: Alyson, Camilo, Eliza e Pedro. Sou imensamente grato a Alexis Huf e Ivan Salvadori pelo acolhimento, pelas ideias e dispendiosas discussões que desanuviaram muitas questões e embrionaram esta dissertação. Suas críticas e colaborações foram fundamentais para aprimorar este projeto e mantê-lo vivo durante todo esse percurso.

Por fim, a CAPES, pela concessão da bolsa, a coordenação do PPGCC-UFSC, especialmente Katiana, bem como os revisores dos artigos e os pesquisadores que eventualmente colaboraram com este trabalho merecem o devido reconhecimento. *Kansha Shite.*

Ontogeny: the history of structural
changes that a system experiences
without losing its identity.
(Mehdi Khosrow-Pour)

RESUMO

A disponibilização de *Web Services* que se destinam a fornecer uma interface de acesso e manipulação a fontes de dados (também conhecidos como serviços de dados) tem aumentado consideravelmente nos últimos anos, mostrando-se uma estratégia prática e eficiente para compartilhamento de dados entre sistemas computacionais. No entanto, em geral, estes serviços não especificam a semântica dos dados, podendo ainda representar os dados em formatos heterogêneos, o que resulta em dificuldades para seu processamento e reúso. Em vista disso, novas abordagens e ferramentas têm surgido com o intuito de auxiliar no enriquecimento semântico de serviços, adicionando significado aos dados geridos e aprimorando processos de composição e descoberta de serviços. Todavia, tais abordagens ainda requerem um esforço humano significativo para construção de ontologias que descrevem os conceitos semânticos utilizados para descrição dos serviços de dados. Além disso, as propostas encontradas na literatura focam, majoritariamente, na geração de descrições semânticas das interfaces dos serviços, dando pouca atenção às representações dos dados por eles fornecidos. Nesse contexto, esta dissertação propõe uma arquitetura, denominada OntoGenesis, capaz de i) construir e evoluir ontologias de domínio para serviços de dados e ii) alinhar as propriedades das ontologias geradas com ontologias de fontes externas, de modo a reutilizar conceitos já existentes e facilitar a integração com outros agentes de *software*. Desse modo, a arquitetura provê mecanismos para enriquecer tanto as descrições quanto as representações fornecidas por serviços de dados, tendo como base os conceitos semânticos da ontologia gerada. Como contribuição, espera-se abstrair, do ponto de vista dos desenvolvedores de *software*, o processo oneroso de enriquecer semanticamente serviços conectados a fontes que armazenam dados sem informação semântica, fomentando, assim, a execução de análises de dados mais sofisticadas. Experimentos utilizando dados abertos nos âmbitos científico e governamental mostram a aplicabilidade da proposta e revelam que são alcançados níveis adequados de conformidade (precisão e cobertura), bem como desempenho superior em relação a abordagens similares encontradas na literatura.

Palavras-chave: *Web Services*. Serviços de Dados. Web Semântica. Construção e *Matching* de Ontologias. Dados Conectados.

ABSTRACT

The availability of Web Services meant to provide an interface to access and manipulate data sources (also called data services) has been sharply increasing in recent years, being a practical and efficient strategy for data sharing among computing systems. However, in general, these services do not specify the semantics of data, and can represent it in heterogeneous formats, which hinders its effective processing and reuse. As a result, many approaches and tools have emerged to support the semantic enrichment of Web services, expliciting semantic to managed data, and enhancing processes such as service composition and discovery. Nevertheless, such approaches still require significant human effort to build ontologies that describe the semantic concepts employed for describing data services. Moreover, most proposals found in the literature focus on the generation of semantic descriptions of the service interface, disregarding the data representations provided by them. In this context, this dissertation proposes an architecture, named OntoGenesis, capable of i) constructing and evolving domain ontologies for data services and ii) matching the properties of such ontologies with ontologies of external sources, in order to reuse existing concepts and ease the integration with other software agents. Thus, the architecture provides mechanisms to enrich both the descriptions and representations provided by data services based on the semantic concepts of the generated ontology. As a contribution, it is expected to minimize, from the viewpoint of software developers, the time-consuming tasks of semantically enriching services attached to sources that store data without semantic information, therefore enabling more sophisticated reasoning tasks. Experiments using open datasets from scientific and governmental domains show the applicability of our proposal and reveal that it can achieve satisfactory levels of compliance (precision and recall), as well as better performance in comparison with similar approaches found in the literature.

Keywords: Web Services. Data Services. Semantic Web. Ontology Construction. Ontology Matching. Linked Data.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AFD	Autômato Finito Determinístico
API	Application Programming Interface
AROMA	Association Rule Ontology Matching Approach
B.O.	Boletim de Ocorrência
FOAF	Friend of a Friend
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IRI	Internationalized Resource Identifier
JAX-RS	Java API for RESTful Web Services
JDK	Java Development Kit
JIT	Just-in-time compilation
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
JVM	Java Virtual Machine
LOD	Linked Open Data
OAEI	Ontology Alignment Evaluation Initiative
OL	Ontology Learning
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Services
P2P	Peer-to-Peer
PARIS	Probabilistic Alignment of Relations, Instances and Schema
RDF	Resource Description Framework
RDFa	Resource Description Framework in Attributes
RDFS	Resource Description Framework Schema
REST	REpresentational State Transfer
RSL	Revisão Sistemática da Literatura
SAWSDL	Semantic Annotations for WSDL
SDS	Serviços de Dados Semânticos
SM	Semantic Mappings (Mapeamentos Semânticos)
SOAP	Simple Object Access Protocol

SOA	Service-oriented architecture
SSP/SP	Secretaria da Segurança Pública do estado de SP
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WSDL	Web Services Description Language
WSMO	Web Service Modeling Ontology
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Este capítulo introdutório contextualiza o problema atacado nesta dissertação e expõe a pergunta de pesquisa, bem como as contribuições do trabalho. Em seguida, são apresentados os objetivos e o método de pesquisa adotado. Ao final, é exposta a organização dos demais capítulos desta dissertação.

1.1 CONTEXTUALIZAÇÃO E PROBLEMA DE PESQUISA

O número de *Web Services*, espalhados em redes públicas e privadas, que se destinam a fornecer dados armazenados em alguma fonte de dados tem aumentado consideravelmente nos últimos anos, mostrando-se uma estratégia prática e eficiente para compartilhamento de dados entre sistemas computacionais. Tais *Web Services*, também chamados de *Web data services* (ou apenas serviços de dados), têm como objetivo fornecer interfaces Web de acesso e manipulação de fontes de dados. O rápido crescimento desses serviços disponibilizados atualmente na Web resultou em uma disseminação de dados representados em formatos heterogêneos e difíceis de serem processados e reutilizados.

Em virtude disso, os padrões e as tecnologias da Web Semântica (BERNERS-LEE; HENDLER; LASSILA, 2001) – como as ontologias, capazes de organizar e relacionar formalmente conceitos de um determinado domínio; o modelo RDF (*Resource Description Framework*), usado para estruturar os dados em triplas; e os dados conectados, que permitem identificar e interligar conjuntos de dados – surgem com o intuito de prover significado aos dados disponíveis na Web, promovendo a sua compreensão não apenas por humanos, mas também por máquinas. Além disso, os serviços de dados tradicionais (sintáticos) podem ser aprimorados com tais tecnologias de modo a prover dados em formatos mais sofisticados e processados por máquinas, além de facilitar a reutilização e a integração de aplicações Web mais complexas. Com a adesão da Web Semântica, então, torna-se possível descrever formalmente não só as relações semânticas entre os serviços e suas capacidades, mas também os dados fornecidos pelos serviços.

Diversos pesquisadores apontam os benefícios de se empregar serviços semânticos, tanto na interoperabilidade dos dados (SALVADORI et al., 2017a), quanto no auxílio para automatizar processos de descoberta, seleção e composição de serviços (MCILRAITH; SON; ZENG, 2001). Além disso, o provimento de dados conecta-

dos e de descrições semânticas em serviços Web promove benefícios importantes em diversas áreas de aplicação. Pode-se citar setores governamentais, áreas de biomedicina, telecomunicações e finanças. No âmbito da segurança pública, por exemplo, um estudo realizado durante esta pesquisa mostrou como dados criminais publicados por meio de serviços de dados semânticos possibilitam análises relevantes para a sociedade (OLIVEIRA et al., 2016).

A implementação e a adoção de serviços de dados semânticos (SDS), todavia, é limitada principalmente pelo formato dos dados armazenados, requerendo que estes estejam descritos semanticamente. Em geral, os dados estão armazenados sintaticamente, isto é, apenas a sua estrutura é especificada, mas não a semântica. Além disso, vários desafios contribuem para essa baixa adoção de SDS. Alguns problemas comuns incluem o tempo e o esforço exigidos na construção de ontologias de domínio e na anotação semântica dos serviços de dados. Essas são tarefas complexas que exigem alto conhecimento do domínio em questão. Adicionalmente, existe ainda o hiato entre construir um serviço com suporte semântico e a possibilidade de explorar um repositório de ontologias abrangendo diversos domínios. Na prática, não é realista assumir que os dados fornecidos pelos serviços serão sempre definidos por uma ontologia universal (FELLAH; MALKI; ELÇI, 2016). Isso traz um novo desafio para o desenvolvimento e integração de SDS, devido à existência de diferentes ontologias que descrevem a mesma entidade do mundo real, provocando problemas de comunicação e interoperabilidade.

Visto que a construção de uma ontologia para uma fonte de dados é uma tarefa difícil e onerosa por natureza, diversas ferramentas surgiram para apoiar os desenvolvedores e especialistas de domínio no processo de construção de ontologias (CIMIANO; VÖLKER, 2005; SALEM; ABDELRAHMAN, 2010; NGUYEN; LU, 2016). Entretanto, tais ferramentas muitas vezes exigem a disponibilidade de um *dump* dos dados para gerar uma ontologia de domínio. Esta limitação dificulta a adoção de tais ferramentas em um ambiente SOA (*Service Oriented Architecture*), onde os dados são disponibilizados por meio de uma interface de serviço. Já as técnicas de *matching* de ontologias (EUZENAT; SHVAIKO, 2007), que buscam obter alinhamentos entre as ontologias, tentam resolver o problema da heterogeneidade. Mais especificamente, as técnicas extensionais são as mais adequadas ao escopo do problema em questão, posto que são utilizadas as instâncias de dados como indivíduos das ontologias para inferir correspondências entre elas.

Por outro lado, os serviços de dados estão suscetíveis a mu-

danças e as ontologias que descrevem os dados devem necessariamente evoluir em paralelo, caso contrário, elas se tornam inconsistentes. Além disso, para realizar o *matching* extensional, os *matchers* de ontologia geralmente assumem que as ontologias de entrada já foram criadas e associadas a uma grande quantidade de instâncias de dados. Isso se torna um desafio quando esses dados são parcialmente fornecidos por interfaces de serviço e quando não há uma relação direta entre os dados descritos pelas ontologias.

Yao et al. (2014) introduzem um mecanismo para construir uma ontologia unificada a partir de um conjunto de documentos JSON (*JavaScript Object Notation*) fornecidos por serviços de dados. No entanto, a ontologia criada não emprega conceitos semânticos definidos em fontes externas com o objetivo de reutilizar conceitos existentes e minimizar os problemas de interoperabilidade. Além disso, em estudos anteriores, pesquisadores direcionaram esforços em abordagens para enriquecer semanticamente *Web Services* (TOSI; MORASCA, 2015). A maioria dos trabalhos encontrados na literatura se concentra no enriquecimento da descrição dos serviços (BRAVO; RODRÍGUEZ; PASCUAL, 2014; LIRA; CAETANO, 2016; LUO et al., 2016), fornecendo apenas ferramentas para auxiliar o processo manual de criação das descrições semânticas. Em contrapartida, poucas propostas abordam o enriquecimento dos dados fornecidos por serviços. Salvadori et al. (2017a), por exemplo, propõem um método para enriquecer representações de *data-based microservices* utilizando links `owl:sameAs` e `rdfs:seeAlso`. Além disso, os autores também propõem um *framework*, o qual visa identificar alinhamentos entre ontologias heterogêneas. Um inconveniente dessa abordagem é o fato de que os *microservices* já devem empregar uma ontologia de domínio, bem como fornecer dados semânticos.

O desafio de se alcançar SDS, portanto, está na construção de ontologias a partir de serviços já existentes e na possibilidade de adaptar e evoluir essas ontologias de acordo com as demandas do seu domínio (ALFARIES; BELL; LYCETT, 2009). Em vista do exposto, foi formulada a seguinte pergunta de pesquisa: **como enriquecer automaticamente serviços de dados tradicionais para fornecimento de descrições e representações semânticas reutilizando conceitos já existentes?**

Para responder essa questão, propõe-se uma abordagem para enriquecimento semântico de serviços de dados de modo dinâmico, isto é, em tempo de execução. Dessa forma, serviços de dados legados conectados a fontes de dados sem semântica podem ser enriquecidos e fornecer dados conectados a conceitos semânticos. Como

contribuição, a abordagem possibilita uma maneira de construir e evoluir progressivamente ontologias de domínio a partir de representações sintáticas fornecidas por serviços de dados, além de reutilizar conceitos utilizados por fontes externas por meio de técnicas de *matching* de ontologias. Espera-se abstrair, do ponto de vista dos desenvolvedores de *software*, o processo oneroso de enriquecer semanticamente serviços conectados a fontes que armazenam dados sem semântica. Ademais, a proposta avança o estado da arte no sentido de prover um novo mecanismo que permita migrar serviços de dados definidos sintaticamente para serviços de dados semânticos. Assim, tarefas mais sofisticadas podem ser executadas, dando ensejo à reutilização e integração de dados entre diversas aplicações.

1.2 OBJETIVO GERAL

O objetivo geral deste trabalho é propor uma abordagem para enriquecimento semântico automático de serviços de dados, possibilitando o fornecimento tanto de descrições quanto de representações semânticas.

1.3 OBJETIVOS ESPECÍFICOS

O objetivo geral desta pesquisa pode ser alcançado atendendo aos principais objetivos específicos:

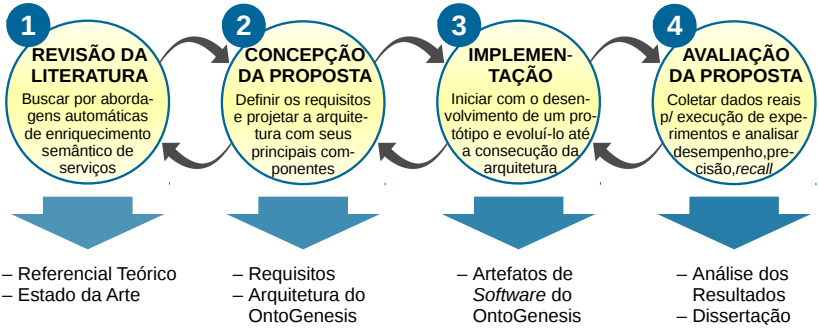
- Definir um mecanismo para construção e evolução de ontologias de domínio a partir de dados sintáticos fornecidos sob demanda por serviços de dados.
- Propor uma técnica de *matching* de ontologias adequada ao cenário de serviços de dados como forma de aumentar a expressividade semântica do serviço, conectando dados e conceitos comuns existentes em fontes externas.
- Definir um mecanismo para construção dinâmica de descrições e representações semânticas de serviços de dados tendo como base os conceitos descritos na ontologia construída.
- Desenvolver uma arquitetura que compreenda os mecanismos e técnicas elaborados nas alíneas anteriores.

1.4 MÉTODO DE PESQUISA

O método de pesquisa foi dividido em quatro etapas, cada uma contemplando as principais tarefas visando à consecução dos objetivos propostos. Os objetivos de cada etapa, bem como os resultados obtidos, estão sumarizados na Figura 1 e detalhados a seguir.

- **Etapa 1: Revisão da Literatura.** Inicialmente, a partir de uma colaboração em uma Revisão Sistemática da Literatura (RSL), foi identificado como a semântica é considerada no contexto de composição, seleção, descoberta e descrição de serviços. Tal RSL ajudou a entender a importância da semântica no contexto desses processos e a identificar nos trabalhos selecionados como a semântica é adotada no âmbito de *Web Services*. As conclusões oriundas dessa RSL deram subsídio a uma nova revisão bibliográfica dedicada a buscar na literatura abordagens e técnicas adicionais tratando exclusivamente de enriquecimento semântico automático de serviços de dados. Esta etapa, portanto, teve como objetivo estudar a área de pesquisa e, sobretudo, analisar o estado da arte de forma a garantir a originalidade das técnicas propostas.
- **Etapa 2: Concepção da Proposta.** Nesta etapa buscou-se definir um mecanismo automático para enriquecimento semântico de serviços de dados. Como resultado, foi concebida uma arquitetura, denominada OntoGenesis (ver Capítulo 4), a qual atende alguns requisitos fundamentais para cumprir o objetivo proposto. Primeiramente, foram contempladas técnicas alinhadas às encontradas na literatura, como *matching* de ontologias e *Ontology Learning* (OL), aspirando à construção e à evolução de ontologias de domínio para serviços de dados. Em seguida, foi essencial definir um mecanismo para que serviços (que originalmente proveem dados sintáticos) fossem capazes de servir dados com semântica, mais precisamente dados conectados, de modo a abstrair do desenvolvedor de *software* grande parte do esforço necessário para a construção de SDS. Além disso, foi projetada a construção automática de uma descrição semântica para o serviço de dados. Por fim, foi concebida uma forma de incorporar ao serviço uma interface de acesso à ontologia criada e à sua nova descrição semântica, permitindo que clientes e sistemas externos possam consumir a ontologia e a descrição geradas para o serviço.

Figura 1 – Etapas do método de pesquisa.



Fonte: Elaborado pelo autor (2018).

- **Etapa 3: Implementação.** O objetivo desta etapa está na implementação da arquitetura do OntoGenesis, definida na etapa 2, de modo a abarcar todos os mecanismos propostos. Com o intuito de avaliar a aplicabilidade da proposta em curto prazo, a implementação foi baseada no processo de prototipação. Portanto, as funcionalidades essenciais da arquitetura – como, por exemplo, a construção de ontologias de domínio para serviços de dados e o mecanismo de *matching* para identificar correspondências com outras ontologias externas – foram desenvolvidas e testadas para se obter resultados preliminares. Após a análise dos resultados, tem-se como foco a implementação dos demais algoritmos e a evolução do protótipo contemplando todos os componentes da arquitetura.

- **Etapa 4: Execução dos Experimentos e Avaliação dos Resultados.** Nesta última etapa foram coletados dados reais para execução dos experimentos. Em seguida, avaliou-se a proposta observando métricas de conformidade, como precisão, cobertura e *F-Measure*. Nesse sentido, foram avaliadas a ontologia gerada e suas relações de equivalência com ontologias externas já existentes. Além disso, métricas de desempenho também foram analisadas. Observou-se, principalmente, o tempo decorrido para realizar a construção e *matching* da ontologia, o tempo para geração dinâmica da descrição e representação semânticas, além do consumo médio de memória e CPU. Por fim, a abordagem proposta foi comparada com outros trabalhos encontrados na literatura, utilizando os mesmos

conjuntos de dados e ambiente de execução.

As etapas realizadas nesta pesquisa são iterativas e o método permite regressar e avançar as etapas conforme a necessidade da pesquisa. Durante a implementação, por exemplo, há a necessidade retornar à etapa 2 para adequar a concepção da proposta com novos requisitos que tendem a surgir. Outrossim, durante a execução da etapa de avaliação, podem ser identificadas melhorias/correções, necessitando, retornar à etapa 3 para implementá-las.

1.5 ESTRUTURA DA DISSERTAÇÃO

O restante deste trabalho está organizado em cinco capítulos:

- O **Capítulo 2 - Fundamentação Teórica** expõe os fundamentos teóricos necessários para entendimento deste trabalho. Nesse capítulo são discutidos assuntos como Web Semântica e suas tecnologias, bem como técnicas de construção e *matching* de ontologias. São apresentados também os principais conceitos atinentes a serviços Web tradicionais (sintáticos) e serviços Web semânticos.

- O **Capítulo 3 - Estado da Arte** discute os trabalhos encontrados na literatura relevantes a esta pesquisa. O estado da arte está organizado em duas categorias: i) trabalhos que abordam construção e *matching* de ontologias, e ii) trabalhos que abordam o enriquecimento semântico de serviços Web.

- O **Capítulo 4 - Abordagem Proposta** apresenta o mecanismo elaborado para enriquecer dinamicamente os serviços de dados tradicionais com semântica. Esse capítulo também detalha a arquitetura proposta, apresentando os seus componentes e os aspectos mais importantes de implementação.

- O **Capítulo 5 - Avaliação e Resultados** descreve o cenário elaborado para a execução dos experimentos e a metodologia adotada. Os resultados obtidos também são discutidos e comparados com abordagens similares encontradas na literatura.

- O **Capítulo 6 - Conclusões** tece as considerações finais e identifica as limitações desta pesquisa, apontando direcionamentos para trabalhos futuros.

Por fim, o **Apêndice A** lista as publicações acadêmicas alcançadas durante o mestrado; o **Apêndice B** fornece detalhes acerca do protocolo utilizado na RSL; o **Apêndice C** exemplifica como os parâmetros dos *endpoints* de um serviço são extraídos, a partir das anotações no código fonte, para gerar a descrição semântica do serviço; e o **Apêndice D** apresenta um exemplo de ontologia de domínio gerada pela arquitetura proposta.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica necessária para entendimento da abordagem proposta neste trabalho. São apresentadas as principais tecnologias empregadas na Web Semântica e discutidos os conceitos-chave relativos a serviços Web e serviços de dados, e o papel da semântica nesse contexto.

2.1 WEB SEMÂNTICA

A *World Wide Web* se apresenta atualmente como um grande repositório de dados – em sua maior parte, dados não estruturados. Em geral, estes dados são dispostos em documentos HTML, destinados ao consumo de seres humanos, o que dificulta a sua compreensão e manipulação por dispositivos computacionais. Enquanto dados estruturados têm uma estrutura fixa e de fácil análise por agentes de *software*, a manipulação de dados não estruturados (ou semiestruturados) ainda é uma tarefa desafiadora (VORUGANTI, 2016). Em vista disso, a Web Semântica busca estender a visão da Web clássica, permitindo que não apenas os documentos sejam disponibilizados e vinculados a outros documentos, mas também os seus conteúdos (BIZER; HEATH; BERNERS-LEE, 2009).

Neste sentido, a visão inicial de Berners-Lee, Hendler e Lassila (2001) consiste em prover uma estrutura semântica para os documentos Web, de modo que os seus dados tenham um significado associado. Com isso, tanto humanos quanto máquinas passam a trabalhar de forma cooperativa navegando pelos documentos e executando tarefas mais sofisticadas, predispondo, assim, ao compartilhamento, reúso e à integração de dados entre diversas aplicações (KASHYAP; BUSSLER; MORAN, 2008). Em outras palavras, a Web Semântica busca automatizar a recuperação e análise de informações relevantes, permitindo que máquinas também compreendam os termos descritos pelos dados e realizem inferências através de *softwares* denominados *reasoners*[1].

Para tanto, a Web Semântica está fundamentada na utilização de alguns padrões e tecnologias, como dados conectados (ou dados ligados), o modelo de dados RDF e ontologias.

[1] Raciocinadores, em português. Neste texto será empregado o termo em inglês, devido ao pouco uso do termo traduzido na literatura científica.

2.1.1 Dados abertos conectados

Dados abertos são aqueles que podem ser livremente usados, modificados e redistribuídos por qualquer pessoa – sujeitos ou não à exigência de licenças (OPEN DEFINITION, 2014). O objetivo é que tanto pessoas físicas quanto jurídicas possam acessar os dados e explorá-los para qualquer propósito. Recentemente, movimentos internacionais vêm encorajando a disseminação da prática de dados abertos, sobretudo com iniciativas de transparência nos registros públicos, também chamados de dados abertos governamentais (DING; PERISTERAS; HAUSENBLAS, 2012). A ideia é incentivar a publicação de dados produzidos por órgãos governamentais com o intuito de efetivar a sua reutilização e integração com outras fontes de dados. Exemplos destas iniciativas podem ser percebidos em alguns países, como o Reino Unido (HALL et al., 2012) e o Brasil (AHMAR; ALMEIDA, 2017).

O conceito de dados conectados (ou *Linked Data*, do inglês), por sua vez, pode ser entendido como um conjunto de boas práticas para a publicação e ligação de conjuntos de dados estruturados na Web. No esteio da Web Semântica, os dados abertos conectados (LOD, do inglês *Linked Open Data*) desempenham um papel fundamental para a publicação, compartilhamento e ligação de dados, dando ensejo à Web de dados (BIZER et al., 2008). Essa interconexão de informações estruturadas advém da combinação de conteúdos da Web com descrições semânticas de seus significados e relacionamentos. Berners-Lee (2006), inventor da Web e um dos precursores do *Linked Data*, sugeriu um esquema de implantação para avaliar a qualidade dos dados abertos conectados. Conforme ilustrado na Figura 2, os dados publicados na Web são classificados em cinco estrelas, de acordo com a sua maturidade.

Os dados que possuem uma estrela são aqueles disponibilizados na Web representados em qualquer formato, sob a utilização de uma licença pública. Dados com duas estrelas, por sua vez, devem

Figura 2 – Classificação cinco estrelas de *Linked Data*.

Fonte: Adaptado de Berners-Lee (2006).

estar estruturados e legíveis por agentes de *software*, por exemplo, uma planilha XLS do Microsoft Excel. A terceira estrela se aplica a documentos em formatos não proprietários, como o CSV, por exemplo. Dados com quatro estrelas devem estar estruturados nos moldes da Web Semântica, usando padrões W3C (*World Wide Web Consortium*), como o RDF (ver seção 2.1.2). Para receber a quinta estrela os dados precisam estar ligados com dados de terceiros a fim de estabelecer contextos e enriquecer sua expressividade. Nesse sentido, as informações estão semanticamente relacionadas.

Dentre os princípios que regem a publicação de dados conectados, destacam-se: i) a utilização de um identificador global único – ou seja, um URI (*Uniform Resource Identifier*) – para identificar e localizar unicamente os recursos; ii) a adoção de um modelo de dados comum (RDF, por exemplo), o qual deve descrever conceitos e instâncias por meio de um URI; e iii) o uso de um protocolo e linguagem de consulta para acesso aos dados (SPARQL *Query Language*). Complementarmente, os *datasets* publicados devem incluir links para outros recursos, de modo que informações adicionais possam ser obtidas (HEATH; BIZER, 2011).

A publicação de dados abertos conectados está em constante crescimento, dispondo de mais de 2 mil *datasets* representados em RDF espalhados pela Web (ABELE et al., 2017). Tais conjuntos de dados abrangem diversos domínios, incluindo informações da Wikipedia (disponibilizados por meio da DBpedia), dados governamentais e geoespaciais, bioinformática, publicações científicas, dentre outros. De acordo com a visão de Bizer, Heath e Berners-Lee (2009), a Web tem um enorme potencial para se tornar um espaço global de dados conectados. Contudo, LOD ainda se encontra incipiente e os dados publicados na Web ainda são, majoritariamente, não estruturados e sem semântica. Isso se deve ao fato de seu uso ainda requerer uma familiaridade dos usuários com o modelo de dados RDF e a sua linguagem de consulta SPARQL, além de um maior conhecimento sobre o conjunto de dados disponível e o seu conteúdo.

2.1.2 RDF

O *framework* de descrição de recursos (RDF, do inglês *Resource Description Framework*) provê um modelo para estruturar e representar os dados de forma que estes possam ser legíveis a humanos e processados por máquinas (W3C, 2014b). Nesse contexto, os dados são denominados de *Web resources*, ou apenas recursos,

que podem ser tanto metadados individualmente identificáveis na Web, quanto informações específicas sobre o seu conteúdo, ou seja, os seus valores.

A declaração de um recurso RDF pode ser formalmente definida por uma tripla $d = (s, p, o)$, onde s é o sujeito, p é o predicado e o é o objeto da declaração. O sujeito s representa o recurso descrito, o predicado p identifica uma característica ou propriedade do sujeito, e o objeto o representa o valor atribuído à característica do sujeito. Ademais, ambos sujeito e predicado são representados por um recurso através de um URI.

Há distintas formas de representação/serialização de um recurso para o RDF, como, por exemplo, RDF/XML, RDFa, N-Triples, Turtle e JSON-LD (JSON for *Linked Data*). Este último trata-se de uma extensão do JSON compatível com sistemas e serviços baseados em RDF. Além de ser um dos formatos mais recentes, o JSON-LD se tornou uma recomendação do W3C e é, desde 2010, tido como um padrão recomendado pelo mesmo órgão (SPORNY et al., 2014). Adicionando-se a isso, o JSON-LD tem sido cada vez mais utilizado pela comunidade de desenvolvedores de *software*[2].

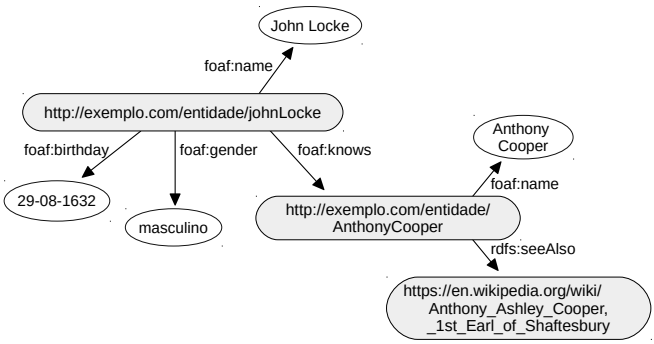
Ademais, um recurso descrito em RDF pode ser representado também como um grafo orientado, no qual os nós representam sujeitos e objetos, e as arestas expressam o predicado, isto é, o relacionamento entre sujeito e objeto. A Figura 3 (a) ilustra o grafo dos dados de uma pessoa, enquanto (b) apresenta a descrição desse recurso em JSON-LD. Os recursos são descritos através de propriedades, as quais possuem valores atribuídos, como por exemplo, o recurso relacionado à "John Locke"possui uma propriedade `foaf:birthday` cujo valor é "29-08-1632". É possível perceber ainda que o recurso de "Anthony Cooper"possui um link `rdfs:seeAlso` para outro recurso semelhante já existente identificado por um URI da Wikipedia.

Uma descrição RDF, entretanto, limita-se no que diz respeito à descrição sintática e individual dos recursos. Para que haja um nível adequado de interpretação dos dados e seus relacionamentos semânticos, é necessário especificar um contexto de aplicação. Para isso, RDFS (*Resource Description Framework Schema*), ou esquema RDF, estende o vocabulário básico do RDF, permitindo que novas relações semânticas e restrições mais complexas possam ser adicionadas e compartilhadas entre os recursos. De acordo com a definição fornecida pelo W3C (W3C, 2014c), um vocabulário, no âmbito da

[2] Estatísticas de uso do JSON-LD: <https://trends.builtwith.com/docinfo/JSON-LD>.

Figura 3 – Exemplo de descrição RDF em (a) grafo orientado e (b) JSON-LD.

(a)



(b)

```
{ "@context" : {
    "foaf": "http://xmlns.com/foaf/0.1/",
    "rdfs": "http://www.w3.orf/2000/01/rdf-schema#"
  },
  "@type":  "Person",
  "@id": "http://exemplo.com/entidade/johnLocke",
  "foaf:name": "John Locke",
  "foaf:birthday": "29-08-1632",
  "foaf:gender" : "male",
  "foaf:knows": [
   {  "@type": "Person",
      "@id": "http://exemplo.com/entidade/AnthonyCooper",
      "name": "Anthony Cooper",
      "rdfs:seeAlso":
       ↪   "https://en.wikipedia.org/wiki/Anthony_
       ↪                  Ashley_Cooper,_1st_Earl_of_Shaftesbury"
   } ]
}
```

Fonte: Elaborado pelo autor (2018).

Web Semântica, define conceitos e relacionamentos (também conhecidos como termos) utilizados para descrever e representar um domínio específico, facilitando, desse modo, a recuperação e interpretação da informação. Logo, RDFS pode ser entendido como uma coleção de recursos do RDF que podem ser utilizados para descrever propriedades de outros recursos RDF, especificado, assim, um conjunto de classes, propriedades e restrições entre os relacionamentos desses recursos (W3C, 2014a).

As classes são abstrações para agrupar recursos com características similares, que podem ser organizadas de forma hierár-

quica (por meio da propriedade `rdfs:subClassOf`). A utilização de classes torna o modelo RDF extensível, visto que as definições de esquemas existentes podem ser herdadas, criando, assim, novas especializações de metadados. Um esquema RDF permite, por exemplo, que recursos sejam instâncias de uma ou mais classes – indicado pela propriedade `rdf:type`. Por outro lado, as propriedades expressam as relações entre classes e suas instâncias, de forma análoga à noção de atributos no paradigma de programação orientada a objetos. As propriedades são descritas no RDFS por meio da classe `rdf:Property`. Por fim, as propriedades `rdfs:range` e `rdfs:domain` são exemplos de restrições que podem ser estabelecidas sobre as propriedades de um recurso. A primeira é utilizada para indicar que os valores de uma determinada propriedade são instâncias de uma ou mais classes, limitando, portanto, os valores que podem ser aplicados a uma propriedade específica. Já a restrição `rdfs:domain` especifica a classe na qual uma determinada propriedade pode ser aplicada, limitando, portanto, as instâncias de classes que podem utilizar a propriedade.

No exemplo da Figura 3 (b) foi utilizada a classe `Person` pertencente ao vocabulário FOAF (*Friend Of A Friend*), o qual expressa o relacionamento entre pessoas. Outros vocabulários estão disponíveis na Web para definir o contexto e significado de recursos sem ambiguidades, como os vocabulários da DBpedia, Dublin-Core, dentre outros. Vale ressaltar que na Figura 3 cada recurso do JSON-LD possui um `@id` e um `@type`. O `@id` identifica unicamente o recurso que está sendo descrito, enquanto o `@type` especifica o recurso descrito pelo JSON-LD com um conceito da ontologia. Além disso, é definido um contexto, denotado pela propriedade `@context`, usado para mapear termos (geralmente palavras curtas) que expandem para um URI – definida por um vocabulário/ontologia.

2.1.3 Ontologias

O termo ontologia origina-se do grego *ontos* (ser) e *logos* (razão), embora este último possa ser interpretado originalmente como palavra ou discurso (ALMEIDA; BAX, 2004). Há na literatura uma razoável distinção do termo ontologia, sendo este visto não apenas no senso filosófico, mas também como um artefato de engenharia utilizado pela computação, sobretudo, pela comunidade de inteligência artificial (AI). Guarino (1998) fornece uma discussão mais aprofundada acerca do assunto e distingue ambos os termos:

> No sentido filosófico, podemos nos referir a uma ontologia como um sistema particular de categorias que descrevem uma certa visão do mundo. Como tal, esse sistema não depende de uma linguagem específica: a ontologia de Aristóteles é sempre a mesma, independentemente da linguagem utilizada para descrevê-la. Por outro lado, no seu uso mais prevalente na AI, uma ontologia se refere a um artefato de engenharia, constituído por um vocabulário específico usado para descrever uma certa realidade, além de um conjunto de premissas explícitas quanto ao significado pretendido dos termos do vocabulário. (GUARINO, 1998, p. 2, tradução nossa).

Doravante, este trabalho utiliza o termo supracitado no contexto da computação. Desde que o termo passou a ser utilizado além da esfera filosófica, inúmeras definições de ontologia foram apresentadas na literatura científica (GÓMEZ-PÉREZ; FERNÁNDEZ-LÓPEZ; CORCHO, 2007). Uma das definições mais citadas – e utilizada como referência neste trabalho – é feita por Gruber (1993), o qual define ontologia como uma especificação explícita de uma conceitualização. Borst (1997) complementa tal definição, indicando que uma ontologia é uma especificação formal de uma conceitualização compartilhada. O objetivo, portanto, é organizar e compartilhar conceitos relacionados a um determinado domínio, explicitando as relações entre diversas entidades e seus axiomas de uma maneira inteligível não apenas por humanos, mas também por agentes de *software*.

As ontologias estão incorporadas ao estudo da Web Semântica de modo a prover vocabulários comuns para diferentes domínios de conhecimento. São amplamente utilizadas em aplicações relacionadas à gestão do conhecimento, inteligência artificial, integração de bancos de dados, recuperação de informações, bioinformática e educação (GÓMEZ-PÉREZ; FERNÁNDEZ-LÓPEZ; CORCHO, 2007). Uschold e Jasper (1999) identificam três principais usos de ontologias: i) auxiliar na comunicação entre seres humanos; ii) alcançar a interoperabilidade entre diferentes *softwares*; e iii) melhorar o *design* e a qualidade dos *softwares*. Tal artefato, portanto, desempenha um papel fundamental na integração de sistemas computacionais.

Segundo Guarino (1997), as ontologias podem ser classificadas de acordo com duas dimensões: nível de detalhamento e nível de dependência.

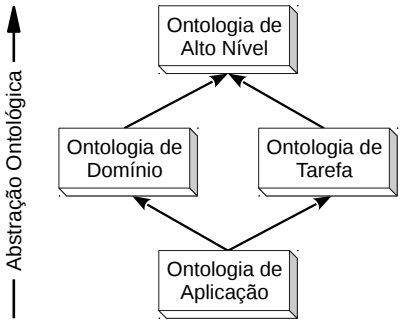
1. No nível de detalhamento, uma ontologia muito detalhada busca especificar o significado de um vocabulário, com o objetivo de estabelecer um consenso acerca do significado dos termos do vocabulário em questão. Por outro lado, em um cenário onde os usuários já detêm um consenso sobre determinados conceitos, pode-se desenvolver e compartilhar uma ontologia mais simples – ou seja, menos detalhada – levando em consideração operações específicas de inferência. Logo, as ontologias são classificadas em dois tipos:

    - Ontologias de referência (também chamadas de *off-line* ou *very detailed*): compreendem uma representação formal e autêntica do domínio de interesse. Tais ontologias são destinadas a auxiliar os seres humanos em tarefas comunicativas, como a negociação de significado ou o estabelecimento de consenso sobre os conceitos de um domínio. São utilizadas como um modelo conceitual, sendo uma referência acerca do conhecimento disponível (GUARINO, 1998; GUIZZARDI, 2007; FALBO, 2014).
    - Ontologias operacionais (também conhecidas como *on-line*, *shareable* ou *lightweight*): baseadas em ontologias de referência, embora com o objetivo de serem compreendidas por agentes de *software*. Dessa forma, ontologias operacionais não se destinam a representar a realidade de forma precisa, mas sim em fornecer propriedades computacionais que as tornam processáveis por máquinas. Nesse sentido, atributos de qualidade, como precisão e veracidade, podem ser relaxados em prol de propriedades computacionais (GUIZZARDI, 2007; FALBO, 2014).

2. No nível de dependência, as ontologias podem ser classificadas em quatro tipos, de acordo com o seu nível de generalidade. A Figura 4 ilustra a dependência entre as ontologias, representando por meio de setas os relacionamentos de especialização. As ontologias inferiores, portanto, especializam termos mais abstratos introduzidos em ontologias de nível superior.

    - Ontologia de alto nível (*top-level ontology*): descreve conceitos gerais como espaço, tempo, assunto, objeto, evento, ação, etc., os quais são independentes de um problema ou domínio específico.

Figura 4 – Tipos de ontologia de acordo com o seu nível de dependência.



Fonte: Adaptado de Guarino (1997).

- Ontologia de domínio (*domain ontology*): descreve o vocabulário comum para compartilhar informações de um domínio específico (por exemplo: bioinformática ou governamental), especializando termos de uma ontologia de alto nível.
- Ontologia de tarefa (*task ontology*): descreve um vocabulário utilizado para realizar uma tarefa ou atividade independente de domínio, especializando termos de uma ontologia de alto nível.
- Ontologia de aplicação (*application ontology*): descreve conceitos baseados tanto em um domínio específico quanto em uma tarefa, sendo uma especialização de ambas as ontologias.

As ontologias de domínio são as mais comumente desenvolvidas e têm aplicações em diversos campos, como medicina, engenharias, modelagem organizacional, governamental, dentre outros (GUIZZARDI, 2007). Uma vez que o uso de ontologias na presente pesquisa visa à integração de dados e ao enriquecimento de serviços com conceitos semânticos atinentes a um determinado domínio de aplicação, a abordagem introduzida neste trabalho busca construir ontologias operacionais de domínio. Portanto, a maioria das discussões subsequentes se aplica a este tipo de ontologia, ainda que este possa ser estendido, com algumas restrições, para outros tipos. Vale ressaltar ainda que uma ontologia de domínio não tem como finalidade descrever todo o conhecimento de um domínio. Conhe-

cimentos empíricos, compilados ou práticos, que são dependentes da tarefa ou aplicação particular, podem estar presentes apenas em uma ontologia de aplicação (FALBO, 1998).

2.1.3.1 Construção de ontologias

Para modelar o domínio de uma ontologia é preciso levar em consideração os seus indivíduos, classes, atributos e relações. Os indivíduos são as instâncias básicas da definição do domínio da ontologia. Já os conceitos são definidos por meio de classes, que formam os tipos dos indivíduos. Os atributos, por sua vez, são as propriedades que os indivíduos podem ter e compartilhar entre si, enquanto as relações representam as formas de relacionamento entre indivíduos. Ademais, os axiomas podem ser providos com o intuito de estabelecer a semântica dos termos da ontologia. Com eles é possível definir e representar informações adicionais sobre os conceitos e suas relações, incluindo restrições sobre propriedades e seus valores.

Definir ontologias para serem utilizadas por *softwares* requer o uso de uma linguagem apropriada que permita descrever conceitos e relacionamentos. Uma das linguagens amplamente empregadas para definir e instanciar ontologias é a *Web Ontology Language* (OWL) – um padrão derivado do RDF e recomendado pelo W3C (W3C, 2004a). Além de permitir *reasoning* baseado em lógica descritiva (BAADER, 2010), a OWL fornece ainda vocabulário adicional (como os predicados `owl:equivalentProperty`, `owl:equivalentClass` e `owl:sameAs`, que definem equivalências entre propriedades, classes e indivíduos definidos em ontologias distintas, respectivamente) para facilitar a interoperabilidade entre *softwares*. A OWL é uma revisão das linguagens DAML e OIL (MCGUINNESS et al., 2002) e possui mais facilidades para expressar o significado e a semântica de conteúdos da Web interpretáveis por máquinas.

Tradicionalmente, as ontologias de domínio são construídas por meio de um trabalho colaborativo entre especialistas do domínio e engenheiros de ontologia. Há na literatura uma variação da expressão "construção de ontologia", podendo ser compreendidos termos similares como desenvolvimento, elicitação, elaboração ou geração de ontologias. Há, ainda, uma gama de metodologias e ferramentas disponíveis que permeiam a construção de ontologias (SUBHASHINI; AKILANDESWARI, 2011). Exemplos de metodologias são a Skeletal (USCHOLD, 1996), TOVE (GRÜNINGER; FOX, 1995) e Methontology (FERNÁNDEZ-LÓPEZ;

GÓMEZ-PÉREZ; JURISTO, 1997); já o Protégé[3] destaca-se na linha ferramental. Tais metodologias foram desenvolvidas com objetivos distintos e aplicações particulares, muitas vezes contemplando técnicas e procedimentos manuais de difícil compreensão para os desenvolvedores de *software*. Em vista disso, a construção de ontologias é considerada, por natureza, como uma tarefa significativamente custosa em termos de tempo e esforço demandados (BUITELAAR; CIMIANO, 2008).

Não é escopo deste trabalho aprofundar-se nas metodologias existentes para desenvolver ontologias, visto que tais métodos se adequam apenas a uma realidade que exige esforço ainda manual e intelectual. A abordagem defendida neste trabalho, por outro lado, busca estimular a capacidade de construir ontologias de domínio automaticamente – ou seja, por um computador, sem intervenção ou supervisão de seres humanos – e de forma incremental, focando essencialmente no enriquecimento semântico de serviços de dados.

As abordagens propostas na literatura para construção de ontologias se baseiam em métodos supervisionados, sendo este um gargalo para se construir sistemas e serviços com suporte semântico (ALFARIES, 2010). A área de pesquisa que busca automatizar e, por conseguinte, facilitar o desenvolvimento de ontologias, minimizando os inconvenientes encontrados na construção manual, é conhecida como *Ontology Learning* (OL). A OL tem como objetivo extrair de forma (semi-)automática elementos ontológicos, como classes, propriedades, relacionamentos e também as instâncias, de diferentes recursos (MAEDCHE; STAAB, 2001). Nesse sentido, diversas técnicas e ferramentas foram propostas com a finalidade de aprender conceitos e relacionamentos, auxiliando, assim, na construção de ontologias. Maedche e Staab (2001) classificam as diferentes abordagens para aprendizado/construção de ontologias de acordo com o tipo de entrada recebido. Tem-se basicamente os seguintes tipos:

- Construção de ontologia a partir de textos: consiste em desenvolver ontologias aplicando técnicas de análise de linguagem natural em textos não estruturados.
- Construção de ontologia a partir esquemas semi-estruturados: envolve a elaboração de ontologias a partir de fontes que contenham alguma estrutura pré-definida, como XML e JSON.
- Construção de ontologia a partir de esquemas estruturados: busca construir ontologias extraindo conceitos e relacionamen-

[3] Sítio do Protégé: <https://protege.stanford.edu/>.

tos relevantes a partir do conhecimento existente em dados estruturados, como bancos de dados relacionais.

A adoção de técnicas de OL é considerada como um passo importante tanto para a construção quanto para a evolução de ontologias. Visto que o domínio está suscetível a mudanças, as ontologias precisam acompanhar tais evoluções e manter-se atualizadas. Dessa forma, construir mecanismos capazes de aprimorar o processo de construção, bem como manutenção, de ontologias durante o desenvolvimento de sistemas é fundamental para difundir o uso de tecnologias basilares da Web Semântica. Um dos desafios encontrados, portanto, é permitir, de forma menos dispendiosa, que as ontologias se adaptem e evoluam em conformidade com as demandas que tendem a surgir no domínio de aplicação. Algumas soluções focadas em automatizar o desenvolvimento de ontologias têm sido propostas na literatura e serão discutidas no Capítulo 3.

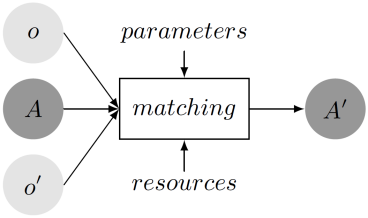
2.1.3.2 *Matching* de ontologias

Conforme já mencionado na introdução desta dissertação, não é realista assumir na prática a existência de uma ontologia universal que descreva todos os conceitos e relacionamentos de um domínio. Cada projeto, equipe de desenvolvimento ou provedor de dados, em geral, define o seu próprio modelo ontológico, tendendo a produzir diversos conceitos que descrevem o mesmo objeto do mundo real (FELLAH; MALKI; ELÇI, 2016).

A despeito de incluir predicados para denotar equivalência de indivíduos, classes e propriedades, como visto na seção anterior, o vocabulário de equivalência da OWL nem sempre é utilizado pelos designers de ontologias. Isso se torna um problema quando os agentes de *software* são confrontados com dados descritos por ontologias heterogêneas, dado que a sua semântica pode não ser compreendida pelo sistema. Ao passo que diferentes conceitos para um mesmo objeto de um domínio são empregados, problemas de interoperabilidade tornam-se frequentes.

Sob este prisma, faz-se necessário estabelecer associações entre os diferentes conceitos de ontologias de maneira a explorar o potencial da Web Semântica. Embora a OL seja uma área de estudo interessante voltada à automatização do processo de construção de uma ontologia, é essencial permitir que os conceitos sejam reutilizados. Além de ampliar as possibilidades de integração, o reúso permite que operações de *reasoning* sejam executadas por agentes

Figura 5 – Processo de *matching* de ontologias.



Fonte: Euzenat e Shvaiko (2007).

de *software*. Um tópico de pesquisa ativo na literatura, conhecido como *matching* de ontologias, busca atacar esse problema detectando automaticamente relações de equivalência entre entidades de ontologias heterogêneas (EUZENAT; SHVAIKO, 2007).

De acordo com Pavel e Euzenat (2013), a operação de *matching* entre duas ontologias O e O' produz um alinhamento A'. Para se obter um alinhamento A', podem ser utilizados, além das ontologias, um alinhamento A já conhecido, parâmetros de *matching* (tais como pesos ou *thresholds*) e recursos externos (como, por exemplo, conhecimentos comuns ou vocabulários de domínio específico que descrevem termos das ontologias), como ilustrado na Figura 5.

Um alinhamento é caracterizado por um conjunto de correspondências entre entidades (como classes e propriedades) das ontologias. Já uma correspondência C pode ser representada por uma 5-tupla $C = (i, e, e', r, f)$, onde i é um identificador da correspondência gerada; e é a entidade (conceito ou indivíduo) pertencente à ontologia O; e' é a entidade pertencente à segunda ontologia O'; r representa o tipo específico do relacionamento identificado (como, por exemplo, uma equivalência de classes ou propriedades, subsunção e disjunção); e f determina o grau de similaridade entre as entidades das ontologias, tendo um valor geralmente compreendido entre o intervalo [0,1]. O f, muitas vezes conhecido como a força do alinhamento, define o grau de confiança da correspondência; portanto, quanto mais próximo de 1, maior é a probabilidade de a correspondência ser verdadeira.

Euzenat e Shvaiko (2007) propõem alguns métodos para identificar a similaridade entre ontologias com base em características específicas das entidades e os classificam da seguinte forma:

- Baseada em Nomes: realiza comparação de *strings* utilizando métodos terminológicos para encontrar entidades semelhantes

com base em seus nomes, rótulos ou comentários.

- Baseada na Estrutura: em vez de comparar somente os nomes das entidades, a estrutura dos elementos encontrados nas ontologias, isto é, a estrutura taxonômica, também é comparada.

- Extensional: analisa as instâncias das classes. Assim, caso as classes de duas ontologias compartilhem indivíduos, o método provavelmente produzirá uma correspondência para elas.

- Baseada em Semântica: método dedutivo que utiliza geralmente um *reasoner* para inferir correspondências entre as ontologias.

Euzenat e Shvaiko (2007) listam ainda diversas aplicações nas quais o processo de *matching* de ontologias vem sendo utilizado como uma solução eficaz para o problema da interoperabilidade. Por exemplo, nas redes P2P (*Peer-to-Peer*) semânticas – que utilizam ontologias para descrever os seus dados e aperfeiçoar tarefas de buscas na rede – o alinhamento entre as ontologias facilita a comunicação entre os nodos e, por conseguinte, a interoperabilidade na rede. No contexto de integração de bases de dados que utilizam diferentes ontologias, os alinhamentos estabelecem correspondências entre os conceitos utilizados pelas diferentes bases e auxiliam na reescrita de consultas, bem como na eliminação de redundâncias e duplicações no resultado.

Ademais, a integração de Serviços Web pode ser aprimorada por meio de Serviços Web Semânticos, os quais utilizam ontologias para descrever, por exemplo, as suas capacidades e os seus dados. Nesse sentido, o alinhamento de ontologias é fundamental para estabelecer relações entre os diversos serviços, interconectar diferentes formatos de entrada e saída e, sobretudo, auxiliar em processos de descoberta e composição de serviços.

2.2 SERVIÇOS WEB

No âmbito de sistemas distribuídos, um serviço pode ser entendido como um elemento computacional que executa determinadas funções, sendo utilizado para compor aplicações distribuídas com baixo custo (PAPAZOGLOU, 2003). Um serviço Web (do inglês *Web Service*), por sua vez, é definido pelo W3C como um sistema projetado para prover interoperabilidade entre as máquinas

em uma rede de intercomunicação (W3C, 2004b). Mais especificamente, um serviço Web pode ser visto como um meio de expor as funcionalidades de um sistema, disponibilizando-as por meio das tecnologias da Web (ALONSO et al., 2004). Logo, um serviço disponibilizado na Web é composto por uma interface identificada por um URI e descrita em um formato processável por máquinas, a qual contém parâmetros de entrada e saída, os tipos de dados e as operações disponíveis. Dessa forma, o serviço Web pode ser descoberto por outros agentes de *software*, os quais podem interagir com o serviço utilizando, usualmente, o protocolo HTTP para transmissão de mensagens. Para fins de simplificação, o termo serviço ao longo desta dissertação se refere a um serviço Web.

Em geral, os serviços são descritos de forma sintática, em que apenas os nomes de operações e a estrutura dos dados de entrada e saída são especificados, mas não a sua semântica. O W3C sugere o uso de três tecnologias para serviços Web: WSDL (*Web Service Description Language*) para descrição das funcionalidades (operações e parâmetros) de um serviço e sua localização; UDDI (*Universal Description, Discovery and Integration*) para publicação e busca de serviços; e SOAP (*Simple Object Access Protocol*) como protocolo de comunicação. Os serviços que utilizam tal protocolo são conhecidos como serviços SOAP.

Por outro lado, a adoção de serviços Web tem evoluído significativamente com a proliferação dos serviços que adotam o estilo arquitetural REST (FIELDING, 2000). Com o REST, os serviços utilizam o mecanismo padrão da Web, em que qualquer entidade é considerada um recurso identificado por um URI que pode ser acessada via operações HTTP. Serviços RESTful – ou seja, que seguem o estilo arquitetural REST – costumam utilizar a WADL (*Web Application Description Language*) para descrever os seus recursos, muito embora a literatura evidencie casos de descrições WSDL sendo fornecidas por tais serviços, como o trabalho de Lira e Caetano (2016).

Há na literatura uma certa confusão com relação aos termos "Serviço REST"e "Web API". Muitos autores consideram ambos os termos como sinônimos, apesar de muitas Web APIs não contemplarem todas as restrições arquiteturais do REST definidas originalmente por Fielding (2000). O estilo REST impõe algumas restrições que tornam os serviços genuinamente RESTful, como, por exemplo, interface uniforme, modelo cliente-servidor, interação entre cliente e servidor sem estado, *caching* de recursos e sistema em camadas. No entanto, é comum que alguns desenvolvedores violem um subconjunto destas restrições (LANTHALER, 2013).

Quanto à sua expressividade, os serviços Web podem ser classificados como tradicionais – que utilizam descrições sintáticas para fornecer dados semi-estruturados – e semânticos, os quais empregam tecnologias da Web Semântica para prover interfaces e dados processáveis por máquinas. Estes últimos buscam aperfeiçoar a descoberta, composição e interoperabilidade de serviços – áreas bem endereçadas na literatura (LEMOS; DANIEL; BENATALLAH, 2015; KÜSTER; LAUSEN; KÖNIG-RIES, 2008), porém com muitos desafios ainda a serem enfrentados (BOUGUETTAYA et al., 2017). Ademais, independente da tecnologia empregada, um serviço pode expor operações relacionadas a um domínio específico, bem como gerir o ciclo de vida das entidades do domínio provendo meios de acesso e manipulação a dados persistidos em alguma fonte de dados. Estes últimos são conhecidos como serviços de dados.

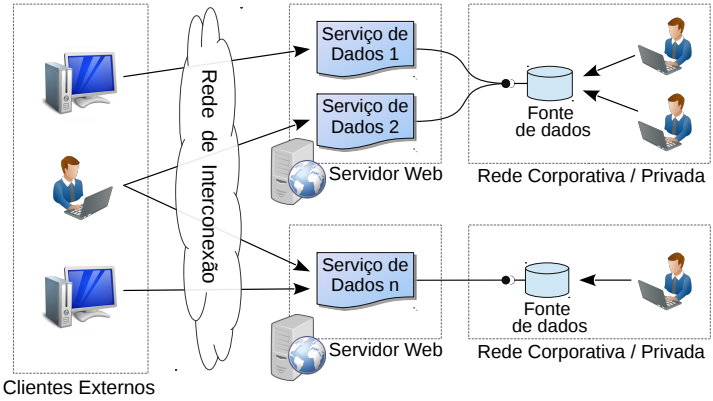
2.2.1 Serviços de dados

Um requisito essencial de aplicações Web que fazem uso intensivo de dados é a capacidade de integração desses dados provenientes da Web. Uma fonte de dados da Web geralmente armazena dados que estão relacionados entre si de acordo com um modelo ou esquema de dados, o qual pode ser desconhecido para os consumidores externos que acessam os dados. Em geral, as fontes de dados são expostas na Web por meio de serviços Web. Os dados, pois, passam a ser providos através de um ou mais serviços – conhecidos como serviços de dados Web, ou simplesmente serviços de dados (BIANCHINI; ANTONELLIS; MELCHIORI, 2015) –, os quais fornecem uma interface Web para lidar com as fontes de dados. A Figura 6 esboça uma visão geral da aplicação de serviços de dados atuando como uma interface de acesso a fontes de dados, pertencentes a uma rede organizacional, para agentes externos.

Bianchini, Antonellis e Melchiori (2015) definem um serviço de dados s como uma operação, método ou consulta para acessar dados de uma determinada fonte de dados. Estes serviços são modelados como um conjunto de: i) entradas de serviço s_i, que consistem em parâmetros que são necessários para invocar o serviço e os dados de acesso; e ii) saídas s_o, representando dados que são acessados através do serviço s.

O acesso a dados é feito através de uma chave de acesso que geralmente se refere a um ou mais recursos, ou seja, um consumidor solicita um recurso para um serviço, passando s_i, e recebe uma representação desse recurso, s_o. A representação de saída pode ser

Figura 6 – Visão geral de aplicação dos serviços de dados.



Fonte: Elaborado pelo autor (2018).

vista como um *snapshot* do estado de um recurso em um determinado momento, disponível em diferentes formatos, como XML, JSON, HTML, etc. Vale ressaltar que os serviços de dados não estão vinculados a nenhuma tecnologia específica; eles podem ser implementados, por exemplo, usando pilhas de tecnologia SOAP ou REST. A principal característica dos serviços de dados, portanto, é fornecer uma abstração de acesso às fontes de dados, atuando como provedores de dados.

Alguns exemplos públicos de serviços de dados são os métodos de Web APIs que fornecem dados oriundos de um domínio específico (por exemplo, o GeoData Demographics API[4]) ou, ainda, serviços que entregam dados em formatos tabulares (como o Google Fusion Tables[5]).

2.2.2 Serviços de dados semânticos

Apesar de os serviços de dados apresentarem uma boa solução para prover acesso e manipulação de dados na Web, os padrões disponíveis não são suficientes para tratar algumas questões de pesquisa, como o intercâmbio de dados legíveis por máquinas, bem como a automatização dos processos de descoberta, seleção e composição de serviços (NACER; AISSANI, 2014; TALANTIKITE; AISSANI; BOUDJLIDA, 2009). Tanto serviços SOAP quanto REST

[4] Disponível em: <http://geodataservice.net>.
[5] Disponível em: <https://developers.google.com/fusiontables/>.

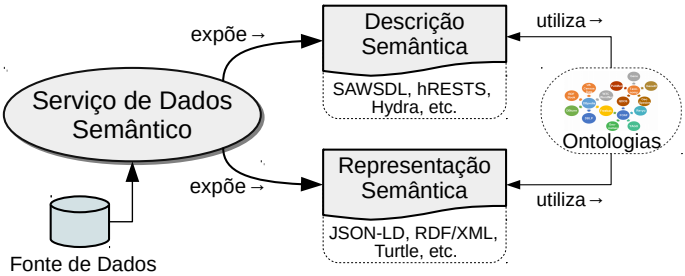
enfrentam dificuldades semelhantes com relação à carência de descrições processáveis por máquinas, visto que as linguagens WSDL e WADL permitem apenas a descrição sintática dos serviços.

Um fator crucial que dificulta a integração de tais serviços, portanto, ocorre no nível conceitual, uma vez que são frequentemente empregadas terminologias distintas (por exemplo, diferentes nomes de atributos), mesmo se tratando de informações sobre o mesmo conceito do mundo real. Para superar tal inconveniente, as tecnologias da Web Semântica (BERNERS-LEE; HENDLER; LASSILA, 2001), tais como ontologias, RDF e *Linked Data* (vistos na Seção 2.1), podem ser aplicadas aos serviços de dados, resultando em serviços de dados semânticos (SDS). Tais serviços surgem como uma forma de reduzir o extenso esforço necessário para manipular os serviços Web tradicionais (sintáticos).

De acordo com McIlraith, Son e Zeng (2001), os serviços semânticos devem expor informações sobre os serviços disponíveis, suas propriedades, interfaces de execução, pré- e pós-condições, em um formato sofisticado legível por máquina. No que diz respeito aos serviços de dados semânticos, os recursos providos também devem estar associados a conceitos semânticos. Por outro lado, Lira et al. (2014) consideram os serviços de dados semânticos como pontos de acesso a dados armazenados nativamente como RDF em uma determinada fonte de dados (conhecida como *triple store*). Em contraste, este trabalho argumenta que os SDS fornecem representações semânticas, independente de como os dados estão armazenados e são mantidos. Assim, enriquecer representações de serviços de dados significa fornecer dados associados a conceitos semânticos definidos em ontologias, utilizando formatos RDF (por exemplo, JSON-LD).

Desse modo, SDS utilizam ontologias com a finalidade de descrever os seus dados e as suas capacidades (por meio de descrições semânticas) e fornecer aos consumidores informações semanticamente enriquecidas, podendo, assim, ser mais facilmente compreendidos por máquinas. A Figura 7 ilustra uma abstração genérica de um serviço de dados semântico expondo descrições e representações que utilizam (referenciam) conceitos definidos em ontologias. As descrições, bem como as representações semânticas, empregam formatos e vocabulários específicos para fornecer informações interpretáveis tanto por agentes de *software* quanto por humanos.

Figura 7 – Serviço de dados semântico.



Fonte: Elaborado pelo autor (2018).

2.2.2.1 Descrições semânticas

A introdução de semântica na descrição de um serviço pode ser realizada por meio de anotações. Uma anotação atribui a um determinado recurso um link para sua descrição semântica, que referencia uma ontologia. O objetivo é ter dados definidos e vinculados de tal forma que seu significado torna-se explicitamente interpretável por *software*, em vez de inteligível apenas por pessoas (NACER; AISSANI, 2014). Há algumas propostas voltadas à descrição de serviços semânticos. Pode-se citar como exemplo a OWL-S (OWL *for Services*), uma linguagem baseada na OWL para descrever anotações semânticas (MARTIN et al., 2007); WSMO (*Web Service Modeling Ontology*), uma ontologia direcionada a descrever vários aspectos relacionados a serviços semânticos (ROMAN et al., 2005); SAWSDL (*Semantic Annotations for* WSDL *and* XML *Schema*), para permitir que anotações semânticas sejam adicionadas a partes da descrição do serviço, como as estruturas de entrada/saída, interfaces e operações (KOPECKÝ et al., 2007); e hRESTS, visto como um microformato HTML (adaptações na semântica HTML para facilitar a indexação e extração de informações) para descrever serviços REST (KOPECKÝ; GOMADAM; VITVAR, 2008).

Um vocabulário para descrição de serviços Web que vem ganhando atenção tanto na indústria quanto na academia é o Hydra (LANTHALER; GÜTL, 2013). Introduzido por Markus Lanthaler, um dos idealizadores do JSON-LD, o Hydra busca simplificar o desenvolvimento de descrições de Web APIs fazendo uso dos benefícios providos pelo *Linked Data*. Uma vantagem do Hydra é a possibilidade de especificar uma série de conceitos comumente utilizados nos serviços, dando ensejo à criação de clientes genéricos de Web APIs.

Figura 8 – Exemplo de operações suportadas descritas com Hydra.

```
1  {
2    "@context": {
3    "hydra":
       ↪   "http://www.w3.org/ns/hydra/context.jsonld",
4    "foaf":  "http://xmlns.com/foaf/0.1/"
5    },
6    "@id":
       ↪   "http://servico-exemplo.com/doc/#comments",
7    "@type": "hydra:Class",
8    "hydra:supportedOperation": [
9    {
10         "@type": "Operation",
11         "title": "Creates a new entity",
12         "method": "POST",
13         "expects": "foaf:Person",
14         "returns": "foaf:Person"
15   }
16   {
17         "@type": "Operation",
18         "title": "Deletes an entity",
19         "method": "DELETE"
20   }
21   ]
22 }
```

Fonte: Elaborado pelo autor (2018).

Tais clientes passam a ter, por meio de um documento JSON-LD, as informações necessárias para efetuar requisições aos serviços.

O Hydra adota o conceito de classes para descrever os recursos (identificados por um URI), contendo as suas propriedades e operações suportadas (LANTHALER; GÜTL, 2013). Cada operação informa também o método HTTP que deve ser utilizado. A Figura 8 ilustra um exemplo de descrição das operações suportadas por um serviço utilizando o Hydra. A linha 8 apresenta as operações com métodos `POST` e `DELETE`. O elemento `expects` (linha 13) determina o tipo de dado que deve ser enviado pelo cliente ao executar a operação. Já o tipo do recurso a ser retornado é indicado pelo elemento `returns` (linha 14).

Às vezes, é inviável para um provedor de serviço de dados construir um URL de acesso a um determinado recurso. Um caso comum são URLs que permitem ao cliente consultar um dado especificando os valores de determinados atributos (*e.g.*, um identificador único). Nesse caso, o URL depende de informações conhecidas apenas pelo cliente, não sendo possível fornecer previamente a URL completa. Uma estratégia para superar tal questão é prover ao cliente um modelo para construir o URL em tempo de execução. O Hydra suporta tal funcionalidade por meio da classe *IriTemplate*, a

qual utiliza os conceitos de um *URITemplate* (GREGORIO et al., 2012) para mapear propriedades a um modelo (*template*) de URL.

Um *URITemplate* permite especificar um URL que inclui parâmetros que devem ser substituídos antes de acessá-la. A sintaxe utilizada geralmente emprega as chaves ({ }) para identificar um parâmetro a ser substituído, como no exemplo: `http://servico-exemplo/query?nome={nomeCompleto}&id={cpf}`. Tanto `{nomeCompleto}` quanto `{cpf}` devem ser substituídos pelos valores dos parâmetros `nome` e `id`, respectivamente, para se construir o URI dereferenciada (do inglês, *dereferenced*) e acessar o recurso desejado. *URITemplates* fornecem um mecanismo eficaz para trabalhar com URIs que compartilham uma estrutura comum, tornando o Hydra uma tecnologia interessante e promissora para descrever semanticamente serviços de dados, especialmente aqueles que adotam (parcial ou totalmente) o estilo arquitetural REST.

Tanto o Hydra quanto as demais tecnologias supracitadas fornecem uma base para construir descrições semânticas de serviços. O Hydra ainda não é considerado um padrão *de facto*, mas vale ressaltar que muitos serviços disponíveis atualmente tampouco fornecem uma descrição legível por máquinas, a exemplo das APIs do Twitter[6]. Igualmente, o Facebook fornece a descrição de seus serviços Web (Graph API) por meio de documentações legíveis apenas por humanos. Há uma ferramenta, Graph Explorer[7], que permite ao usuário navegar visualmente nos métodos da Graph API. Entretanto, os URLs das requisições ainda são criados manualmente pelo usuário. Portanto, ainda há muito a ser explorado no âmbito de descrição semântica de serviços Web, sobretudo em pesquisas focadas em automatizar a sua construção, com o intuito de facilitar e fomentar o seu uso.

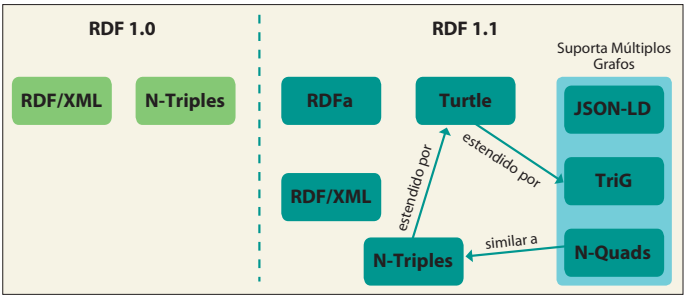
2.2.2.2 Representações semânticas

Existem diferentes formas de representar/serializar recursos fornecidos por serviços Web, como o XML, JSON, HTML, etc. Representações semânticas usualmente utilizam o modelo de dados RDF (conforme visto na seção 2.1.2). A atual versão do RDF 1.1 ampliou significativamente a quantidade de formatos de serialização em comparação à versão 1.0, conforme exibido na Figura 9. Um conceito interessante adicionado à versão 1.1 do RDF é o de

[6] Exemplo de descrição dispnoível em: <https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user_timeline.html>.

[7] Disponível em: <https://developers.facebook.com/tools/explorer>.

Figura 9 – Representações disponíveis nas versões 1.0 e 1.1 do RDF.



Fonte: Isotani e Bittencourt (2015).

múltiplos grafos. Com essa nova funcionalidade é possível adicionar outros grafos ao grafo original de um documento RDF. Com isso, produz-se conjuntos de dados conectados, os quais podem ser acessados por um único IRI (ISOTANI; BITTENCOURT, 2015).

O RDF/XML, primeira serialização desenvolvida para RDF, fornece uma sintaxe XML para se construir grafos RDF. O N-Triples é o formato de serialização mais simples e intuitivo que existe, possuindo uma estrutura de `<sujeito> <predicado> <objeto>`, onde cada linha de um código N-Triples equivale a uma tripla RDF. O RDFa (*Resource Description File in Attributes*) adiciona uma série de atributos às tags HTML. Com o RDFa é possível disponibilizar um arquivo HTML contendo diversos atributos de um documento RDF. O Turtle foi criado para ampliar as possibilidades de descrição de um documento N-Triples, permitindo, por exemplo, descrever prefixos e IRIs relativos na estrutura do documento. Já o formato N-Quads trata-se também de uma extensão do N-Triples, utilizado para o intercâmbio de datsets, e o TriG estende o Turtle para permitir a representação de múltiplos grafos.

Por fim, o JSON-LD, um dos mais recentes formatos de serialização, surgiu como uma extensão do JSON, sendo bastante intuitivo para programadores já familiarizados com esta sintaxe. Um exemplo de representação em JSON-LD é ilustrado na Figura 3 (b). Como já discutido na seção 2.1.2, o JSON-LD é uma maneira leve de descrever dados conectados, permitindo a inclusão de informações semânticas em documentos JSON existentes. Outro ponto interessante do JSON-LD é a sua compatibilidade com o JSON, o que permite que as bibliotecas e os analisadores (*parsers*) sejam reaproveitados pelos consumidores.

Destarte, este trabalho emprega o JSON-LD como principal formato das representações semânticas fornecidas por SDS. Entretanto, vale destacar que os diversos formatos RDF disponíveis podem ser convertidos de um para outro sem muita dificuldade por meio de bibliotecas e *frameworks* existentes – o Apache Jena[8], por exemplo, manipula modelos de dados RDF como objetos Java e permite serializar os modelos em qualquer formato da Figura 9.

2.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Na primeira seção deste capítulo foram discutidos os principais conceitos atinentes a dados abertos conectados, ao modelo de dados RDF e às principais classificações de ontologias. Foram apresentados ainda os elementos básicos para se construir uma ontologia por meio da OWL e visto como a área de *Ontology Learning* facilita o processo de construção de ontologias. Foi destacada também a importância de técnicas de *matching* para se obter alinhamentos entre ontologias heterogêneas e facilitar a interoperabilidade dos sistemas computacionais.

Na segunda seção foram introduzidas as principais tecnologias no âmbito de serviços Web e apresentado o conceito de serviço de dados como um tipo especializado de serviço dedicado a fornecer uma interface Web de acesso e manipulação a fontes de dados. Foi visto como tecnologias da Web Semântica podem ser incorporadas a serviços Web de modo a auxiliar processos como descoberta, seleção, composição de serviços, entre outros. É no esteio desta fundamentação teórica que o capítulo seguinte explora o estado da arte relacionado a esta pesquisa e discute os principais trabalhos contemplados em ambas as áreas apresentadas neste capítulo.

[8] Disponível em: <https://jena.apache.org/>.

3 ESTADO DA ARTE

Neste capítulo são discutidas as propostas encontradas na literatura relevantes a esta pesquisa. A seção 3.1 introduz a revisão da literatura realizada e apresenta os principais aspectos considerados para a análise dos trabalhos relacionados. A seção 3.2 discute as abordagens mais representativas encontradas no âmbito da construção e *matching* de ontologias, bem como de enriquecimento semântico de serviços Web. Por fim, a seção 3.3 sumariza a discussão deste capítulo, pontuando possibilidades de melhoria nas propostas existentes e oportunidades de pesquisa na área.

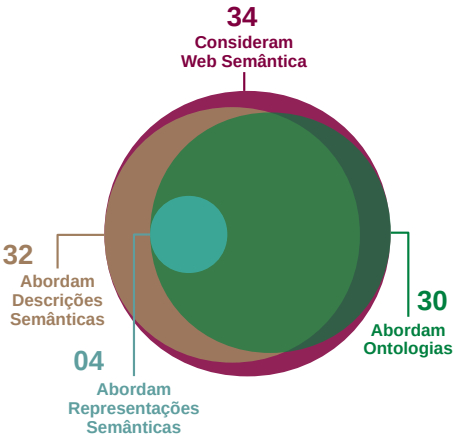
3.1 REVISÃO DA LITERATURA

A partir de uma Revisão Sistemática da Literatura (RSL) realizada de forma colaborativa, foi identificado como as tecnologias da Web Semântica são utilizadas em *Web Services* no contexto de composição, seleção, descoberta e descrição de serviços (consultar Apêndice B para mais detalhes). Tal RSL ajudou a entender a importância da semântica no âmbito desses processos e a identificar nos trabalhos selecionados como a semântica é adotada nesse contexto. Com base na análise dos trabalhos de revisão (*surveys*) selecionados, foi possível obter um conjunto de trabalhos primários, citados pelos artigos, relacionados com o tema desta dissertação.

Dentre os 48 trabalhos selecionados ao final da RSL que abordam a temática de composição de serviços Web, 34 mencionam a adoção de tecnologias da Web Semântica. Destes 34, 31 trabalhos utilizam semântica para descrever as interfaces dos serviços, enquanto os outros três utilizam a semântica para outros propósitos, como, por exemplo, para modelagem de processos utilizando BPEL (*Business Process Execution Language*). Dentre os 32 trabalhos que utilizam a semântica para descrição de serviços, apenas quatro abordam também o enriquecimento das representações fornecidas por serviços de dados, conforme ilustrado na Figura 10. O Apêndice B sumariza os trabalhos que abordam serviços Web semânticos em uma tabela comparativa.

De acordo com Tosi e Morasca (2015), o uso de ontologias e ferramentas para anotar serviços Web tem se tornando uma questão fundamental para ajudar a difundir serviços Web semânticos. Dentre os 34 artigos que abordam serviços Web semânticos, 30 mencionam a construção e/ou uso de ontologias (sejam ontologias de domínio específico, que descrevem os dados, ou ontologias voltadas

Figura 10 – Trabalhos da RSL que abordam semântica.



Fonte: Elaborado pelo autor (2018).

para a descrição de serviços, como OWL-S e WSMO) como um aspecto importante a ser considerado quando do enriquecimento semântico das capacidades do serviço (descrição) e dos seus dados (representações). As ontologias tornaram-se o artefato de modelagem *de facto*, sendo empregadas em uma variedade de aplicações, proeminentemente na Web Semântica. No entanto, é consenso entre os autores que a construção de ontologias de forma prática e menos dispendiosa continua a ser uma tarefa desafiadora.

Embora as ontologias sejam úteis para aprimorar o conhecimento do domínio no qual está inserido um serviço Web, Nacer e Aissani (2014) argumentam que, para alcançar efetivamente a interoperabilidade e a integração de dados, ainda há o desafio principal de reutilizar os conceitos semânticos por meio de alinhamentos. Além de permitirem a integração e a conexão de dados semânticos, os alinhamentos são fundamentais para relacionar conceitos, evoluir ontologias e unificá-las (EUZENAT; SHVAIKO, 2007). Com isso, as técnicas de *matching* de ontologias, além de outras vantagens, favorecem a área de serviços semânticos quando da automatização dos processos de composição, descoberta e seleção de serviços.

As conclusões resultantes da RSL deram subsídio a um novo levantamento bibliográfico dedicado a buscar na literatura abordagens e técnicas adicionais tratando exclusivamente de enriquecimento semântico automático de serviços de dados. Com isso, novas

abordagens, além das encontradas nos trabalhos primários oriundos da RSL, foram incorporadas ao estudo da arte deste trabalho. A Tabela 1 apresenta os trabalhos mais representativos que possuem alguma correlação com a proposta desta dissertação, totalizando 21 artigos de conferências e periódicos. Alguns critérios de comparação foram explorados para fins de análise com este trabalho, como:

- a construção de ontologias, determinando se a abordagem oferece suporte para construir de forma (semi-)automática uma ontologia de domínio;
- a adoção de técnicas de *matching* de ontologias;
- o enriquecimento da representação, ou seja, se a abordagem oferece suporte para geração de representações fornecidas por serviços, utilizando o modelo de dados RDF;
- o enriquecimento da descrição, isto é, se a abordagem fornece suporte para gerar uma descrição semântica para serviços Web ou, ainda, enriquecer uma descrição sintática já existentes com conceitos semânticos; e
- se a abordagem proposta é automática, e sendo assim não necessita da intervenção de um usuário especialista no domínio.

Duas conclusões podem ser obtidas da tabela. Primeiro, nenhuma das propostas encontradas na literatura fornece suporte a todos os cinco critérios elencados. Além disso, a despeito de muitos autores direcionarem esforços no enriquecimento (semi-)automático de serviços Web (TOSI; MORASCA, 2015), pouca atenção é dada ao enriquecimento semântico de representações fornecidas por serviço Web. Grande parte das propostas encontradas na literatura foca em enriquecer semanticamente as descrições do serviço.

As principais abordagens encontradas nos trabalhos relacionados são discutidas a seguir.

3.2 DISCUSSÃO DOS TRABALHOS RELACIONADOS

Os trabalhos selecionados para revisão foram classificados em dois grupos, embora haja uma intersecção entre eles: i) os que tratam de construção e *matching* de ontologias, apresentados na Seção 3.2.1; e ii) os que propõem abordagens para enriquecimento semântico de serviços, descritos na Seção 3.2.2. Este último grupo de trabalhos pode ainda ser dividido em dois subgrupos: enriquecimento

da descrição de serviços, e enriquecimento das representações fornecidas por serviços de dados. O primeiro se concentra em adicionar informação semântica para descrever as interfaces dos serviços, enquanto o segundo busca enriquecer as representações com conceitos semânticos de modo a prover dados conectados.

Tabela 1 – Comparativo dos trabalhos relacionados a esta pesquisa.

Trabalho	**Construção de Ontologia**	***Matching* de Ontologia**	**Enriq. da Representação**	**Enriq. da Descrição**	**100% Automática**
David, Guillet e Briand (2007)		✓			✓
Suchanek, Abiteboul e Senellart (2011)		✓			✓
Tran, Ichise e Ho (2011)		✓			✓
Zapilko e Mathiak (2014)		✓			✓
Nunes et al. (2013)		✓			✓
Salem e AbdelRahman (2010)	✓				
Segev e Sheng (2012)	✓				
Nguyen e Lu (2016)	✓				
Bravo, Rodríguez e Pascual (2014)	✓			✓	
Cremaschi e Paoli (2017)				✓	✓
Cao, Falleri e Blanc (2017)				✓	✓
Mohr e Walther (2014)				✓	
Rathinam, Parthiban e Mariakalavathy (2014)				✓	✓
Lin et al. (2016)				✓	✓
Duo, Juan-Zi e Bin (2005)		✓		✓	✓
Zhang, Chen e Feng (2013)				✓	✓
Schwichtenberg, Gerth e Engels (2017)	✓	✓		✓	
Saquicela, Vilches-Blazquez e Corcho (2011)		✓		✓	✓
Yao et al. (2014)	✓		✓		
Freire, Freire e Souza (2017)		✓	✓		
Salvadori et al. (2017a)		✓	✓		✓
OntoGenesis	✓	✓	✓	✓	✓

3.2.1 Construção e *matching* de ontologias

Conforme discutido na seção 2.1.3.2, Euzenat e Shvaiko (2007) identificam 4 grupos de abordagens de *matching* de ontologias, cada uma com desafios específicos no cenário de enriquecimento semântico de serviços. As técnicas baseadas em nomes exigem que os elementos da ontologia sejam rotulados no mesmo idioma e tenha ainda elementos rotulados com valores semelhantes. As técnicas baseadas em semântica e as baseadas em estrutura analisam as características da ontologia, como, por exemplo, a hierarquia de classes e as relações existentes entre as classes.

A técnica que mais se assemelha ao problema atacado nesta pesquisa é o *matching* extensional, a qual pode ser adaptada para o enriquecimento de serviços, conforme visto em Salvadori et al. (2017a). Tais técnicas, no entanto, buscam identificar correspondências entre duas ontologias a partir do compartilhamento dos seus indivíduos (DAVID; GUILLET; BRIAND, 2007; SUCHANEK; ABITEBOUL; SENELLART, 2011). Por exemplo, o AROMA, proposto por David, Guillet e Briand (2007), produz alinhamentos entre ontologias quando os seus indivíduos são equivalentes. Caso os indivíduos não compartilhem pelo menos duas propriedades, o alinhamento não é satisfatório. Tal ocorrência é comum quando os dados obtidos através da interface do serviço não estão relacionados com os dados utilizados em fontes externas utilizadas para realizar o enriquecimento, ou seja, os seus indivíduos não são os mesmos. Antagonicamente, além de construir dinamicamente ontologias de domínio, abordagem defendida neste trabalho permite que correspondências sejam feitas no nível de propriedades, isto é, realiza o *matching* das ontologias com base nos valores de suas propriedades, e não na equivalência do indivíduo como um todo. À luz de tais evidências, evidencia-se que os algoritmos de *matching* extensional apresentam características distintas do problema que enfrentamos, mais especificamente *matching* de propriedades.

Por outro lado, há poucos trabalhos que focam especificamente no *matching* de propriedades de ontologias. Tran, Ichise e Ho (2011) propõem um sistema de agregação de similaridade para realizar o *matching* de ontologias, o qual emprega algumas medidas de similaridade para alinhar propriedades de objetos (*object properties*) baseados em seus *domains* e *ranges*. O sistema é divido em duas partes principais: na primeira parte é realizado o cálculo e combinação de diferentes medidas de similaridade; e a segunda parte busca extrair o alinhamento com base no cálculo das medidas

de similaridade. O sistema calcula primeiro cinco medidas básicas diferentes para criar cinco matrizes de similaridade, como uma medida de similaridade baseada na distância entre *strings*, medida de similaridade baseada na base de sinônimos do WordNet, etc. Para cada medida é atribuído um peso e, com base nesse peso, é obtida uma matriz de similaridade. Ao final, é realizado um processo de refinamento e as matrizes são combinadas de modo a produzir uma única matriz final, por meio da qual são obtidos os alinhamentos. A avaliação do sistema foi feita sob ontologias reais. No entanto, os autores afirmam que os resultados dos alinhamentos das propriedades ainda não são suficientemente bons.

Zapilko e Mathiak (2014) identificam a relação exata entre dois objetos oriundos de um *dataset* de dados conectados em larga escala, usando dados governamentais. O objetivo da abordagem proposta pelos autores é alinhar as instâncias das ontologias separadamente e calcular um valor (chamado de *score*, que utiliza variações do coeficiente Jaccard) da sobreposição entre as instâncias para melhorar o *matching* das propriedades de objetos. Uma desvantagem das abordagens propostas por Tran, Ichise e Ho (2011) e por Zapilko e Mathiak (2014) é que ambas estão limitadas às propriedades do objeto.

Nunes et al. (2013), por outro lado, utilizam algoritmos genéticos para o *matching* de propriedades de dados (*datatype properties*). Essa abordagem, todavia, foca em encontrar relações complexas de um para muitos. Em outras palavras, os autores estão interessados em mapear propriedades que são compostas por outras propriedades (por exemplo, mapeando *first name* e *last name* para *full name*).

Como visto no Capítulo 2, algumas metodologias para construção de ontologias podem ser encontradas na literatura (GRÜNINGER; FOX, 1995; USCHOLD, 1996; FERNÁNDEZ-LÓPEZ; GÓMEZ-PÉREZ; JURISTO, 1997). Por ainda necessitarem de um árduo trabalho manual e, sobretudo, serem de difícil compreensão por parte dos engenheiros de ontologias e desenvolvedores de *software*, tais metodologias foram pouco adotadas por estes grupos, dando ensejo ao surgimento de ferramentas e abordagens que visam a automatização do processo de construção de uma ontologia.

Salem e AbdelRahman (2010) estendem a ferramenta Text2Onto (CIMIANO; VÖLKER, 2005) com o objetivo de propor um novo construtor de ontologias que busca extrair ontologias de domínio específico a partir de um corpus textual de múltiplos domínios. Para isso, a abordagem constrói uma ontologia de domínio

para cada conjunto de conceitos relacionados extraídos do texto.

Segev e Sheng (2012) propõem uma abordagem para construção de ontologias baseada nas descrições de serviços Web. A abordagem parte da premissa que os serviços usualmente consistem de uma descrição textual, que descreve o que o serviço faz, e uma descrição WSDL, que descreve como acessá-los. Assim, o método proposto pelos autores utiliza técnicas de recuperação da informação e processamento de linguagem natural para extrair os termos mais relevantes dos documentos dos serviços, bem como das descrições WSDL. Com isso, são identificados os principais conceitos, os quais são utilizados para construir a ontologia.

Nguyen e Lu (2016) apresentam um novo método para construir uma ontologia de domínio de páginas da Web. A abordagem se concentra em soluções semi-automáticas para construção de ontologias, usando para isso dados da Web e focando essencialmente no aprimoramento dos resultados de sistemas de recomendação Web que utilizam ontologias.

Não obstante, todas as abordagens encontradas sofrem com alguns inconvenientes. Em primeiro lugar, a maioria delas depende de modelos de ontologias proprietários ou muito específicos, o que dificulta a sua ampla aplicabilidade. Adicionalmente, tais abordagens não são totalmente automáticas, posto que se concentram em auxiliar usuários especialistas do domínio durante o desenvolvimento de uma ontologia. Por fim, os métodos tradicionais de construção de ontologias geralmente exigem como entrada um enorme conjunto de texto não estruturado (SALEM; ABDELRAHMAN, 2010; LEE et al., 2007) ou dados originários de páginas Web (NGUYEN; LU, 2016). Logo, tais pesquisas carecem de soluções que suportem a construção e evolução de ontologias com base em informações fornecidas sob demanda, como os dados obtidos por meio de interfaces de serviços Web.

3.2.2 Enriquecimento semântico de serviços Web

Embora os benefícios de se empregar serviços semânticos já estejam bem documentados, tanto na academia quanto na indústria (BACHLECHNER; FINK, 2008; BRUIJN et al., 2005; BLAKE et al., 2012), a real adoção e implementação de tais serviços ainda é deficiente. Muitos dos desafios que contribuem para essa baixa adoção já foram identificados em trabalhos anteriores (NACER; AISSANI, 2014; BACHLECHNER; FINK, 2008; BARROS et al., 2011). Alguns problemas comuns incluem a carência de ferramentas efetivas,

o alto custo de se adotar SOA e tecnologias da Web Semântica, a curva de aprendizado íngreme e a falta de consenso acerca dos padrões de modelagem semântica. A seguir são discutidos os principais trabalhos encontrados que buscam enriquecer semanticamente atributos de serviços Web, como a sua descrição e suas representações.

Alguns autores fornecem suportes ferramentais para auxiliar a descrição semântica de serviços Web. O SDWS (*Semantic Description of Web Services*), por exemplo, proposto por Bravo, Rodríguez e Pascual (2014), se destaca por gerar descrições semânticas baseadas em uma coleção de outros serviços Web. As descrições são geradas a partir da construção de um modelo ontológico genérico para integrar serviços descritos em diferentes linguagens. Outras ferramentas focadas na geração de descrição semânticas de serviços Web podem ser encontradas na literatura, como SWS Editor (*Semantic Web Services Editor*) (LIRA; CAETANO, 2016), ASSARS (*Automated Structural Semantic Annotation for RESTful Services*) (LUO et al., 2016) e SWEET (*Semantic Web sErvices Editing Tool*) (MALESHKOVA; PEDRINACI; DOMINGUE, 2009). Não obstante, tais ferramentas auxiliam o usuário no processo de anotação semântica apenas em descrições de serviços, fornecendo, assim, mecanismos semi-automáticos de enriquecimento.

Cao, Falleri e Blanc (2017) apresentam uma abordagem automática para extração de *URI Templates*, a qual analisa linguagem natural em páginas HTML com o intuito de produzir uma descrição para um determinado serviço REST. Para gerar a descrição, os autores se valem da OpenAPI[1], uma especificação para criar interfaces legíveis por máquinas de modo a descrever, produzir, consumir e visualizar serviços Web RESTful. Essa é uma abordagem automática interessante, visto que pode ser usada como uma caixa preta que recebe como entrada apenas a raiz de um URL e produz uma descrição do serviço contemplando os devidos *URI Templates*. Todavia, os autores não consideram a inclusão de semântica nos parâmetros dos URIs extraídos.

Cremaschi e Paoli (2017) propõem uma abordagem cujo objetivo é automatizar a criação de descrições semânticas para serviços Web a fim de aprimorar a interoperabilidade e facilitar a composição automática de serviços. O método proposto busca coletar dados de amostras de APIs existentes e associá-los a conceitos apropriados advindos de ontologias compartilhadas. A descrição gerada

[1] Especificação OpenAPI: <https://www.openapis.org/>.

utiliza uma extensão da especificação OpenAPI/Swagger[2] desenvolvida pelos próprios autores, a qual define novos elementos na descrição do serviço para suportar anotações semânticas. A especificação OpenAPI, portanto, é enriquecida adicionando duas novas propriedades: (i) `classAnnotation` para manter as anotações relacionadas à entidade do dado coletado; e (ii) `propertyAnnotation` para armazenar as anotações que representam as relações entre as entidades. Assim, as anotações semânticas incluídas na descrição assumem a forma de URIs que identificam os conceitos e as relações nas ontologias compartilhadas. Embora a abordagem considere dados amostrais e enriqueça a descrição, o serviço continua fornecendo dados sintáticos.

O trabalho de Mohr e Walther (2014) propõe uma abordagem baseada em templates para a geração de serviços Web com suporte semântico. Um template deve ser composto de uma descrição (a qual deve conter entradas, saídas, pré- e pós-condições), um determinado *workflow* descrito semanticamente e um conjunto de restrições com expressões lógicas. Após a construção do template por um especialista, são produzidas tanto a descrição semântica quanto a implementação de novos serviços. Exige-se, portanto, um trabalho ainda manual e profundo conhecimento da abordagem proposta e de lógica de primeira ordem para construção dos templates. Ademais, a corretude da implementação do serviço, bem como a sua descrição semântica, está fortemente acoplada ao template gerado.

Alguns trabalhos direcionam esforços em mecanismos automáticos para extrair semântica de serviços Web utilizando métodos de categorização de serviços (RATHINAM; PARTHIBAN; MARIAKALAVATHY, 2014; LIN et al., 2016). Nesse sentido, o objetivo é classificar semanticamente os serviços por termos similares. Para tanto, são extraídas informações do perfil do serviço (o qual contém o histórico comportamental dos serviços, ou seja, como esses serviços são usados e quem os invocou) e de sua descrição. Com base nessas informações, são inferidas as relações semânticas entre os serviços a fim de obter o conhecimento semântico relevante. O método proposto por Lin et al. (2016) aplica a categorização dos serviços e o processamento de suas equivalências utilizando apenas as informações básicas do serviço. No entanto, não são apresentas quais ontologias de domínio são utilizadas para realizar a anotação semântica. Rathinam, Parthiban e Mariakalavathy (2014), por sua vez, utilizam o *WordNet Ontology Framework* e SUMO (*Suggested*

[2] OpenAPI: <https://www.openapis.org/>.

Upper Merged Ontology) para auxiliar na inferência semântica e, ao final, classificam um conjunto de descrições WSDL em categorias já predefinidas (como educação, finanças, científica, etc.). Apesar de a categorização semântica facilitar a descoberta de serviços relevantes a uma determinada requisição, as propostas descritas não consideram os recursos geridos pelos serviços, sobretudo quando se trata de serviços de acesso a dados. Ademais, as abordagens são executadas *offline*, independentemente das requisições feitas para os serviços.

Uma abordagem automática que incorpora mecanismos de mapeamento de ontologia é proposta por Duo, Juan-Zi e Bin (2005). Os autores definem diversas regras para traduzir um esquema XML de um serviço Web em uma ontologia OWL, a qual é utilizada para realizar um mapeamento com outras ontologias já existentes. Com base no resultado gerado do mapeamento, uma descrição OWL-S é construída para o serviço. Embora utilizem técnicas de *matching* de ontologias para enriquecimento semântico de estruturas sintáticas, os autores não consideram os indivíduos de cada ontologia de modo a aplicar o *matching* extensional. O método de *matching* aplicado se baseia apenas em semelhanças estruturais morfológicas entre duas ontologias.

O método proposto por Zhang, Chen e Feng (2013) busca unir os URIs existentes no DBpedia com parâmetros de serviços SOAP. Para alcançar esse resultado, a abordagem analisa a descrição WSDL do serviço – na qual é feita a validação e recuperação dos elementos necessários (interfaces e parâmetros) do serviço. Em seguida, cada parâmetro é refinado com o intuito de regularizar o seu nome, visto que estes muitas vezes são definidos de forma irregular (ou seja, com abreviações e sem espaçamento entre as palavras). Por fim, consulta-se na ontologia do DBpedia conceitos similares aos dos parâmetros já refinados, além de recuperar também os dados (instâncias) existentes no DBpedia que possuam alguma correspondência a um parâmetro do serviço. Como há casos em que instâncias do DBpedia não estão necessariamente associadas a um conceito de sua própria ontologia, a abordagem utiliza técnicas de mapeamento das ontologias para obter o conceito com maior similaridade semântica, de modo a utilizá-lo para anotar os parâmetros de entrada e saída de uma descrição WSDL. Não obstante, as representações fornecidas pelo serviço não são efetivamente enriquecidas. Além disso, a anotação de saída considera apenas a assinatura do método (ou operação) descrita no WSDL. Nesse caso, supõe-se que o nome da operação possui a estrutura $getAbyB$, sendo A o retorno da operação em questão. Por exemplo, para a operação

getWeatherByPlaceNameResult, é extraído *Weather* como sendo o objeto de retorno, para então aplicar a anotação. A extração, pois, sustenta-se na hipótese de que os nomes de operações obedecerão o padrão preestabelecido.

Em um estudo de caso realizado por Schwichtenberg, Gerth e Engels (2017), descrições (mencionadas pelos autores como especificações) semânticas OWL-S são criadas de forma semi-automática a partir de especificações Open API puramente sintáticas. A abordagem proposta extrai ontologias de domínio a partir de esquemas JSON, que fazem parte das especificações sintáticas. A ontologia extraída é, então, alinhada com uma ontologia global previamente definida. Embora os autores utilizem técnicas de *matching* para reutilizar conceitos já definidos em uma ontologia global, os autores não mencionam qual *matcher* foi utilizado no estudo de caso. Ademais, a abordagem estabelece links entre as especificações sintática e a semântica, formando o que os autores denominam de *service grounding*.

Saquicela, Vilches-Blazquez e Corcho (2011) propõem um método automático para anotar parâmetros de serviços Web geoespaciais com base em recursos externos, como o repositório de ontologias do DBpedia[3] e serviços de sinônimos[4] e de sugestões de termos[5]. Inicialmente, um especialista do serviço deve informar ao sistema um URI válido contendo os parâmetros do serviço a ser enriquecido. O sistema invoca o serviço com base no URI e analisa a resposta para gerar uma nova descrição sintática. Com base nessa descrição, o sistema consulta uma amostra de classes e propriedades da ontologia do DBpedia. Havendo uma correspondência exata dos conceitos da ontologia com os elementos sintáticos, utilizam-se esses conceitos a fim de buscar outras instâncias que forneçam informações geoespaciais, como, por exemplo, latitude ou longitude. As instâncias das classes que correspondem exatamente aos parâmetros do serviço Web são registradas e armazenadas em um repositório local. No caso de parâmetros para os quais não foram encontradas correspondências exatas, buscam-se sinônimos ou sugestões alternativas dos termos para consultá-los novamente no DBpedia com o intuito de obter novas correspondências. Embora esse mecanismo de anotação semântica se aplique de maneira automática, a abordagem gera como saída apenas uma anotação semântica, não enriquecendo

[3] Ontologia do DBpedia: <http://wiki.dbpedia.org/Ontology>.
[4] Base Synonyms: <http://www.synonyms.net>.
[5] Yahoo Boss: <https://developer.yahoo.com/boss/search/boss_guide/Spelling_Suggest.html>.

o serviço em si. Além disso, a proposta se limita a um domínio específico, tratando exclusivamente dados geoespaciais.

Yao et al. (2014) propõem um *framework* cujo objetivo é transformar dados semi-estruturados – especificamente um conjunto de documentos no formato JSON fornecidos por serviços Web – em documentos semanticamente enriquecidos. Para tanto, os elementos do JSON são, inicialmente, convertidos em triplas RDF similares ao Turtle. Em seguida, é feito um mapeamento semântico para construir ontologias e instâncias baseadas nos metadados dos documentos. O passo posterior é o enriquecimento semântico, no qual são identificadas declarações e restrições de elementos da ontologia e adicionados rótulos e comentários de acordo com regras preestabelecidas. Por fim, o *framework* contempla uma fase de *ontology merging*, em que são considerados outros *datasets* para identificar relacionamentos entre conceitos de diferentes ontologias visando à construção de uma ontologia unificada. Apesar de demonstrar ser um estudo interessante para o enriquecimento semântico de dados semi-estruturados, os autores não endereçam o enriquecimento automático de serviços Web, nem consideram a geração de descrição de serviço. Além disso, o *framework* considera apenas JSON como entrada e gera apenas uma ontologia unificada, a qual deve ser, ao fim do processo, validada por um usuário especialista. Outro inconveniente é que o artigo não fornece os dados utilizados nos experimentos, o que dificulta a sua replicação e futuras comparações com essa abordagem.

Freire, Freire e Souza (2017) apresentam uma abordagem para converter documentos JSON em dados RDF que utilizam ontologias de domínio. Os autores fornecem um suporte ferramental que implementa a abordagem proposta e que adota um *matcher* de ontologia, o qual produz correspondências de equivalências entre as propriedades dos objetos JSON e os termos da ontologia do domínio específico relacionado ao *dataset* JSON de entrada. A ferramenta extrai os metadados do JSON e identifica os tipos mais apropriados das entidades dos objetos do JSON. Por fim, a ferramenta executa o *matcher* para produzir termos equivalentes a propriedades de uma ontologia de domínio e realiza o processo de conversão dos dados em RDF. Nos experimentos é utilizada uma medida de confiança da força da correspondência obtida pelo *matcher*, também chamada de *threshold*, com valor entre 0 e 1. No trabalho, foi usado o valor $0,8$ de modo a considerar apenas as equivalências cuja a força seja superior a esse *threshold*. Os autores, no entanto, não consideram o enriquecimento do serviço Web que fornece os documentos JSON.

Ademais, o alinhador utilizado (DAVID et al., 2011) emprega a técnica baseada em nomes, ou seja, as equivalências são geradas com base em atributos linguísticos das propriedades, não considerando os indivíduos (valores associados às propriedades).

Um método de composição que explora o potencial de interseção de dados observados em descrições de *microservices* de dados é proposto por (SALVADORI et al., 2017a). O método proposto pelos autores se concentra na criação de *links* entre recursos semânticos, de modo que as representações fornecidas por *microservices* sejam, assim, enriquecidas com *links* `owl:sameAs` e `rdfs:seeAlso`. Além disso, os autores propõem um *framework* chamado Alignator, o qual emprega técnicas de *matching* de ontologias para identificar equivalências entre os conceitos de ontologias distintas que descrevem diferentes *microservices*. Entretanto, tal abordagem considera apenas os serviços que i) já adotam uma ontologia de domínio previamente definida e ii) já fornecem representações de dados conectados. Ademais, as descrições semânticas dos serviços já devem ter sido criadas. Apesar de ser uma abordagem que foca essencialmente no enriquecimento de representações semânticas fornecidas por *microservices* de dados, o método não suporta serviços que não fornecem dados conectados nem descrições que manipulam dados sintáticos.

3.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

A revisão sistemática conduzida em colaboração com outros pesquisadores ajudou a entender a problemática envolvida no contexto de composição, descoberta, seleção e descrição de serviços, e a identificar a importância da Web Semântica na consecução desses processos. Durante a revisão, foram analisados os trabalhos primários mais representativos citados pelos trabalhos selecionados e identificada a existência de abordagens que direcionam esforços tanto na construção de ontologias que descrevem serviços, quanto na elaboração de serviços semânticos. A partir dos resultados obtidos, uma coleta bibliográfica adicional foi conduzida com o objetivo de buscar abordagens que focam especificamente no enriquecimento semântico de serviços, as quais complementaram os artigos primários selecionados para análise.

A partir do estudo desses trabalhos, observam-se algumas oportunidades de melhorias e novas direções de pesquisa. Muitos pesquisadores têm se dedicado a criar novas ontologias que, na prática, não favorecem a integração dos dados e a composição dos

serviços devido à diversidade de conceitos criados para descrever o mesmo objeto do mundo real. Além disso, a maioria dos trabalhos encontrados na literatura propõem ferramentas para aprimorar serviços Web com tecnologias da Web Semântica, requerendo ainda intervenção humana. Ademais, pouca atenção tem sido dada às representações fornecidas pelos serviços Web; os trabalhos majoritariamente buscam enriquecer a descrição dos serviços.

Em suma, este capítulo teve como objetivo apresentar o estado da arte e identificar as lacunas existentes nos trabalhos encontrados de modo a garantir a originalidade da abordagem proposta nesta dissertação. O próximo capítulo apresenta a proposta, detalhando o mecanismo para enriquecimento semântico de serviços de dados, o qual constrói ontologias de domínio para descrever os dados geridos pelos serviços. A proposta considerada aspectos pouco explorados em trabalhos anteriores, como a produção dinâmica de representações semânticas utilizando os conceitos da ontologia criada, bem como o reúso de conceitos já empregados em outras fontes de dados.

4 ABORDAGEM PROPOSTA

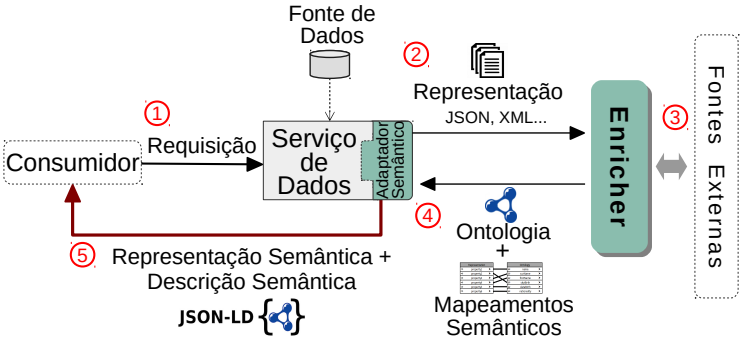
Conforme exposto no Capítulo 3, muitas pesquisas têm direcionado esforços para o enriquecimento de descrições de serviços utilizando como instrumento anotações semânticas. No entanto, as abordagens encontradas na literatura não atacam o problema de enriquecer automaticamente o serviço, de modo que este forneça, em tempo de execução, tanto descrições quanto representações associadas a conceitos semânticos. Este capítulo introduz um mecanismo automático para enriquecimento semântico de serviços de dados e detalha os componentes da arquitetura proposta neste trabalho.

4.1 MECANISMO DE ENRIQUECIMENTO SEMÂNTICO DE SERVIÇOS DE DADOS

Este trabalho propõe uma abordagem para enriquecer semanticamente serviços de dados de maneira dinâmica. Neste contexto, dinâmico significa em tempo de execução, isto é, serviços de dados implantados (em um *container* Web, por exemplo) devem ser capazes de vincular conceitos semânticos, definidos em ontologias de domínio, a dados sintáticos e fornecê-los utilizando o modelo RDF. O enriquecimento é alcançado gerando tanto uma descrição quanto representações semânticas que devem ser fornecidas pelo serviço de dados. Para isso, é necessário um mecanismo capaz de construir e evoluir ontologias de domínio a partir de representações sintáticas disponíveis sob demanda pelo serviço. Alinhado a isso, um adaptador semântico (referenciado como *Semantic Adapter* nos artigos em inglês que descrevem a arquitetura e ao longo desta dissertação) deve ser incluído no serviço de dados de modo a viabilizar as associações semânticas e, por conseguinte, fornecer aos consumidores novas representações, bem como uma nova descrição, de maneira dinâmica e transparente ao consumidor. Assim, para que o serviço seja enriquecido dinamicamente, é preciso que este importe a biblioteca do *Semantic Adapter* antes da sua implantação (ver seção 4.2.3 para mais detalhes).

A Figura 11 apresenta uma visão geral do mecanismo para enriquecimento semântico de um serviço de dados. Inicialmente, um consumidor envia uma requisição para um serviço. A requisição é interceptada pelo *Semantic Adapter*, que envia a um *Enricher* uma representação sintática (serializada, por exemplo, em XML ou JSON) do dado requerido. O *Enricher* deve ser capaz de extrair todos os elementos da representação sintática e construir uma onto-

Figura 11 – Esquema de enriquecimento dinâmico de serviços de dados.



Fonte: Elaborado pelo autor (2018).

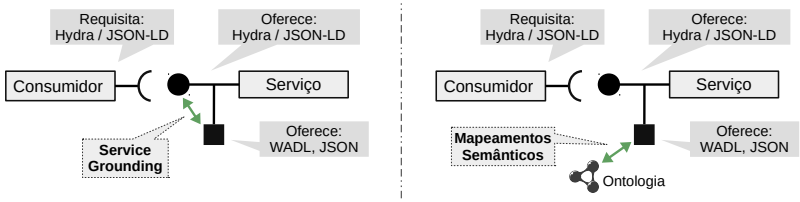
logia de domínio para o serviço, incluindo classes, propriedades de dados (*datatype properties*) e propriedades de objetos (*object properties*), bem como links de propriedades equivalentes identificadas em ontologias já existentes. Tais equivalências podem ser descobertas utilizando fontes externas que possuem relações com o domínio da ontologia gerada.

Como saída, o *Enricher* retorna a ontologia de domínio criada, juntamente com associações semânticas – denominadas Mapeamentos Semânticos ($\mathcal{SM}$) – entre os atributos sintáticos das representações do serviço e os conceitos da nova ontologia. Um mapeamento semântico é definido como uma tripla $\mathcal{SM} = \{a, c, t\}$, onde a é o atributo oriundo da representação sintática, c é o conceito semântico representado na ontologia e t é seu tipo: uma classe, uma propriedade de dados ou uma propriedade de objetos. Suponha que tenha sido criada uma propriedade de dados c, onde $c =$ `"http://data-service/ontology#name"`, para o atributo $a =$ `"name"` de uma determinada representação. O mapeamento semântico gerado deve ser $\mathcal{SM} =$ {`<"name">`, `<"http://data-service/ontology#name">`, `<"Datatype-property">`}.

Os mapeamentos semânticos se assemelham ao conceito de *service grounding* exposto por Schwichtenberg, Gerth e Engels (2017), conforme discutido no Capítulo 3. O *service grounding* estabelece ligações entre os elementos de uma descrição sintática e os elementos de uma descrição semântica equivalente. Por outro lado, os mapeamentos semânticos estabelecem ligações entre os atributos

existentes em descrições e representações sintáticas com os conceitos (utilizados pelas novas descrições e representações semânticas) existentes na ontologia de domínio. A Figura 12 ilustra um serviço oferecendo especificações tanto sintáticas quanto semânticas, e a atuação do *service grounding* (à esquerda) e dos $\mathcal{SM}$ (à direita).

Figura 12 – Service grounding *versus* Mapeamentos semânticos.



Fonte: Adaptado de Schwichtenberg, Gerth e Engels (2017).

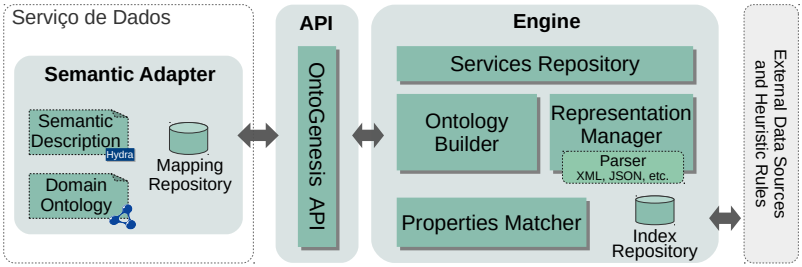
Os mapeamentos semânticos são úteis para gerar não apenas representações, mas também descrições semânticas para o serviço de dados. Nesse sentido, de acordo com o conjunto de $\mathcal{SM}$ produzidos pelo *Enricher*, as representações sintáticas são automaticamente convertidas em documentos JSON-LD pelo *Semantic Adapter*, que, por sua vez, os envia ao consumidor como retorno da requisição. Do mesmo modo, a descrição semântica para o serviço é gerada de acordo com os parâmetros de entrada e saída, e é publicada por meio de um *endpoint* de modo que os clientes semânticos possam consumí-la e, então, interpretar as capacidades do serviço.

Conforme mostrado na Figura 11, o *Semantic Adapter* é um componente conectado a um serviço de dados que tem como objetivo interceptar o retorno das requisições para fornecer dados conectados aos consumidores. O JSON-LD é utilizado para serializar não apenas as representações fornecidas pelo serviço, mas também a sua descrição, a qual define as entidades que um serviço gerencia e provê informações de como obtê-las. O *Semantic Adapter* pode ser visto como um conector responsável pela comunicação entre o serviço e o *Enricher*. A próxima seção apresenta a arquitetura interna do *Enricher* e detalha o funcionamento do *Semantic Adapter*.

4.2 ARQUITETURA DO ONTOGENESIS

Segundo Martin Fowler (2003 apud HOHMANN, 2003, pp. XXI), chegar a uma definição absoluta de arquitetura de *software* não é uma tarefa fácil, visto que este se trata de um termo relati-

Figura 13 – Visão geral da arquitetura do OntoGenesis.



Fonte: Elaborado pelo autor (2018).

vamente subjetivo. Cada engenheiro de *software* descreve de forma ímpar a arquitetura do seu sistema, selecionando as principais partes que o compõem e como elas se encaixam e se comunicam. Hohmann (2003) menciona ainda que a chave para se compreender uma arquitetura é adotar uma imagem que represente um ponto de vista do *software*. Perry e Wolf (1992), por outro lado, buscam ser mais precisos, alegando que uma arquitetura deve abarcar dados, elementos de processamento e conectores, os quais conectam os elementos de forma lógica. Neste trabalho, entende-se que uma arquitetura deve apresentar uma abstração dos componentes (chamados também genericamente de elementos por alguns autores) de um sistema computacional, as suas capacidades e como estes interagem entre si.

O OntoGenesis é uma arquitetura composta de componentes e subcomponentes que interagem entre si visando ao enriquecimento semântico de serviços de dados de forma automática. A arquitetura do OntoGenesis é dividida em três componentes principais, conforme ilustrado na Figura 13. O primeiro, o OntoGenesis *Engine*, é responsável pela construção de uma ontologia para o serviço de dados e pela criação dos mapeamentos semânticos de acordo com os elementos das representações sintáticas do serviço. Ele utiliza ainda fontes de dados externas já enriquecidas semanticamente para identificar equivalências e aprimorar a ontologia de domínio construída. O segundo componente, chamado OntoGenesis API, é uma Web API que fornece uma interface de comunicação para acessar as funcionalidades fornecidas pelo *Engine*. Finalmente, o *Semantic Adapter* trata-se de uma biblioteca para acessar o OntoGenesis API e auxilia os serviços de dados no fornecimento dinâmico de descrições e representações semânticas.

É importante notar que tanto o *Engine* quanto o API atuam

como o *Enricher* do mecanismo de enriquecimento ilustrado na Figura 11. As próximas seções apresentam as funções de cada componente e as suas interações.

4.2.1 OntoGenesis *Engine*

O objetivo do *OntoGenesis Engine*, além de gerar os mapeamentos semânticos, é construir ontologias de domínio para serviços de dados tradicionais/sintáticos que estejam alinhadas a outras ontologias externas já existentes. Conforme mostrado na Figura 13, o OntoGenesis *Engine* compreende cinco subcomponentes, que são detalhados a seguir: *Services Repository*, *Representation Manager*, *Ontology Builder*, *Index Repository* e *Properties Matcher*.

- **Services Repository**

O *Services Repository* gerencia as principais informações relacionadas aos serviços de dados. As informações armazenadas incluem o nome do serviço, o seu URI de acesso (incluindo o *host*/IP, porta e o *path*) e os recursos semânticos criados pelo OntoGenesis, isto é, a ontologia de domínio e os mapeamentos semânticos. Além disso, o *Services Repository* auxilia na operação dos outros subcomponentes, fornecendo as informação necessárias acerca dos serviços registrados no OntoGenesis.

- **Representation Manager**

O *Representation Manager* busca extrair os elementos de uma representação sintática fornecida por um serviço de dados (como os atributos e seus valores) úteis para o processo de construção da ontologia. Para este fim, ele fornece uma abstração comum para qualquer formato de dados, de modo que *parsers* específicos possam ser encapsulados neste subcomponente sem muito esforço. Isto permite ao OntoGenesis *Engine* lidar adequadamente com diversos formatos de dados, como JSON, XML, CSV, entre outros.

Como os resultados do *Representation Manager* são usados por outros subcomponentes, detalhes específicos do formato da representação são descartados em prol de uma abstração comum. Com isso, emprega-se uma abstração inspirada em objetos, na qual uma representação é composta por um conjunto de objetos, cada um contemplando um mapeamento dos nomes de atributos para um conjunto de valores pertencentes ao atributo. Quanto aos valores dos atributos, pode-se dividir em valores de objeto (criando

uma estrutura em árvore apontando para outros objetos) e valores primitivos (numéricos, *strings* e booleanos). Os JSON *arrays*, especificamente, são considerados como valores múltiplos para o mesmo nome de atributo, ou seja, a ordem do *array* não é considerada na abstração da representação.

- **Ontology Builder**

O *Ontology Builder* tem o papel de criar e gerenciar ontologias de domínio para cada serviço de dados registrado no *Services Repository*. Para se construir uma ontologia que descreva os dados geridos pelo serviço, o *Ontology Builder* analisa os elementos sintáticos extraídos pelo *Representation Manager*. Caso uma ontologia de domínio já tenha sido construída a partir de uma representação anterior enviada pelo serviço, o *Ontology Builder* atualiza a ontologia do domínio com os novos elementos identificados. Portanto, a ontologia do serviço evolui à medida que novas representações são fornecidas ao OntoGenesis. O processo envolvendo a construção da ontologia de domínio é detalhado na seção 4.2.1.1 deste capítulo.

- **Index Repository**

O *Index Repository* é responsável por armazenar, em uma base de dados NoSQL do tipo chave-valor, índices de dados literais originários tanto de fontes externas quanto dos serviços de dados para os quais o OntoGenesis construirá as ontologias de domínio. Dois tipos de índices foram projetados com o objetivo de atender as necessidades específicas do OntoGenesis: um para indexar as informações dos dados fornecidos pelo serviço e outro para indexar os dados extraídos de fontes externas. Considerando que os conjuntos de dados possuem propriedades e valores, ambos os índices usam a propriedade como chave para indexação do valor (ou objeto, no caso das fontes externas) no conjunto de dados. Somente as triplas das fontes externas cujo objeto é um literal são armazenadas no índice.

Os dados associados a propriedades oriundas das representações fornecidas pelos serviços são mantidos em um índice na forma $p \rightarrow o$, onde p é um predicado RDF (mais especificamente, a propriedade que foi criada na ontologia com base nos elementos da representação sintática) e o é a forma léxica de um valor literal observado para p. Em outras palavras, esse índice é implementado como uma tabela *hash*, em que um mapa de propriedades aponta para um conjunto de valores. Ainda que esse primeiro índice não seja tão eficiente para fins de consulta, ele permite uma enumeração

eficiente de o – o que é satisfatório para o OntoGenesis, visto que os dados dos serviços são enviados sob demanda, diferentemente das fontes externas, em que os dados normalmente são enviados de forma esporádica e em larga escala. Assim, o índice criado a partir dos dados de fontes externas é representado por um mapa baseado em *hash* de propriedades que apontam para um Autômato Finito Determinístico (AFD) que aceita todos valores (`range` de valores literais das triplas RDF) observados para a propriedade (predicado). Esse índice permite que pesquisas por termos semelhantes sejam feitas de forma eficiente. Detalhes referentes a esse índice, tais como o processo de construção dos autômatos e a consulta baseada em similaridade, estão expostos na seção 4.2.1.2 deste trabalho.

- **Properties Matcher**

Por fim, o *Properties Matcher* é um subcomponente do *Engine* que tem como finalidade identificar equivalências entre o conjunto de termos de cada propriedade fornecida por um serviço de dados, e o conjunto de termos de cada propriedade existente em fontes externas. Para tanto, ele acessa o *Index Repository* para obter todos os termos associados às propriedades, isto é, recupera o índice que armazena os dados do serviço e o índice que armazena os dados das fontes externas para executar o *match* entre os objetos das propriedades dos diferentes conjuntos de dados. O objetivo do *Properties Matcher*, portanto, é descobrir as possíveis equivalências entre a ontologia de domínio construída e as ontologias já conhecidas usadas por fontes externas. A seção 4.2.1.3 descreve os pormenores do processo de *match* e apresenta o algoritmo concebido para estabelecer as equivalências entre as ontologias.

4.2.1.1 Construção da ontologia

A Figura 14 ilustra a amostra de uma ontologia gerada em RDF/XML (b) com base nos elementos de uma dada representação em JSON (a). Os valores contidos no JSON representam dados reais – publicados pela Secretaria da Segurança Pública do estado de São Paulo (SSP/SP) – de um Boletim de Ocorrência (B.O.) com informações sobre uma pessoa envolvida (vítima, testemunha ou autor de um crime). Os nomes dos atributos existentes na representação são mapeados para uma instância de propriedade na ontologia. O documento de referência da OWL (W3C, 2004a) distingue duas categorias principais de propriedades que um construtor de ontolo-

gias pode definir. Assim, o tipo da propriedade é determinado pelo *Ontology Builder* da seguinte forma:

1. `owl:ObjectProperty` se usado exclusivamente com valores que referenciam um objeto da representação (como as linhas 5 e 11 da Figura 14 (a) e (b), respectivamente); ou

2. `owl:DatatypeProperty` se usado exclusivamente com valores primitivos (como as linhas 6 e 20 da Figura 14 (a) e (b), respectivamente).

O URI da instância da propriedade é determinado por uma concatenação simples do nome do atributo com o prefixo da ontologia, o qual é definido de acordo com o URI do serviço armazenado no *Services Repository*.

As classes da ontologia são geradas a partir de duas formas. Na primeira, qualquer instância p de um `owl:ObjectProperty` origina uma nova classe C (e.g., linha 10 da Figura 14 (b)), bem como a tripla $\langle p$ `rdfs:range` $C\rangle$ (linha 13). Toda propriedade q extraída dos valores de objeto que correspondam ao nome de atributo de p, terão também a tripla $\langle q$ `rdfs:domain` $C\rangle$ (linhas 20-31). A segunda forma é o nome do *endpoint* de onde a representação foi originária. Uma classe R gerada dessa maneira será o `rdfs:domain` de todas as propriedades que correspondem aos nomes de atributos encontrados nos objetos raiz das representações que compartilham o mesmo nome do *endpoint* (e.g., linha 16). A razão para isso é que, dentro de um serviço Web tradicional, *endpoints* (por exemplo, um método de recurso em JAX-RS[1]) irão prover entidades do mesmo tipo. Ademais, o nome desse *endpoint* pode ser inferido por uma ferramenta que automatiza o envio de representações ao OntoGenesis.

Em paralelo ao processo de construção da ontologia de domínio, o *Ontology Builder* também produz os Mapeamentos Semânticos ($\mathcal{SM}$), conforme apresentado na seção 4.1. Para cada propriedade p da ontologia de um tipo t criado em conformidade com um atributo a presente na representação enviada pelo serviço, é gerada uma tripla $\mathcal{SM} = \{a, p, t\}$, útil para associar os conceitos ontológicos na representação, gerando, assim, os os documentos JSON-LD.

Embora a ontologia produzida pelo *Ontology Builder* forneça conceitos semânticos relacionados aos dados providos pelo serviço, tais conceitos só são conhecidos pelo serviço de dados. Com o objetivo de permitir uma integração mais rica com outras aplicações

[1] JSR 339: <https://jcp.org/en/jsr/detail?id=339>.

Figura 14 – Amostra de: (a) Representação de um Boletim de Ocorrência em JSON; e (b) Ontologia em RDF/XML construída a partir da representação.

(a)

```
{ "BoletimOcorrencia": {
      "idBO": "2015-10004-794",
      "local": "Estação do Metrô ...",
      ...
      "pessoaEnvolvida": {
         "nome": "CARLOS ALBERTO DOS SANTOS",
         "rg": "015****18",
         "nacionalidade": "Brasileira",
         "localNascimento": "Sao Paulo-SP",
         "dataNascimento": "21-12-1966",
         "genero" : "Masculino",
         ...
      }
  }
}
```

(b)

```
 <rdf:RDF
    xmlns="http://servico-exemplo/ontology#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xml:base="http://servico-exemplo/ontology">
  <owl:Ontology rdf:about="http://servico-exemplo/ontology"/>
  <owl:Class rdf:ID="BoletimOcorrencia"/>
  <owl:Class rdf:ID="PessoaEnvolvida"/>
  <owl:ObjectProperty rdf:ID="temPessoaEnvolvida">
    <rdfs:domain rdf:resource="#BoletimOcorrencia"/>
    <rdfs:range rdf:resource="#PessoaEnvolvida"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="idBO">
    <rdfs:domain rdf:resource="#BoletimOcorrencia"/>
    <rdfs:range rdf:resource="xsd:string"/>
  </owl:DatatypeProperty>
   ...
  <owl:DatatypeProperty rdf:ID="nome">
    <rdfs:domain rdf:resource="#PessoaEnvolvida"/>
    <rdfs:range rdf:resource="xsd:string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="rg">
    <rdfs:domain rdf:resource="#PessoaEnvolvida"/>
    <rdfs:range rdf:resource="xsd:string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="nacionalidade">
    <rdfs:domain rdf:resource="#PessoaEnvolvida"/>
    <rdfs:range rdf:resource="xsd:string"/>
  </owl:DatatypeProperty>
   ...
</rdf:RDF>
```

Fonte: Elaborado pelo autor (2018).

ou serviços existentes no âmbito da Web Semântica, é essencial que a ontologia construída reutilize (ou se alinhe a) conceitos definidos por ontologias/vocabulários abertos e já conhecidos. Para tanto, o OntoGenesis busca descobrir novas relações de equivalência (especificamente propriedades) entre a ontologia construída e fontes externas (e.g., ontologias, *datasets* semânticos e bases de conhecimento), a fim de aprimorar e expandir a ontologia e, posteriormente, permitir que tarefas de *reasoning* sejam concretizadas.

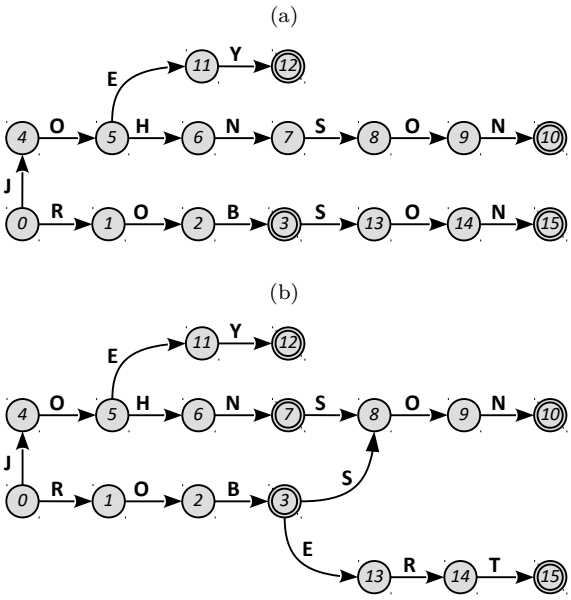
4.2.1.2 Construção dos índices e consulta baseada em similaridade

Conforme já mencionado, o índice gerenciado pelo *Index Repository* responsável por armazenar os dados obtidos de fontes externas foi projetado na forma $p \rightarrow M$, onde p é um predicado e M é um Autômato Finito Determinístico (HOPCROFT; ULLMAN, 1990). O conjunto de cadeias que M aceita é chamado de linguagem reconhecida por M e é simbolicamente representado por $\mathcal{L}(M)$. Logo, $\mathcal{L}(M)$ diz respeito ao conjunto de todos os termos extraídos dos objetos o de p presentes nas triplas RDF das fontes externas.

Um Autômato Finito Determinístico (AFD) é representado graficamente como um grafo rotulado direcionado, no qual os nós são os estados do AFD e os vértices representam as transições marcadas com caracteres de entrada (HOPCROFT; ULLMAN, 1990). A tarefa principal de um AFD M é reconhecer se uma determinada sequência de caracteres de entrada ($w = w_1 w_2 \dots w_n$) pertence à linguagem do autômato. Para tanto, é feita a leitura do próximo caractere de entrada realizando a transição correspondente no autômato. Esse processo inicia com w_1 no estado inicial – denotado graficamente por uma seta de entrada sem estado predecessor – e termina após o processamento de w_n, quando não há mais caracteres de entrada. Nesse ponto, $w \in \mathcal{L}(M)$ se, e somente se, o estado corrente se encontra em um estado de aceitação – graficamente representado como um círculo duplo.

A construção dos autômatos ocorre durante o carregamento, no OntoGenesis, de um *dataset* de uma fonte externa. Os autômatos são construídos de tal forma que a linguagem reconhecida pelo autômato M, para o predicado p, seja o conjunto de todas as formas lexicais de objetos literais observados para p nas fontes de dados externas. A Figura 15 (a) ilustra um exemplo de um AFD construído que aceita os termos *JOEY*, *JOHNSON*, *ROB* e *ROBSON*. O algoritmo para adicionar um novo valor literal w em M consiste em

Figura 15 – Amostra de um AFD aceitando termos referentes a uma propriedade `dbo:firstName`: a) AFD em construção; b) AFD minimizado com novos termos.



Fonte: Elaborado pelo autor (2018).

tentar reconhecer w, e quando o reconhecimento falha, insere-se as transições e estados necessários para que w seja aceito por M. Por exemplo, adicionar o termo *JOHN* ao AFD ilustrado na Figura 15 (a) requer apenas marcar o estado 7 como final. Por outro lado, ao adicionar o termo *ROBERT*, é necessário incluir as transições e estados para *ERT*, após o termo *ROB* ter sido reconhecido no estado 3 da Figura 15 (a). Uma vez que todas as triplas das fontes externas foram processadas e adicionadas ao autômato, o AFD M é minimizado – para fins de melhoria de desempenho durante as consultas – e persistido no índice . A Figura 15 (b) ilustra o AFD já minimizado e aceitando os novos termos do exemplo citado.

Um requisito importante referente ao índice supracitado é o processamento rápido de consultas baseadas em similaridades em que, dada uma *string* w, uma *string* s no índice com $d(w, s) \leq t$ é selecionada, onde d é uma função de similaridade de texto. Por questões de simplicidade, consideramos neste trabalho a distância

de Levenshtein com um *threshold* $t = 1$ ($d_L(w, s) \leq 1$), embora outras medidas de similaridade possam ser aplicadas de maneira análoga. A distância Levenshtein representa o número mínimo necessário de inserção, remoção ou substituição para transformar uma *string* a em uma *string* b. Por exemplo, a distância Levenshtein entre os termos "mesa" e "meta" é 1, enquanto $d_L(mesa, maca) = 2$.

Para realizar a consulta que aplica a medida de similaridade Levenshtein, deve-se calcular um autômato Levenshtein $LEV_n(w)$ que aceita todas as cadeias onde $d_L(w, s) \leq 1$, isto é, todas as cadeias cuja distância de Levenshtein de w esteja dentro do *threshold*. Verifica-se, então, se $\mathcal{L}(M) \cap \mathcal{L}(LEV_n(w)) \neq \emptyset$. Uma vez que o autômato de interseção executa de forma concorrente a intersecção dos autômatos, tal verificação de interseção vazia pode ser feita diretamente sem a necessidade de materializar $LEV_1(w)$, conforme exposto por Schulz e Mihov (2002). Caso o resultado da intersecção entre os autômatos seja vazio, entende-se que o autômato M não reconhece nenhuma cadeia do conjunto de possíveis variações de w com Levenshtein ≤ 1. Em outras palavras, o valor atribuído a uma propriedade p' oriunda de uma representação do serviço não possui nenhuma equivalência aos valores associados à propriedade p existente em uma fonte externa, visto que não há termos que atendam à medida de similaridade estabelecida.

Regras heurísticas também podem ser adicionadas como fontes de informação para o processo de alinhamento das propriedades. Tais regras são padrões representados como expressões regulares, vinculados a uma propriedade p. Um exemplo simples de uma regra $\mathcal{R}$ é `dbo:date` → `0[1-9]|[12][0-9]|3[01] / 0[1-9]|1[0-2] / [0-9]{4}`. Quando os termos de uma propriedade p_1 (de um serviço de dados) correspondem, por exemplo, com tal regra $\mathcal{R}$, então p_1 pode ser considerada como uma propriedade equivalente a `dbo:date`. Uma vez que o índice de fontes externas consiste em um AFD M, as regras heurísticas podem ser incorporadas diretamente em M, permitindo que qualquer *string* que corresponda a uma regra seja considerada parte do índice.

4.2.1.3 *Matching* de propriedades

Com o intuito de calcular o grau de sobreposição de dados entre duas propriedades (contidas nos índices geridos pelo *Index Repository*) e encontrar propriedades equivalentes mais precisas, é importante ter um valor de força para cada *match* de propriedade executado. A seguinte equação foi elaborada para calcular a força $\mathcal{F}$

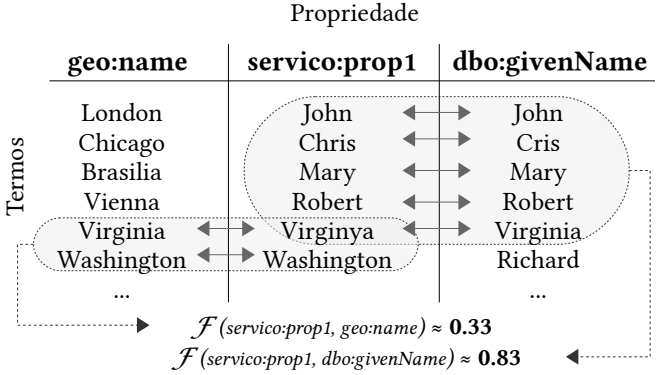
(valor compreendido entre 0 e 1) do *match* entre duas propriedades p_1 e p_2:

$$\mathcal{F}_{(p_1,p_2)} = \frac{|\mathcal{V}p_1 \bigcap_s \mathcal{V}p_2|}{|\mathcal{V}p_1|} \tag{4.1}$$

onde $\mathcal{V}p_1$ é o conjunto de termos providos pelo serviço de dados para uma dada propriedade p_1, e $\mathcal{V}p_2$ é o conjunto de valores associados a uma propriedade p_2 existente em uma fonte externa. A intersecção $\bigcap_s$ utiliza um algoritmo baseado em similaridade para verificar se um termo $t_1 \in \mathcal{V}p_1$ é similar a um termo $t_2 \in \mathcal{V}p_2$ para, então, ser considerado como uma correspondência válida. Como já descrito anteriormente, neste trabalho foi empregada a distância de Levenshtein, embora outras medidas de similaridade possam ser incorporadas à abordagem. Com base na força da sobreposição de termos, é possível identificar o grau de correspondência entre duas propriedades. Logo, quanto maior a força, maior a probabilidade de as propriedades serem equivalentes.

A Figura 16 ilustra um exemplo de aplicação da Equação 4.1 em um cenário com três propriedades. `servico:prop1` representa uma propriedade oriunda de uma ontologia construída automaticamente para um serviço de dados, enquanto as outras duas propriedades provêm de fontes de dados externas. Os retângulos com cantos arredondados representam $\mathcal{V}p_1 \bigcap_{d_L \leq 1} \mathcal{V}p_2$. Apesar de `geo:name` ter intersecções com a propriedade `servico:prop1`, pode-se ob-

Figura 16 – Exemplo da sobreposição de valores associados a propriedades e aplicação da força de equivalência.



Fonte: Elaborado pelo autor (2018).

servar que a força dessa relação (0,33) é menor que a força entre `servico:prop1` e `dbo:givenName` (0,83). Em vista disso, podemos definir um limiar t (*threshold*) da força de equivalência $\mathcal{F}$ de modo que os pares de propriedades equivalentes cuja força foi significativamente baixa ($\mathcal{F} \leqslant t$) sejam desconsiderados da ontologia gerada.

Algorithm 1 Matching de Propriedades

```
1:  procedure MatchProperties(𝒪, ℐ, ℐ′, α)
2:      for each P1 ∈ GetProperties(ℐ) do
3:          𝒱p1 ← GetValues(ℐ, P1)
4:          for each P2 ∈ GetProperties(ℐ′) do
5:              𝒱p2 ← GetValues(ℐ′, P2)
6:              overlap ← 0
7:              for each v ∈ 𝒱p1 do
8:                  if v ∈ 𝒱p2 then
9:                      overlap ← overlap + 1
10:                 end if
11:             end for
12:             strength ← overlap / |𝒱p1|
13:             equivProp ← ⟨P1  owl:equivalentProperty  P2⟩
14:             if strength ≥ α then
15:                 add equivProp to 𝒪
16:             else if equivProp ∈ 𝒪 then
17:                 remove equivProp from 𝒪
18:             end if
19:         end for
20:     end for
21: end procedure
```

O Algoritmo 1 apresenta os principais passos para determinar equivalências de propriedades a partir da comparação entre os valores das propriedades fornecidos por serviços de dados e por fontes de dados externas. O procedimento recebe como entrada uma ontologia $\mathcal{O}$ construída para o serviço de dados; os índices $\mathcal{I}$ e $\mathcal{I}'$ contendo, respectivamente, dados das representações dos serviços e fontes externas; e o *threshold* para a força de equivalência representado por α. O algoritmo inicia realizando o *match* de todas as propriedades obtidas de fontes externas (linhas 2-4). Para cada par P_1, P_2, o algoritmo contabiliza[2] quantos valores enumerados em $\mathcal{V}p_1$ (linha 3) estão presentes também em $\mathcal{V}p_2$ (linhas 5-10). A força é

[2] Como foi adotada a medida de similaridade de Levenshtein, a implementação

computada na linha 12, em consonância com a Equação 4.1. Na sequência, é criada uma tripla declarando que P_1 e P_2 são equivalentes (linha 13) e esta é adicionada à ontologia se a força satisfizer o limite α, ou removido da ontologia, caso contrário (linhas 14 -17).

Para melhorar a eficiência, a implementação do algoritmo no OntoGenesis *Engine* foi feita de tal forma que o resultado da força de equivalência seja armazenado na memória e atualizado para cada novo conjunto de valores fornecidos pelo serviço de dados. Assim, torna-se dispensável recalcular a interseção de todo o conjunto para atualizar a força de equivalência, sendo necessário calcular apenas a interseção com os novos termos recebidos.

4.2.2 OntoGenesis API

O OntoGenesis API é uma API baseada em REST destinada a ser acessível para todos os serviços de dados que devem ser enriquecidos semanticamente. Essa API expõe duas funcionalidades principais: i) registrar o serviço de dados no *Index Repository*, e ii) acionar o OntoGenesis *Engine* para construir/evoluir a ontologia de domínio e os mapeamentos semânticos com base nas representações recebidas dos serviços registrados. Nota-se, pois, que o OntoGenesis API é uma camada intermediária de comunicação entre os serviços de dados e o OntoGenesis *Engine*.

Quando um determinado consumidor interage com um serviço já registrado no OntoGenesis, tal serviço envia de forma transparente (detalhes na seção 4.2.3) ao OntoGenesis API o recurso solicitado pelo consumidor para ser enriquecido semanticamente. O OntoGenesis API invoca o *Engine* e retorna ao serviço de dados a sua nova ontologia, juntamente com os mapeamentos semânticos. Desse modo, os serviços de dados legados que manipulam representações puramente sintáticas podem se registrar no OntoGenesis e fornecer dinamicamente uma ontologia de domínio e dados enriquecidos com semântica.

O OntoGenesis API suporta configurações personalizadas (Figura 17), como o *threshold* para a força de equivalência de propriedades e um tamanho de *buffer* de representações. O *threshold* deve ser configurado com valores de 0 a 1 e é usado durante a execução do algoritmo de *matching* de propriedades (Algoritmo 1). O tamanho do *buffer* de representação indica quantas representações fornecidas

do algoritmo empregou a técnica *Levenshtein automata imitation* (SCHULZ; MIHOV, 2002) para computar as buscas de w em M, em um tempo linear ao comprimento de w – o que evita a enumeração de $\mathcal{V}p_2$.

Figura 17 – Arquivo de configuração do OntoGenesis API.

```
server.port: 8090
config.maxNumberRepresentationsBuffer: 1
#Limiar da força de equivalência de propriedades
config.propertiesIntersectionThreshold: 0.80
#Base usada para armazenar os índices
config.externalSourceBD: "dfas.db"
```

Fonte: Elaborado pelo autor (2018).

pelo serviço de dados devem ser enviadas ao OntoGenesis *Engine* concomitantemente. Quando o *buffer* é personalizado com um valor superior a zero, as representações de cada serviço são agrupadas em blocos antes de serem enviadas para o *Engine*. Assim, em vez de encaminhar representações uma a uma, o API envia um lote de representações como forma de reduzir as chamadas e aprimorar o desempenho do serviço em relação ao tempo de resposta. É importante ressaltar que as requisições que terão desempenho aprimorado são as que estão armazenadas no *buffer*. Durante a n-ésima requisição, sendo $n-1$ o tamanho do *buffer*, o tempo de resposta tende a subir, visto que n representações são enviadas ao *Engine* para executar o processamento de enriquecimento (incluindo a atualização da ontologia, dos mapeamentos semânticos e a operação de *matching* das propriedades). Essa estratégia de *"bufferizar"* as representações foi utilizada nos experimentos que buscam comparar o OntoGenesis com outros *matchers* de ontologias (ver seção 5.5).

Ademais, o OntoGenesis API oferece uma interface que permite carregar novas fontes de dados externas no *Index Repository*. Consequentemente, os usuários podem carregar novos *datasets*, para que seus dados sejam considerados em tempo de execução pelo *Properties Matcher* em futuras requisições dos serviços de dados.

4.2.3 *Semantic Adapter*

O *Semantic Adapter* é um componente que deve ser anexado a um serviço de dados que necessita fornecer dinamicamente uma descrição e representações associadas a conceitos semânticos. Conforme mostrado na Figura 11, este componente é responsável pela interação entre o serviço de dados e o OntoGenesis API. Além de realizar a importação no serviço, é necessário que o desenvolvedor responsável pelo serviço indique, em um arquivo de configuração, o URI através do qual o OntoGenesis API está respondendo.

Este componente armazena três elementos principais: a descrição semântica criada para o serviço de dados, a qual emprega o Hydra como vocabulário para descrever os seus recursos; a ontologia de domínio gerada pelo *Engine*; e o *Mapping Repository* (ver Figura 13). Este último tem a finalidade de armazenar as associações dos elementos extraídos das representações e da descrição do serviço com os conceitos definidos na ontologia (ou seja, os mapeamentos semânticos). Ele é acessado sempre que o serviço responde a qualquer requisição. Tanto a descrição semântica quanto a ontologia são disponibilizadas pelo *Semantic Adapter* através de um *endpoint* do serviço, de modo a torná-los acessíveis.

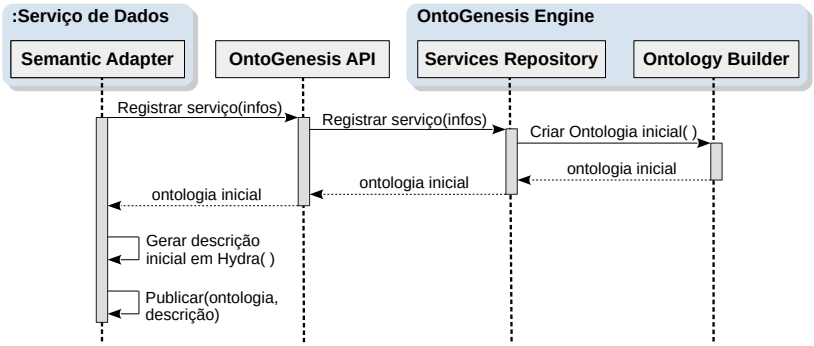
O *Semantic Adapter* possui essencialmente três funcionalidades: i) registro do serviço no OntoGenesis *Engine*; ii) interceptação das respostas para enriquecimento das representações de retorno; e iii) enriquecimento da descrição do serviço.

4.2.3.1 Registro do serviço

Ao usar o *Semantic Adapter*, o registro do serviço no OntoGenesis é executado automaticamente. A Figura 18 ilustra o diagrama de sequência representando a interação entre os componentes da arquitetura quando um serviço é implantado (passa pelo processo de *deploy*) em um *container* ou servidor de aplicação.

Durante a implantação, o *Semantic Adapter* é automaticamente invocado e envia ao OntoGenesis API uma solicitação de registro do serviço passando as informações necessárias para realizar tal operação, como *host*, porta e nome do serviço. O API direciona o pedido ao *Engine*, o qual efetua o registro no *Services Repository*. Em seguida, o *Ontology Builder* também é invocado para criar uma ontologia *dummy* inicial para o serviço. Como ainda não há dados oriundos das representações para criar os conceitos ontológicos, a ontologia contém apenas o arcabouço das informações gerais (e.g., linhas 1 a 8 da Figura 14). A ontologia inicial é retornada ao *Semantic Adapter*, que a publica para ser acessível aos consumidores. Uma descrição inicial do serviço (chamada de descrição *dummy*) também é gerada e publicada, embora nesse momento inicial ainda não possua nenhuma informação sobre os dados geridos pelo serviço, posto que nenhuma representação foi enriquecida até então. O processo de geração e enriquecimento da descrição é discutido mais adiante.

Figura 18 – Registro de um serviço de dados no OntoGenesis.



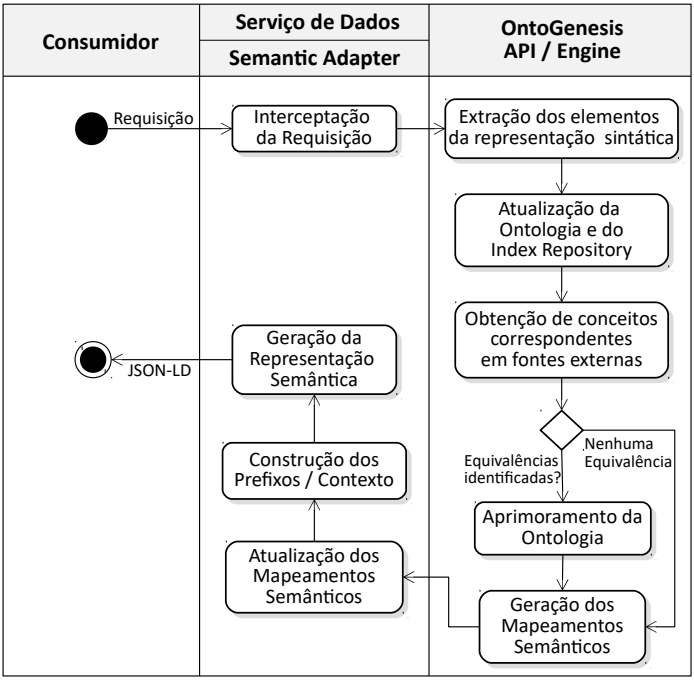
Fonte: Elaborado pelo autor (2018).

4.2.3.2 Geração das representações semânticas

A geração de representações semânticas é feita dinamicamente e de forma transparente. Quando algum consumidor solicita um determinado dado ao serviço, o *Semantic Adapter* intercepta a sua resposta e invoca o OntoGenesis API, que delega ao *Engine* a construção/evolução da ontologia de domínio juntamente com os mapeamentos semânticos. A ontologia é atualizada e publicada pelo serviço através do *Semantic Adapter*. É com base nela e nos mapeamentos semânticos que uma nova representação semântica serializada em JSON-LD é gerada e retornada ao cliente. Dessa forma, em vez de responder uma representação sintática, o serviço é fornece dados conectados (*Linked Data*) aos seus consumidores. O diagrama de atividades da Figura 19 sumariza o fluxo de processamento executado quando um determinado consumidor envia uma requisição a um serviço de dados registrado no OntoGenesis.

Para criar as representações em JSON-LD, o *Semantic Adapter* leva em consideração os tokens `@id`, `@type` e `@context` (LANTHALER; SPORNY; KELLOGG, 2014) – conceitos introduzidos na seção 2.1.2 deste trabalho. Tais tokens são gerados durante a penúltima fase de processamento, conforme ilustrado na Figura 19 (Construção dos Prefixos/Contexto). Os atributos do JSON-LD são mantidos da mesma forma que declarados na representação sintática, de modo a ter pouco impacto nos nomes dos campos presentes na representação original. Um dos principais benefícios dessa estratégia é permitir que não apenas os consumidores semânticos, mas

Figura 19 – Fluxo de processamento do enriquecimento semântico da representação.



Fonte: Elaborado pelo autor (2018).

também os não semânticos (ou seja, aqueles que conseguem processar apenas representações sem nenhum informação semântica) processem as novas saídas do serviço de dados, uma vez que os rótulos dos atributos se mantêm inalterados. O trecho de código da Figura 20 mostra um contexto JSON-LD mapeando os atributos da representação JSON com os conceitos da ontologia ilustrados no exemplo anterior da Figura 14.

4.2.3.3 Geração da descrição semântica

A descrição do serviço de dados também é enriquecida com recursos semânticos durante uma requisição. O *Semantic Adapter* armazena uma descrição semântica do serviço que contém todos os seus *endpoints*, os quais são identificados durante a implantação do serviço. A descrição gerada emprega o vocabulário Hydra

Figura 20 – Excerto do contexto JSON-LD.

```
1  {   "@context": {
2         "PoliceReport": "http://service-example/ontology/PoliceReport",
3         "reportID": "http://service-example/ontology/reportID",
4          ...
5         "personInvolved": "http://service-example/ontology/hasPersonInvolved",
6         "name": "http://service-example/ontology/name",
7         "docID": "http://service-example/ontology/docID",
8         "birthDate": "http://service-example/ontology/birthDate",
9         "nationality": "http://service-example/ontology/nationality",
10         ...
11    },
12    "@id": "http://service-example/policeReport?reportID=2015-10004-794",
13    "@type": "PoliceReport", ...
14 }
```

Fonte: Elaborado pelo autor (2018).

(LANTHALER; GÜTL, 2013), e define as entidades que o serviço de dados é capaz de gerenciar e indica como essas entidades podem ser obtidas. Para cada *IRITemplate* Hydra construído, são gerados os mapeamentos de cada uma de suas variáveis, além de sua operação alinhada ao método HTTP correspondente.

O código da Figura 21 descreve um *IriTemplate* (conforme apresentado na Capítulo 2) a partir de uma descrição gerada para um serviço de dados. A linha 22 mostra as operações suportadas e o tipo de recurso a ser retornado é indicado pelo elemento `returns` (linha 26). Inicialmente, tanto o `returns` quanto o `mapping` (linha 15) não possuem associações a conceitos semânticos. Estes são enriquecidos à medida que requisições são feitas por consumidores aos *endpoints* do serviços. Ademais, por meio da classe *IriTemplate* do Hydra, é possível mapear as variáveis de um URL – que será construído pelo consumidor em tempo de execução – para as propriedades da ontologia. No exemplo da Figura 21, `idPerson` deve ser substituído por um valor associado à propriedade "docId" utilizado pela classe "Person", a fim de acessar o recurso desejado.

O Algoritmo 2 mostra os passos básicos realizados pelo *Semantic Adapter* para enriquecer os mapeamentos da descrição Hydra. O procedimento é chamado após o retorno da ontologia pelo OntoGenesis. Como entrada, o algoritmo recebe o URI requisitado pelo consumidor, bem como o método HTTP utilizado na requisição (`GET`, `POST`, `DELETE`, etc.). Inicialmente, o *Semantic Adapter* busca na descrição gerada por um *IRITemplate* associado ao URI invocado pelo consumidor. Se o URI não estiver representado na

Figura 21 – Exemplo do *TemplatedLink* da descrição Hydra.

```
1  {
2    "@context":"http://www.w3.org/ns/hydra/context.jsonld",
3    "@id":"http://service-example/doc",
4    "@type":"hydra:ApiDocumentation",
5    "hydra:supportedClass":[
6      {
7        "@id":"http://service-example/ontology/EntryPoint",
8        "@type":"hydra:Class",
9        "hydra:supportedProperty": [
10         {
11           "@id":"findPerson"
12           "rdfs:range":"http://service-example/ontology/PersonInvolved",
13           "@type":["hydra:TemplatedLink","hydra:IriTemplate"],
14           "hydra:template":"http://service-example/person/rg/{?idPerson}",
15           "mapping":[
16             {
17               "@type":"IriTemplateMapping",
18               "variable":"idPerson",
19               "property":"http:/service-example/ontology/docID",
20               "required":true
21             } ],
22           "supportedOperation":[
23             {
24               "@type":"Operation",
25               "method":"GET",
26               "returns":"http://service-example/ontology/PersonInvolved"
27             } ]
28         } ]
29     } ]
30 }
```

Fonte: Elaborado pelo autor (2018).

descrição, um novo *IRITemplate* é criado para tal URI (linha 2). Posteriormente, na linha 3 são extraídos todos os parâmetros do URI (ver Apêndice C para mais detalhes). Para cada parâmetro existente, o algoritmo verifica se existe um mapeamento no *Mapping Repository* (linhas 7-8). Em caso positivo, a propriedade da ontologia associada ao parâmetro sintático é recuperada e adicionada como propriedade do mapeamento do *IRITemplate* (linhas 9-10). Caso contrário, o parâmetro e seus valores são adicionados em uma estrutura JSON, além de adicionar o parâmetro em uma variável auxiliar, $P_{\neg SM}$ (linhas 12-14). Após o processamento de todos os parâmetros contidos no URI, o algoritmo verifica se há parâmetros que não tenham sido associados a algum conceito semântico (devido à falta de mapeamentos semânticos) e, em caso afirmativo, envia ao OntoGenesis API um JSON contendo todos estes parâmetros e seus valores (linhas 17-18). Quando o *Engine* recebe tal JSON, todo o fluxo para adicionar na ontologia os conceitos semânticos relacio-

nados aos parâmetros contidos no documento JSON é executado. No final do processo, a ontologia atualizada é retornada ao serviço, juntamente com novos mapeamentos semânticos. As linhas 19 a 23 representam o mesmo processo de enriquecer os parâmetros descritos anteriormente para as linhas 8 a 10. Finalmente, todos os parâmetros do URI possuem uma propriedade na ontologia do serviço, tornando possível, então, estabelecer associações nos valores do `mapping` da descrição Hydra.

Já o Algoritmo 3 é responsável por associar o conceito semântico ao campo `returns` na descrição. Tal procedimento é feito com base no tipo de entidade da representação. O tipo pode ser encontrado na propriedade `@type` do JSON-LD gerado como retorno da requisição, conforme mostrado na Figura 20.

Este trabalho fornece uma implementação do *Semantic Adapter* compatível com a especificação JAX-RS, que padroniza a forma de criação de serviços REST na plataforma Java. Em suma, essa implementação busca prover o suporte tecnológico necessário para viabilizar o fornecimento de descrição e representações semânticas de forma dinâmica. Para tanto, faz-se uso das interfaces e classes de *Filters* e *Interceptors*[3], de modo a executar todo o processo de enriquecimento de forma transparente ao serviço e aos consumidores. Adicionalmente, o *Semantic Adapter* i) substitui as representações sintáticas providas originalmente pelo serviço por representações enriquecidas semanticamente, em formato JSON-LD, e ii) constrói uma descrição semântica para o serviço utilizando o vocabulário Hydra. Esse componente também disponibiliza *endpoints* para compartilhar a ontologia construída e a descrição Hydra de acordo com seu URI base. Dessa forma, as aplicações semânticas podem recuperar a descrição do serviço, bem como a sua ontologia, para entender as capacidades do serviço de dados e executar tarefas de *reasoning*.

[3] <https://jersey.github.io/documentation/latest/filters-and-interceptors.html>.

Algorithm 2 Enriquecimento dos mapeamentos da descrição

```
1: procedure EnrichMapping(URI, httpMethod)
2:     uriTemp ← GetOrCreateUriTemplate(URI, httpMethod)
3:     P_uri ← GetParametersOf(uriTemp)
4:     P_¬SM ← ∅
5:     json ← ∅
6:     for each param ∈ P_uri do
7:         if param ∈ SM then
8:             op ← SM.getOntologyPropertyOf(param)
9:             paramMapping ← uriTemp.getMappingOf(param)
10:            paramMapping.property ← op
11:        else
12:            V ← GetValuesOf(param)
13:            json.add(param, V)
14:            P_¬SM.insert(param)
15:        end if
16:    end for
17:    if P_¬SM ≠ ∅ then
18:        SendToOntoGenesis(json)        ▷ Updates: SM and Ontology
19:        for each param ∈ P_¬SM do
20:            op ← SM.getOntologyPropertyOf(param)
21:            paramMapping ← uriTemp.getMappingOf(param)
22:            paramMapping.property ← op
23:        end for
24:    end if
25: end procedure
```

Algorithm 3 Enriquecimento do retorno da operação.

```
1: procedure EnrichSupportedOperation(URI, httpMethod, jsonLD)
2:     uriTemp ← GetOrCreateUriTemplate(URI, httpMethod)
3:     operation ← GetOperation(uriTemp, httpMethod)
4:     type ← ExtractTypeFrom(jsonLD)
5:     operation.returns ← type
6: end procedure
```

4.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo introduziu o mecanismo de enriquecimento semântico de serviços de dados e apresentou a arquitetura proposta, intitulada OntoGenesis. Foram detalhados os três componentes principais que constituem a arquitetura: *Engine*, API e *Semantic Adapter*.

O *Engine* é responsável por construir ontologias de domínio para os serviços de dados registrados e aprimorar tais ontologias adicionando relações de equivalências já conhecidas e empregadas em fontes externas. Com base na equação e no algoritmo proposto, as ontologias tendem a evoluir à medida que novas representações são enviadas pelo serviço. O OntoGenesis API, por sua vez, se destina a prover uma interface Web que fornece as funcionalidades do *Engine*, além de permitir que customizações sejam feitas em algumas funções. Já o *Semantic Adapter* tem a função de interceptar as respostas do serviço de dados e enriquecer de forma transparente as representações de saída e a sua descrição com conceitos semânticos definidos na ontologia construída. Ademais, ele também disponibiliza *endpoints* de acesso à ontologia e à descrição criada.

Um ponto importante a ser destacado é que a geração da descrição e das representações semânticas é feita no serviço de dados por meio do *Semantic Adapter*, o qual armazena os mapeamentos semânticos e a ontologia de domínio. Tal característica possibilita que os serviços registrados no OntoGenesis continuem fornecendo dados conectados mesmo quando da indisponibilidade do *Engine* e/ou *API*. Além disso, a representação semântica gerada é retornada ao consumidor somente se este suportar o formato JSON-LD. Caso contrário, a representação sintática original é retornada.

Por fim, vale ressaltar que a proposta desta pesquisa se diferencia em alguns aspectos de trabalhos similares encontrados na literatura. O OntoGenesis avança no sentido de fomentar a construção e a evolução de ontologias de domínio para serviços de dados automaticamente, enquanto a maioria das propostas encontradas na literatura apresenta soluções que ainda exigem intervenções humanas significativas ao longo do processo de enriquecimento semântico. Os experimentos realizados para avaliação da proposta também se diferenciam dos trabalhos encontrados na literatura por terem sido conduzidos utilizando dados abertos reais e de interesse público, conforme exposto no próximo capítulo.

5 AVALIAÇÃO E RESULTADOS

Inicialmente, este capítulo introduz as principais medidas utilizadas para avaliar as técnicas de *matching* de ontologias. Em seguida, são apresentados o cenário projetado para avaliar a abordagem proposta, a metodologia experimental e os resultados alcançados. O objetivo dessa avaliação é demonstrar a aplicabilidade do OntoGenesis e analisar as medidas de conformidade e de desempenho utilizando dados abertos reais. Em relação às medidas de conformidade, foram analisadas precisão, cobertura e *F-Measure*. Quanto às medidas de desempenho, observou-se o tempo de processamento das principais funcionalidades da arquitetura, bem como o consumo de memória e de CPU. Por fim, este capítulo apresenta um experimento adicional projetado com o intuito de comparar o OntoGenesis com abordagens similares encontradas na literatura.

5.1 MEDIDAS DE AVALIAÇÃO

Diante do crescente número de algoritmos/alinhadores (também conhecidos como *matchers*) de ontologias que vem sendo propostos ao longo dos anos, surgiu a necessidade de definir e organizar as principais medidas para comparar os diversos *matchers* existentes e avaliar a qualidade dos alinhamentos gerados. As principais medidas usadas pela OAEI (*Ontology Alignment Evaluation Initiative*[1]) são as de conformidade e desempenho (EUZENAT; EHRIG; CASTRO, 2005).

As medidas de conformidade objetivam avaliar o grau de conformidade dos métodos de alinhamento, comparando um alinhamento A, gerado por um *matcher*, com um alinhamento de referência R. Dentre as mais utilizadas, estão as medidas de precisão, cobertura (mais conhecida como *recall*) e *F-Measure*.

- **Precisão**: representa a fração das correspondências relevantes dentre o total obtido. Em outras palavras, mede quantas correspondências estão corretas, dentre todas obtidas (A), de acordo com o alinhamento de referência (R). A medida é calculada através da Equação 5.1, resultando um valor compreendido entre o intervalo $[0, 1]$, sendo 1 o melhor caso e 0 o pior.

[1] Site da OAEI: <http://oaei.ontologymatching.org/>.

$$P(A,R) = \frac{|R \bigcap A|}{|A|} \tag{5.1}$$

- **Cobertura**: também denominada revocação, representa a fração das correspondências relevantes obtidas (A), dentre todas as correspondências existentes no alinhamento de referência (R). A Equação 5.2 ilustra o cálculo da cobertura, o qual resulta em um valor compreendido entre o intervalo $[0,1]$, sendo 1 o melhor caso e 0 o pior.

$$R(A,R) = \frac{|R \bigcap A|}{|R|} \tag{5.2}$$

- **F-Measure**: também conhecido como F_1 ou *F-score*, define uma única medida que considera tanto a precisão quanto a cobertura. A Equação 5.3 apresenta a fórmula utilizada para o seu cálculo. Quanto maior o valor de α, mais importância é dada à precisão em relação à cobertura. Por isso, $\alpha = 0.5$ é utilizado com mais frequência. Como resultado, têm-se de igual modo um valor compreendido entre o intervalo $[0,1]$.

$$F_{1_\alpha}(A,R) = \frac{P(A,R) \times R(A,R)}{(1-\alpha) \times P(A,R) + \alpha \times R(A,R)} \tag{5.3}$$

Outras medidas de conformidade, porém pouco comuns, são o *Fallout*, que mede, em termos percentuais, as correspondências falso positivas, e o *Overall*, que mede o esforço necessário para corrigir o alinhamento obtido (a proporção do número de erros no tamanho do alinhamento esperado).

As medidas de desempenho, por outro lado, consistem em mensurar o consumo de recursos para se obter os alinhamentos. Diferentemente das medidas de conformidade, as medidas de desempenho dependem do ambiente de execução, considerando, principalmente, o *hardware* sob o qual o *matcher* está executando. As duas principais medidas utilizadas são descritas a seguir.

- **Tempo de processamento**: mede a velocidade com que um *matcher* executa em um dado cenário e ambiente de execução. O resultado é o tempo total despendido para obtenção dos alinhamentos entre duas ontologias, podendo ser medido em segundos ou minutos.

- **Consumo de Memória**: mede a quantidade de memória RAM utilizada para executar todo o processo de *match*. O resultado é geralmente representado em MB (*Megabytes*).

Outra medida de desempenho apresentada por Euzenat, Ehrig e Castro (2005) diz respeito à escalabilidade. Esta pode ser avaliada tanto por meio de estudos teóricos quanto por sequências de *benchmarks* complexos, tendo pouca utilização prática.

5.2 CENÁRIO EXPERIMENTAL

O OntoGenesis foi implantado em um ambiente contendo um serviço de dados que fornece representações sintáticas sobre uma determinada entidade de um domínio. O *Semantic Adapter* foi anexado ao serviço a fim de interceptar as requisições e se comunicar com o OntoGenesis *API* e *Engine* para realizar o seu enriquecimento semântico. Como fontes externas, foram utilizados vocabulários amplamente conhecidos, bem como conjuntos de dados semânticos descritos por ontologias públicas. Todos os dados utilizados nos experimentos são reais e foram coletados a partir de fontes abertas de diferentes organizações. Com isso, foram planejados dois experimentos, detalhados a seguir, para avaliar as medidas de conformidade e desempenho.

- **Experimento 1 – Domínio Científico**

No primeiro experimento, o serviço fornece metadados referentes a publicações científicas. Os metadados são obtidos de um *dataset* disponibilizado pelo DBLP[2] em formato XML sem informação semântica. O serviço de dados, portanto, fornece diversas informações referentes à base do DBLP, como autores, título, ano, editor, nome do periódico/evento, editores, etc. Como fontes externas ao OntoGenesis, foram incorporados conjuntos de dados semânticos do mesmo domínio do serviço de dados. Assim, foram utilizados *datasets* do ScholarlyData[3] juntamente com a ontologia que descreve os seus dados[4], além dos vocabulários abertos do FOAF[5] (*Friend of a Friend*) e do Dublin Core[6].

[2] *Dataset* do DBLP: <https://dblp.uni-trier.de/xml/>.
[3] *Dataset* do ScholarlyData: <http://www.scholarlydata.org/dumps/>.
[4] *Conference Ontology*: <http://www.scholarlydata.org/ontology/doc/>.
[5] Especificação FOAF: <http://xmlns.com/foaf/spec/>.
[6] Dublin Core Metadata Initiative: <http://dublincore.org/specifications/>.

O ScholarlyData (NUZZOLESE et al., 2016) é uma versão refatorada do *Semantic Web Dog Food* (SWDF) – um *dataset* semântico que contém metadados de artigos, pessoas, organizações e eventos relacionados a conferências acadêmicas e tido como referência pela comunidade de Web Semântica. Portanto, ele possui metadados similares aos providos pelo DBLP, o que potencializa a eficácia do OntoGenesis em identificar correspondências entre a ontologia gerada para o DBLP e as ontologias externas. Metadados de mais de 10 eventos da área de semântica estão contemplados no ScholarlyData, como o ESWC, ISWC, WWW, dentre outros. Tais metadados são descritos pela *Conference Ontology* (NUZZOLESE et al., 2016), uma versão aprimorada da *Semantic Web Conference Ontology* e utilizada para modelar conhecimento sobre conferências acadêmicas. Já o FOAF é um vocabulário que descreve pessoas, suas atividades e suas relações com outras pessoas e/ou objetos, enquanto o Dublin Core define um conjunto de metadados para catalogar itens bibliográficos e descrever recursos físicos e digitais.

- **Experimento 2 – Domínio Governamental**

O Experimento 2 é baseado em dados abertos governamentais publicados pela SSP/SP, que recentemente passou a divulgar em seu portal online[7] os boletins de ocorrência (B.O.) registrados pelo órgão. São publicadas informações sobre o local do incidente, data, horário, bem como dados dos envolvidos (vítimas e autores do crime), como nome, número do documento, data de nascimento, nacionalidade, local de nascimento, gênero, cútis, entre outros. A despeito de ser uma iniciativa importante de transparência das informações para a sociedade, a SSP/SP tem publicado as informações de forma incompleta e em um formato inadequado para serem analisadas ou até mesmo integradas a outras fontes de dados. Os boletins são publicados em páginas HTML sem uma padronização de estrutura e sem nenhum tipo de informação semântica associada. Ademais, detalhes acerca do histórico das ocorrências são omitidos e funcionalidades básicas de busca ou filtragem inexistem.

Em vista disso, foi desenvolvida uma ferramenta[8] capaz de recuperar as páginas HTML publicadas no portal da SSP/SP e extrair as informações em um formato mais apropriado. Como resultado, para cada B.O. processado, a ferramenta gera um documento no formato JSON – mais facilmente consumido por outros sistemas.

[7] Portal da SSP/SP: <http://www.ssp.sp.gov.br/transparenciassp/>.
[8] Disponível em: <https://dx.doi.org/10.6084/m9.figshare.3856074>.

Essa ferramenta foi implementada em um escopo mais amplo, cujo objetivo foi realizar um *mashup* dos dados dos boletins com outras fontes disponíveis na Web e de outros órgãos governamentais. Informações adicionais sobre a ferramenta e as análises realizadas sob estes dados podem ser encontradas em (OLIVEIRA et al., 2016).

Foram selecionadas duas categorias de B.O. registrados no ano de 2015 para serem extraídos pela ferramenta e utilizados no Experimento 2: homicídio doloso e morte suspeita. Tais categorias foram escolhidas em virtude da grande atenção dada a esses tipos de crime/morte. Foi implementado, então, um serviço de dados, cuja fonte é o conjunto de documentos JSON (obtidos como resultado da ferramenta), contendo uma interface com filtros de acesso aos dados dos boletins e das pessoas envolvidas. O Experimento 2, portanto, objetiva enriquecer semanticamente tais informações a fim de publicar os dados da SSP/SP de maneira adequada e de facilitar o seu reúso.

Como fontes externas ao OntoGenesis, foram utilizados *datasets* do DBpedia[9] e do Geonames[10], além do vocabulário FOAF. O DBpedia é enorme banco de dados que oferece diversas informações semânticas extraídas da Wikipedia. Já o Geonames trata-se um banco de dados geográfico que contém mais de 160 milhões de triplas RDF disponíveis na Web e também inclui um vocabulário para representar informações sobre lugares, como latitude, longitude, nome do local, etc. Dois subconjuntos de dados do Geonames foram utilizados em razão do potencial de intersecção com os dados dos boletins de ocorrência: nome de países e nome de locais do Brasil. Com relação ao DBpedia, foi selecionado o *dataset* Person-Data, o qual contém mais de 8 milhões de triplas representando atributos de pessoas, como nome, data de nascimento, país, etc. Ademais, foram utilizadas regras heurísticas para a cútis e o gênero.

5.3 METODOLOGIA

Para avaliar o comportamento da abordagem proposta, foram estabelecidos três valores diferentes para um *threshold* α da força de equivalência das propriedades: 0.4, 0.6 e 0.8. Para fins experimentais, o tamanho do *buffer* de representação foi configurado como 0, ou seja, o OntoGenesis deve processar cada solicitação por vez. Ao receber uma representação do serviço de dados, o OntoGenesis

[9] *Datasets* do DBpedia: <http://wiki.dbpedia.org/Downloads2015-10>.
[10] *Datasets* do GeoNames: <http://download.geonames.org/export/dump/>.

extrai todos os valores, verifica a sobreposição de termos com fontes externas e, por fim, atualiza a ontologia do serviço de dados com as novas propriedades e as equivalências identificadas, de acordo com o *threshold* α configurado. Posto que este trabalho considera como correspondências as equivalências de propriedades, é possível resumir a equivalência em uma tripla $E = \{p, p', n\}$, onde p e p' são as propriedades da ontologia O gerada e da ontologia externa O', respectivamente, e n expressa a força da equivalência de tais propriedades. Vale ressaltar que para a equivalência ser adicionada a O, o valor de n deve ser maior que o *threshold* definido, isto é, $\langle p \texttt{ owl:equivalentProperty } p' \rangle \in O \Leftrightarrow n \geq \alpha$.

O serviço de dados implementado para o Experimento 1 fornece ao OntoGenesis, por meio do *Semantic Adapter*, representações JSON que podem conter até nove atributos relacionados a um artigo (*author*, *editor*, *title*, *booktitle*, *year*, *series*, *address*, *journal* e *publisher*). Os primeiros cinco atributos possuem uma ou mais propriedades equivalentes em fontes externas, enquanto os demais não possuem nenhum tipo de correspondência. Portanto, estes devem estar associados a uma propriedade do tipo *Datatype* na ontologia gerada, porém sem propriedade equivalente.

Já o serviço de dados implementado para o Experimento 2 fornece ao OntoGenesis representações JSON que podem conter até dez atributos de uma pessoa (RG, primeiro nome, nome completo, sexo, cútis, data de nascimento, naturalidade, nacionalidade, profissão e instrução). Nove desses atributos possuem propriedades equivalentes em fontes externas e apenas o grau de instrução não possui correspondências. Este, portanto, deve estar associado a uma propriedade do tipo *Datatype* na ontologia, porém sem propriedade equivalente. É importante destacar que todos os atributos das representações fornecidas pelo serviço foram rotulados em português, literalmente como consta na fonte de dados da SSP/SP, enquanto as propriedades de fontes externas permaneceram em inglês.

Para validar os alinhamentos gerados, foram observados os *datasets* utilizados e foi criado manualmente um alinhamento de referência para cada experimento. Assim, para cada requisição, foram comparadas as equivalências geradas com as equivalências de referência e foram então calculadas precisão, cobertura e *F-Measure*.

Os experimentos foram conduzidos utilizando dados aleatórios de artigos (Experimento 1) e de pessoas envolvidas (Experimento 2), com 7 rodadas de 100 requisições enviadas ao OntoGenesis pelo serviço de dados para cada um dos *thresholds* definidos. O número de rodadas e de requisições demonstrou-se satisfatório de

acordo com os resultados obtidos por meio de um experimento piloto, conduzido com o intuito de identificar um tamanho adequado dos *datasets* utilizados (conforme detalhado na seção a seguir).

Todos os experimentos foram realizados em um computador com um processador Intel i7 2.5GHz, equipado com 8GB de memória, com Ubuntu 16.04, usando o Oracle Java Development Kit (JDK) 8. Algumas bibliotecas/*frameworks* disponíveis para a plataforma Java também foram integradas à solução, a exemplo do Apache Jena 3.3.0 para o processamento e manipulação de triplas RDF e de ontologias, do Spring Framework, e do Jersey RESTful Web Services Framework[11] (uma implementação da especificação JAX-RS para desenvolvimento de serviços que adotam o estilo REST). Como banco de dados empregado pelo *Index Repository*, foi usado o MapDB[12], um banco de dados *open source* do tipo chave-valor.

A fim de possibilitar a replicação dos experimentos, os códigos-fonte, bem como as instruções de configuração e os resultados obtidos estão disponíveis em repositórios públicos[13].

5.4 RESULTADOS OBTIDOS

As próximas subseções discutem os resultados obtidos dos experimentos realizados. A subseção 5.4.1 apresenta um experimento piloto conduzido a fim de estabelecer um tamanho adequado para os *datasets* utilizados. Já as subseções 5.4.2 e 5.4.3 analisam as medidas de conformidade e de desempenho, respectivamente.

5.4.1 Tamanho dos conjuntos de dados

Para fins de avaliação, foi selecionado, para o Experimento 1, um *subset* do DBLP que contempla apenas os metadados (sintáticos) dos artigos publicados nos mesmos eventos contidos no *dataset* do ScholarlyData. Tal estratégia foi adotada por duas razões principais: 1) reduzir o volume de dados do DBLP geridos pelo serviço, visto que o seu *dump* original contém mais de 2GB, e 2) obter dados com maior potencial de intersecção com as fontes externas – a fim de aumentar a proporção de coocorrências entre os diferentes *datasets* e obter análises mais precisas das medidas de conformidade. Quanto às fontes externas, o Experimento 1 manteve todos os da-

[11] Disponível em: <https://jersey.github.io/>.
[12] Disponível em: <http://www.mapdb.org/>.
[13] https://github.com/brunocnoliveira/iiwas2017-ontogenesis-experiments
https://github.com/brunocnoliveira/dblp-scholarly-ontogenesis-experiments

Tabela 2 – Resultados para o tamanho do *Subset* do DBpedia.

Tamanho do *Subset*	**Média dos tempos com 95% de IC**	**Média dos maiores F-Measures**
20%	103.84 ± 5.70 ms	0.7097
40%	121.25 ± 7.30 ms	0.7503
60%	154.86 ± 12.99 ms	0.6542

dos disponibilizados pelo ScholarlyData, resultando em um *Index Repository* aproximadamente 27% maior que o do Experimento 2.

Com relação ao Experimento 2, o *dataset* de pessoas do DBpedia teve de ser reduzido, visto que o tamanho do seu *dump* (de aproximadamente 2GB) tende a afetar significativamente o consumo de memória. Com o objetivo de identificar um tamanho de *subset* do DBpedia apropriado para fins de avaliação, foi realizado um experimento piloto utilizando 20%, 40% e 60% dos termos mais frequentes do DBpedia. Quatro rodadas de execução de 100 requisições foram feitas para cada tamanho (20, 40 e 60%) usando as mesmas configurações detalhadas na seção 5.3 com o *threshold* da força de equivalência configurado para 0.8. No total, para esse piloto, foram realizadas 12 execuções com 100 requisições cada uma.

A Tabela 2 apresenta o tempo médio de cada requisição feita pelo serviço de dados ao OntoGenesis, juntamente com seus intervalos de confiança (IC); e a média das maiores *F-Measures* (dentro das 100 requisições) calculadas para cada rodada de execução. Os resultados mostram que o tamanho do *subset* tem pouco impacto nos tempos de processamento. Observando os intervalos de confiança, percebe-se que o tempo pode variar menos que 13 ms. Isso acontece porque a quantidade de processamento a ser executado é proporcional ao número de atributos da representação. Algumas representações do conjunto de dados da SSP/SP têm apenas uma fração dos 10 atributos possíveis, sendo processadas mais rapidamente do que as representações com mais atributos.

Além disso, nenhuma diferença significativa foi observada entre as *F-Measures* computadas para cada tamanho de *subset*, variando aproximadamente 0.1 ou menos. No entanto, uma vez que mais falsos positivos foram encontrados no maior *subset*, uma propriedade equivalente imprecisa foi identificada, o que leva a uma ligeira diminuição do *F-Measure* para o *subset* com 60%. A fim de manter um tamanho do *subset* adequado e um maior *F-Measure*, foram utilizados os 40% dos termos mais frequentes do DBpedia

para a realização do Experimento 2.

Por fim, esse experimento piloto serviu de base para definir a quantidade de requisições e rodadas de execuções explicitadas na metodologia (seção 5.3). Visto que os resultados relacionados ao tempo de processamento variaram em torno de 6% e o *F-Measure* teve uma oscilação inferior a 0.1, foram mantidas as 100 requisições e as rodadas experimentais foram incrementadas de 4 para 7.

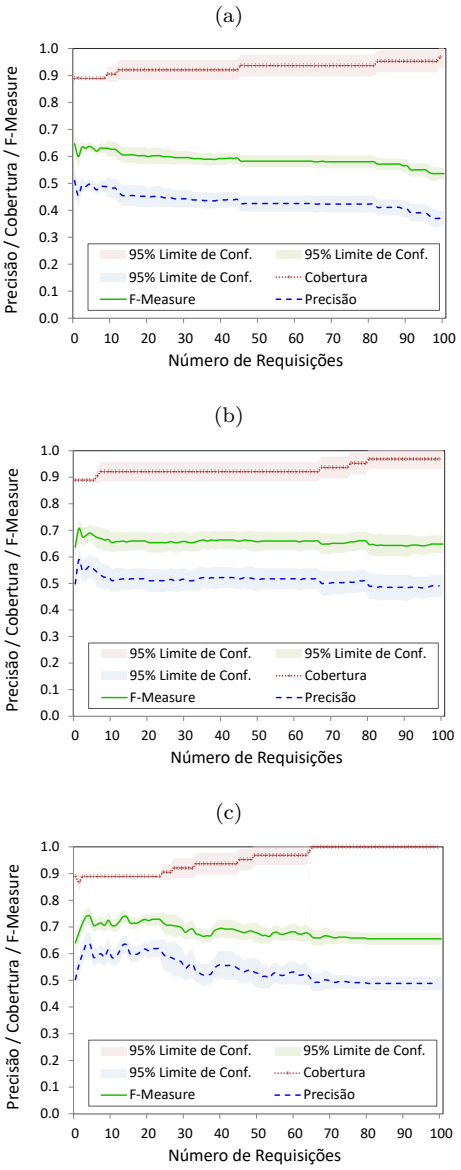
5.4.2 Análise de conformidade

A Figura 22 apresenta a curva média de precisão, cobertura e *F-Measure* do Experimento 1 por requisição, considerando as 7 rodadas de execução e seus intervalos de confiança a 95% na área sombreada ao redor das linhas. Com o *threshold* definido para 0.4 (Figura 22 (a)), é possível observar um decréscimo da precisão e um aumento tênue no *recall*. No cenário com $\alpha = 0.6$ (Figura 22 (b)), há uma estabilidade da precisão, alcançando níveis maiores de precisão apenas a partir da 80ª requisição, aproximadamente. Já na Figura 22 (c) (com *threshold* $\alpha = 0.8$), observa-se uma maior variação na precisão e um crescimento mais notório do *recall*, alcançando 1.0 com intervalo de confiança de 0 a partir da 66ª requisição.

Alguns fatores explicam a variação inicial da precisão acompanhada pela sua leve queda ilustrada na Figura 22 (c). Propriedades como `givenName` e `familyName`, tanto do FOAF quanto da *Conference Ontology*, são identificadas pelo OntoGenesis, em alguns casos, como equivalentes à propriedade `author` da ontologia criada para os dados do DBLP. Contudo, tais equivalências são tidas como falso positivas, reduzindo, assim, a precisão. Apenas `foaf:name` e `cofunc:name` são aceitas como equivalentes de `author`.

Além disso, existe o fato de propriedades da ontologia gerada para os dados do DBLP possuírem mais de uma equivalência com propriedades das ontologias que descrevem o ScholarlyData. Isto se deve ao fato de os metadados das conferências no ScholarlyData estarem associados a mais de uma ontologia. Por exemplo, um autor de um artigo do ESWC é descrito pela *DatatypeProperty* `cofunc:name` da *Conference Ontology*, enquanto um autor no ICWS é descrito pela propriedade `foaf:name` do FOAF. Já outras conferências usam a propriedade `dc:creator` do Dublin Core. Ademais, existe o fato de que a propriedade `cofunc:name` pode descrever tanto o nome de um autor quanto o nome de um evento. Isto porque tal propriedade possui domínio em `Agent` – que é uma superclasse de `Organisation` e `Person` – o que tende a reduzir a precisão.

Figura 22 – Experimento 1: curvas de precisão, cobertura e *F-Measure* utilizando *threshold* da força de equivalência (a) $\alpha = 0.4$ (b) $\alpha = 0.6$ e (c) $\alpha = 0.8$.



Fonte: Elaborado pelo autor (2018).

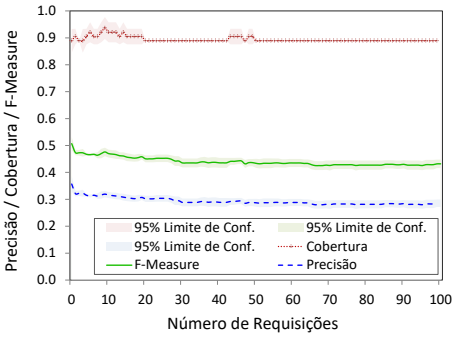
Com relação ao Experimento 2, a Figura 23 apresenta a curva média de precisão, cobertura e *F-Measure* por requisição, considerando as 7 rodadas de execução e seus intervalos de confiança a 95% na área sombreada ao redor das linhas. A Figura 23 (a) mostra que no cenário com um *threshold* da força de equivalência configurado para 0.4, o *recall* mantém-se estável, alcançando quase 0.9 após 20 requisições, enquanto há uma leve diminuição da precisão para 0.3. No cenário com um *threshold* de 0.6 (Figura 23 (b)), o *recall* aumenta antes de 10 requisições serem executadas, enquanto a precisão aumenta suavemente, chegando a quase 0.4. Por outro lado, no cenário com um *threshold* de 0.8 (Figura 23 (c)), as medidas aumentam significativamente, obtendo uma precisão de quase 0.6, com *recall* e *F-Measure* de aproximadamente 0.85 e 0.7, respectivamente.

Para todos os *thresholds* avaliados, após algum tempo, o OntoGenesis foi capaz de enriquecer, com propriedades equivalentes, 8 das 9 propriedades que possuíam alguma equivalência (aproximadamente 0.89 de *recall*). Além disso, foram encontradas equivalências falso positivas. A razão por trás disso é que existem algumas propriedades diferentes compartilhando os mesmos termos frequentes. Por exemplo, `service:nome` (propriedade da ontologia gerada para o serviço) tem correspondências com `dbpedia:name`, `dbpedia:surName` e `dbpedia:givenName`. No entanto, consideramos apenas `dbpedia:name` como a correspondência correta. Da mesma forma, `service:naturalidade` (propriedade relacionada ao local de nascimento do envolvido) teve correspondências com `dbpedia:name` e `geo:name`, o que justifica a baixa precisão alcançada nos experimentos com *thresholds* menores. A razão de o OntoGenesis ter definido uma equivalência para esta última propriedade é que existem diversos lugares no Brasil que possuem nomes de pessoas. Essas situações podem ser tratadas incrementando o *threshold*, visto que quanto maior o seu valor, maior a precisão. Entretanto, é importante definir um valor de *threshold* equilibrado a fim de manter um valor de *recall* adequado, caso contrário, este tende a diminuir.

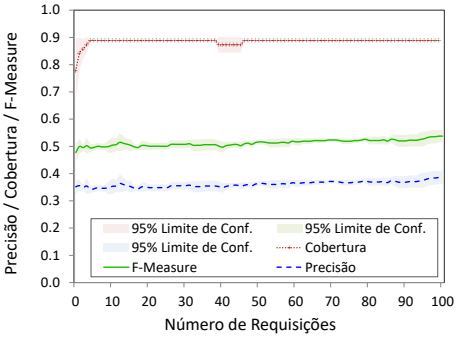
O *recall* e a precisão também estão relacionados com a sobreposição de dados das representações do serviço e das fontes externas. Nas primeiras requisições com o *threshold* $\alpha = 0.8$, observa-se uma ligeira diminuição do *recall*, já que muitos valores de propriedades do serviço não coincidem com os do DBpedia e Geonames, o que leva a uma força de equivalência $n \leqslant \alpha$. Portanto, em cenários com *thresholds* mais altos, a força tende a se equilibrar à medida que novos valores são passados ao OntoGenesis *Engine*.

Figura 23 – Experimento 2: curvas de precisão, cobertura e *F-Measure* utilizando *threshold* da força de equivalência (a) $\alpha = 0.4$ (b) $\alpha = 0.6$ e (c) $\alpha = 0.8$.

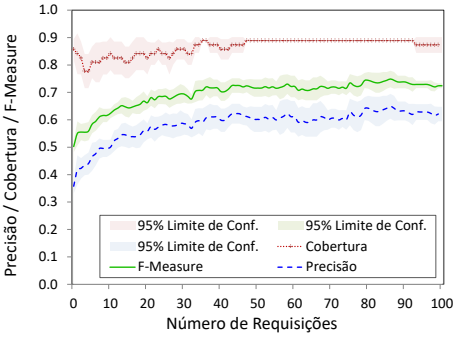
(a)



(b)



(c)



Fonte: Elaborado pelo autor (2018).

5.4.3 Análise de desempenho

As Tabelas 3 e 4 apresentam o tempo médio de processamento dos Experimentos 1 e 2, respectivamente, juntamente com seus respectivos intervalos de confiança. O processamento foi dividido em quatro etapas: geração dos mapeamentos semânticos (SM) e construção da ontologia; processo de *matching* de propriedades; geração das representações em JSON-LD; e enriquecimento da descrição Hydra. Foram considerados os resultados observados nas 7 rodadas de execução, para cada valor de *threshold* α avaliado. Percebe-se que não há alteração significativa no tempo de execução entre os *thresholds* empregados. Ademais, o tempo de processamento é independente do número de requisições, pois as requisições anteriores não são reprocessadas, sendo recalculado somente o valor da força das propriedades equivalentes.

Adicionalmente, a Figura 24 mostra um *boxplot* para o tempo médio (entre as 7 rodadas) de processamento de cada uma das 100 requisições. Esse gráfico representa o quartil de 50% em torno da mediana e o intervalo interquartil de 1.5 para *outliers*. Enquanto a média observada para $\alpha = 0.8$ é ligeiramente maior, as três distri-

Tabela 3 – Experimento 1: Tempo de processamento (ms) com 95% de intervalo de confiança.

α^*	**Map. Sem. + Ontologia**	***Matching* de Propriedades**	**Geração Represent.**	**Enriq. Descrição**
0.4	19.74 ± 0.85	28.91 ± 1.77	4.15 ± 0.38	1.03 ± 0.02
0.6	20.44 ± 0.97	28.30 ± 1.98	4.04 ± 0.35	0.95 ± 0.02
0.8	20.76 ± 1.03	29.87 ± 2.19	4.25 ± 0.33	1.19 ± 0.03

*α = *Threshold* da força de equivalência

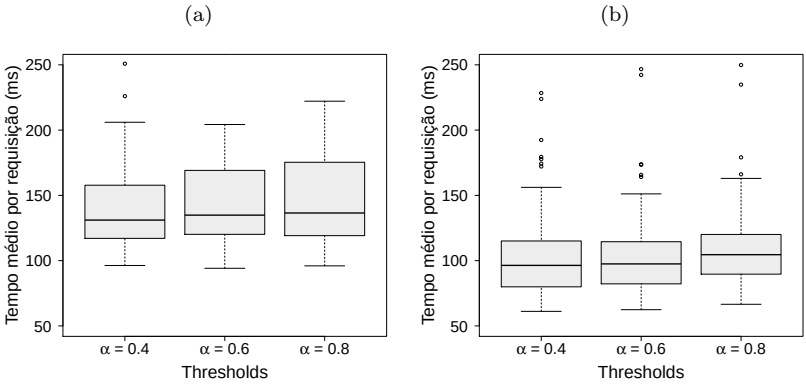
Tabela 4 – Experimento 2: Tempo de processamento (ms) com 95% de intervalo de confiança.

α^*	**Map. Sem. + Ontologia**	***Matching* de Propriedades**	**Geração Represent.**	**Enriq. Descrição**
0.4	19.10 ± 1.17	19.57 ± 2.48	3.57 ± 0.32	1.21 ± 0.02
0.6	17.98 ± 0.97	18.05 ± 1.42	3.39 ± 0.26	1.12 ± 0.02
0.8	19.58 ± 1.08	21.22 ± 1.75	4.23 ± 0.35	1.19 ± 0.03

*α = *Threshold* da força de equivalência

buições de amostragem se sobrepõem, sugerindo que, na prática, o efeito do *threshold* no tempo de processamento é pouco significante.

Figura 24 – Análise do tempo das requisições para: (a) Experimento 1 e (b) Experimento 2.



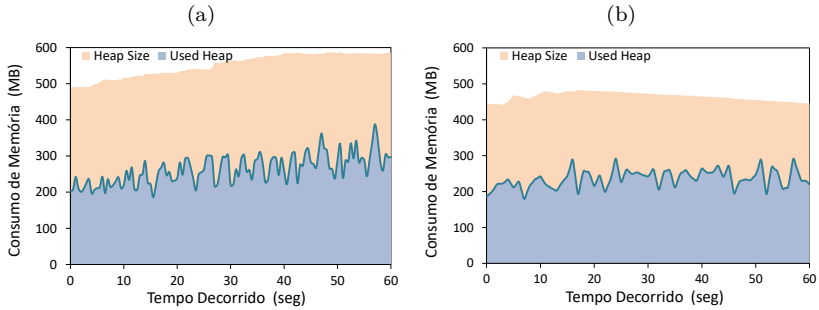
Fonte: Elaborado pelo autor (2018).

Ao contrário das medidas de conformidade, as medidas de desempenho dependem do ambiente no qual os experimentos são executados. Em vista disso, as mesmas configurações de ambiente descritas anteriormente foram usadas. Porém, para analisar a média de consumo de memória e CPU, a metodologia experimental foi adaptada de modo que um consumidor envie requisições aleatórias ao serviço de dados por um período determinado. Neste caso, o tempo de execução é fixo, e não mais a quantidade de requisições. Assim, as variáveis de consumo de memória e CPU foram monitoradas durante 60 segundos após uma fase de *preheating* (pré-aquecimento) do OntoGenesis. Tais medidas foram observadas na JVM (*Java Virtual Machine*) na qual o OntoGenesis foi executado.

A fase inicial de *preheating* consistiu no disparo de várias requisições durante 5 segundos e teve como objetivo minorar os efeitos introduzidos pelo compilador JIT (*Just-In-Time*) do Java. Vale frisar que o consumo de recursos relacionados à configuração do OntoGenesis (incluindo o carregamento das fontes externas e a construção dos índices) não é contabilizado neste experimento.

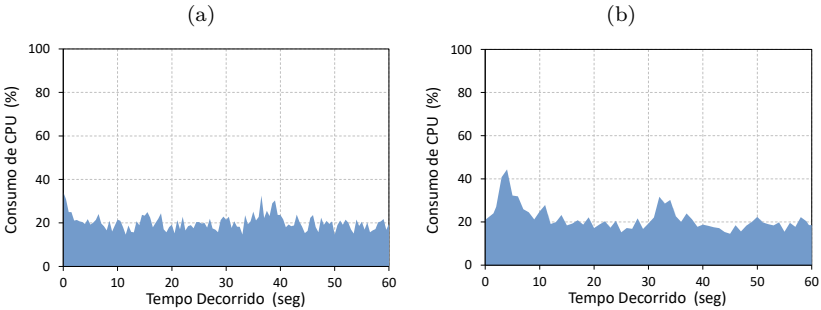
As Figuras 25 e 26 apresentam o consumo médio de memória e de CPU, respectivamente, considerando as 7 rodadas de execução. Como pode ser visto na Figura 25 (a), a média do tamanho de

Figura 25 – Consumo médio de memória do OntoGenesis para: (a) Experimento 1 e (b) Experimento 2.



Fonte: Elaborado pelo autor (2018).

Figura 26 – Consumo médio de CPU do OntoGenesis para: (a) Experimento 1 e (b) Experimento 2.



Fonte: Elaborado pelo autor (2018).

memória *Heap* utilizada no Experimento 1 chega a quase 400MB, apresentando uma leve tendência de crescimento no consumo. Já a Figura 25 (b) mostra que quantidade média de memória *Heap* utilizada no Experimento 2 chega a 300MB e há poucas variações ao longo do tempo. Como consequência do tamanho do índice do Experimento 1 ser maior que o do Experimento 2, nota-se um maior consumo de memória no primeiro caso.

Os experimentos realizados consideram requisições síncronas de um único cliente, o que evita a ocupação total de CPU pelo processo do *Engine*, já que uma parte significativa do tempo é dedi-

cada à comunicação síncrona HTTP com o serviço e provavelmente ocorre enquanto o *Engine* não é executado pelo escalonador do sistema operacional. Portanto, o consumo médio da CPU atinge até 40% e permanece abaixo desse valor durante a execução. Visto que esse é um valor médio, observou-se que houve naturalmente picos de uso de CPU, embora, durante parte do tempo, o consumo de CPU tenha ficado abaixo de 20%.

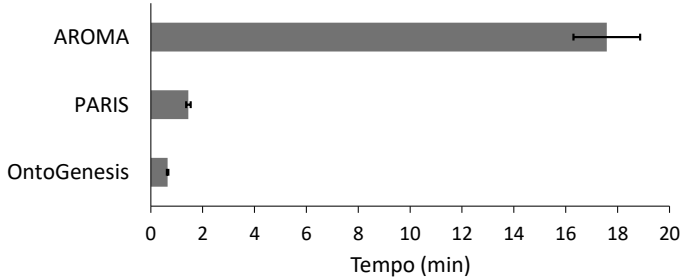
5.5 COMPARATIVO COM OUTRAS ABORDAGENS

As técnicas de alinhamento extensional podem ser utilizadas no âmbito de serviços Web semânticos (SALVADORI et al., 2017a) e, de maneira similar ao OntoGenesis, exploram as instâncias das ontologias, que descrevem os dados fornecidos por serviços, para identificar possíveis correspondências ontológicas. Os *matchers* extensionais identificam as correspondências entre diferentes ontologias considerando todo o indivíduo da ontologia, diferentemente do OntoGenesis que utiliza os indivíduos para inferir correspondências entre as propriedades de acordo com a intersecção dos seus valores, ou seja, propriedades com identificadores correspondentes ou com valores no mesmo domínio são consideradas equivalentes.

Para comparar a abordagem proposta neste trabalho com *matchers* extensionais tradicionais, o cenário de enriquecimento semântico de serviços de dados descrito neste capítulo foi adaptado para um cenário de alinhamento de ontologias. O objetivo desse experimento é alinhar a ontologia construída pelo OntoGenesis (sem alinhamentos) com as ontologias do DBpedia, Geonames e o vocabulário do FOAF. Para este fim, dois arquivos foram gerados e passados como entrada para os *matchers*. O primeiro contém a ontologia construída pelo OntoGenesis juntamente com 1.021 instâncias (de pessoas envolvidas em um boletim de ocorrência) descritas nessa ontologia. O segundo arquivo agrupa as ontologias externas e as instâncias do DBpedia e GeoNames. Foram avaliados dois *matchers* extensionais, PARIS (SUCHANEK; ABITEBOUL; SENELLART, 2011) e AROMA (DAVID; GUILLET; BRIAND, 2007), replicando o experimento 7 vezes para cada um deles.

Como resultado, tanto o PARIS quanto o AROMA não produziram alinhamentos para as propriedades das ontologias. Como os *matchers* extensionais se alicerçam na coocorrência de indivíduos entre as ontologias, nenhuma correspondência foi obtida, diferentemente da abordagem proposta nesta dissertação, em que o alinhamento se baseia no compartilhamento dos valores das propriedades.

Figura 27 – Comparativo dos tempos de processamento.



Fonte: Elaborado pelo autor (2018).

O desafio para esses *matchers*, bem conhecidos pela comunidade de alinhamento de ontologias, é a ausência de compartilhamento de indivíduos entre os dados providos pelo serviço e os dados de fontes externas. Observou-se, por exemplo, que o único valor literal compartilhado entre os dados fornecidos pelo serviço (referente às pessoas envolvidas) e os dados das fontes externas (pessoas do DBpedia e lugares do GeoNames) é "Preta". Este valor diz respeito à cor da pele de uma pessoa, além de ser também o nome de um lugar existente no *dataset* do GeoNames (<http://www.geonames.org/3391216>). Estendendo a comparação para Levenshtein com $n = 1$, há 2.324 (0,20%) indivíduos nos dados externos que compartilham um único valor literal com os dados do serviço. Entretanto, nenhum desses objetos compartilha o valor de mais de uma propriedade com o conjunto de dados fornecido pelo serviço, o que se torna um desafio para os *matchers* extensionais que se baseiam na similaridade de um conjunto de propriedades para inferir a equivalência.

Além disso, os resultados também não foram favoráveis em relação ao desempenho. Em média, o AROMA levou quase 18 minutos, o PARIS levou 1 minuto e 45 segundos e o OntoGenesis gastou em torno de 39 segundos. A Figura 27 ilustra esses tempos de processamento, considerando o intervalo de confiança das 7 execuções.

Outra abordagem similar ao OntoGenesis foi proposta por Yao et al. (2014), conforme discutido no Capítulo 2. Os autores, no entanto, não forneceram os dados utilizados nos experimentos, o que dificulta uma comparação mais assertiva. Por outro lado, com base nos resultados apresentados no artigo, é possível inferir que

a abordagem proposta pelos autores é capaz de processar, no melhor caso, 2.038 triplas por segundo. Em contraste, o OntoGenesis processou aproximadamente 96.300 triplas/seg no cenário adaptado que utilizou os dados da SSP/SP, DBpedia e GeoNames.

6 CONCLUSÕES

Os serviços de dados têm sido cada vez mais utilizados em decorrência da popularização das arquiteturas orientadas a serviços (SOA). Contudo, as fontes de dados acessadas por esses serviços, em geral, armazenam os dados sem explicitar a sua semântica, o que dificulta o reúso e a integração eficiente dos dados. Os serviços de dados semânticos (SDS) surgem para superar tais dificuldades, empregando as tecnologias da Web Semântica com o intuito de prover significado aos dados e promover a sua compreensão por outros agentes de *software*, o que, em última análise, tende a melhorar a capacidade de reutilização e a interoperabilidade dos serviços.

Embora a literatura apresente diversos benefícios oriundos do uso de SDS , como a descoberta e composição de serviços, a sua efetiva implementação e ampla adoção ainda são limitadas. Visto que os dados são armazenados geralmente sem vínculo semântico, faz-se necessário construir uma ontologia de domínio que descreva os dados, além de estabelecer as associações semânticas entre a ontologia e o serviço. Não obstante, para de fato alcançar a interoperabilidade e o reúso dos serviços, é importante que os conceitos da ontologia criada estejam alinhados a ontologias já existentes e bem conhecidas. Isso traz um novo desafio para o desenvolvimento e a integração de SDS, devido ao grau de complexidade e o tempo despendido para executar tais tarefas. Além disso, a eficácia das ferramentas disponíveis para construção e alinhamento de ontologias depende da disponibilidade de um volume significativo de dados, o que é inviável quando o seu acesso é limitado através de uma interface de serviço que fornece os dados sob demanda.

Analisando os trabalhos propostos na literatura acerca do enriquecimento semântico de serviços Web, observou-se que, além de requererem um alto grau de intervenção humana, as abordagens propostas focam, majoritariamente, na geração de descrições semânticas das interfaces dos serviços, dando pouca atenção às representações dos dados por eles fornecidos. Ademais, os trabalhos mais relevantes no âmbito de enriquecimento semântico de serviços que abordam a construção de ontologias não preveem o seu alinhamento com outras ontologias já existentes.

Para superar os desafios mencionados, foi apresentado nesta dissertação um mecanismo para enriquecer dinamicamente serviços de dados, de tal modo que estes sejam capazes de fornecer descrições e representações associadas a conceitos semânticos. Alinhado a esse mecanismo, foi proposta uma arquitetura, denominada On-

toGenesis, a qual visa a construção e evolução de ontologias de domínio que descrevem os dados fornecidos pelos serviços, e a identificação de relações de equivalências entre as propriedades da ontologia criada com propriedades existentes em ontologias externas. Tais equivalências são adicionadas à ontologia e tanto a descrição quanto as representações fornecidas pelo serviço são associadas aos conceitos dessa ontologia. Diferentemente das abordagens encontradas na literatura, o OntoGenesis destina-se a enriquecer semanticamente serviços de dados com a mínima intervenção humana. Assim, espera-se abstrair, do ponto de vista dos desenvolvedores de *software*, o processo oneroso de enriquecer semanticamente serviços conectados a fontes que armazenam dados sem informação semântica, fomentando a execução de análises de dados mais sofisticadas.

Uma das principais dificuldades encontradas para avaliar a abordagem proposta foi a identificação de dados abertos sintáticos que compartilham valores de propriedades com *datasets* semânticos. Ainda assim, neste trabalho foram conduzidos alguns experimentos utilizando dados abertos. O primeiro experimento realizado buscou enriquecer um serviço de dados que fornece metadados sobre publicações disponíveis no DBLP, utilizando conceitos definidos na *Conference Ontology* e utilizados pelos *datasets* do ScholarlyData. Já o segundo experimento confrontou dados abertos disponibilizados pela SSP/SP sem semântica com dados do DBpedia e GeoNames. Os resultados mostram a aplicabilidade da proposta e revelam que são alcançados níveis adequados de conformidade (precisão e cobertura). Ademais, ganhos de desempenho também foram observados para o OntoGenesis em um experimento que o comparou com abordagens similares encontradas na literatura.

O mecanismo de enriquecimento semântico de serviços de dados, juntamente com a arquitetura do OntoGenesis e sua avaliação experimental, formam os principais artefatos que atingem os objetivos definidos na introdução deste trabalho. Como contribuições acadêmicas, foram publicados três artigos frutos desta dissertação. Publicações adicionais no âmbito de composição de serviços e Web Semântica também foram alcançadas em colaboração com outros estudantes. A lista dos artigos pode ser conferida no Apêndice A.

6.1 LIMITAÇÕES E TRABALHOS FUTUROS

Algumas limitações derivam do escopo desta pesquisa. Embora os resultados obtidos neste trabalho tenham sido considerados satisfatórios, as análises realizadas estão estreitamente relacionadas

aos *datasets* disponibilizados pela SSP/SP e DBLP, bem como as ontologias e as instâncias extraídas do DBpedia, GeoNames e ScholarlyData. Portanto, parte das conclusões a respeito dos resultados, em especial das medidas de conformidade, foi feita em torno desses conjuntos de dados. Assim, para explorar ainda mais o potencial do OntoGenesis, sugere-se avaliar o seu uso integrado a outras ferramentas e em novos cenários com diferentes *datasets*.

Outras questões também podem ser objeto de melhorias em trabalhos futuros. Além da distância de Levenshtein, é possível incorporar outras funções de similaridade durante a comparação dos termos das propriedades, como Jaccard, Jaro–Winkler, etc. Adicionalmente, a equação da força de equivalência pode ser aprimorada de modo a considerar também a frequência de cada termo associado às propriedades, visando à redução dos falsos positivos sem introduzir falsos negativos.

Uma questão ainda em aberto é como avaliar e validar formalmente e com mínima intervenção humana a ontologia primária construída pelo OntoGenesis. Complementarmente, um mecanismo para identificar a equivalência de classes, além das propriedades equivalentes, também pode ser explorado em trabalhos posteriores.

Outras técnicas de *matching* de ontologias podem ser combinadas com o intuito de gerar melhores resultados. Isso se deve ao fato de cada técnica possuir características distintas, permitindo que elas sejam usadas de forma complementar. Por exemplo, um *matcher* baseado no WordNet é capaz de encontrar correspondências que um *matcher* baseado em nomes, o qual utiliza comparações puramente léxicas, teria dificuldade de identificar. Neste caso, seria possível distinguir propriedades que possuem o mesmo *range* (valor associado), porém possuem semântica diferente.

Por fim, um trabalho futuro promissor diz respeito a uma abordagem para filtrar fontes de dados específicas – durante a busca de correspondências – em consonância com o domínio do serviço de dados. Apesar de essa proposição estar fora do escopo da presente pesquisa, a consideremos uma contribuição relevante para aprimorar a arquitetura proposta. Em cenários contendo várias fontes de dados externas conectadas ao OntoGenesis, seria possível obter melhores resultados realizando o *match* apenas com as fontes externas pertencentes ao mesmo domínio do serviço de dados.

REFERÊNCIAS

ABELE, A. et al. *Linking Open Data Cloud Diagram*. 2017. Disponível em: <http://lod-cloud.net/>. Acesso em: 20 dez. 2017.

AHMAR, M. V. T.; ALMEIDA, G. A. A. Brazil: The Transparency Portal of the Federal Government. In: FALK, S.; RÖMMELE, A.; SILVERMAN, M. (Ed.). *Digital Government: Leveraging Innovation to Improve Public Sector Performance and Outcomes for Citizens*. [S.l.]: Springer International Publishing, 2017. cap. Part II, p. 129–147. ISBN 978-3-319-38793-2.

AHMED, R.; BOUTABA, R. A Survey of Distributed Search Techniques in Large Scale Distributed Systems. *IEEE Communications Surveys & Tutorials*, v. 13, n. 2, p. 150–167, 2011. ISSN 1553-877X.

AL-SHARGABI, B.; SHEIKH, A. E.; SABRI, A. Web Service composition survey: State of the art review. *Recent Patents on Computer Science*, v. 3, n. 2, p. 91–107, 2010.

ALFARIES, A. *Ontology Learning for Semantic Web Services*. Tese (Doutorado) — Brunel University, UK, 2010. Disponível em: <http://dspace.brunel.ac.uk/handle/2438/4667>. Acesso em: 25 jan. 2017.

ALFARIES, A.; BELL, D.; LYCETT, M. Ontology Learning for Semantic Web Services. In: *Proceedings of the 14th Annual UK Association of Information Systems Conference (UKAIS)*. Oxford University, Oxford, U.K: [s.n.], 2009. p. 27–36.

ALMEIDA, M.; BAX, M. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. *Ciência da Informação*, v. 32, n. 3, 2004. ISSN 1518-8353.

ALONSO, G. et al. *Web Services: Concepts, Architectures and Applications*. [S.l.]: Springer-Verlag Berlin Heidelberg, 2004. ISBN 3-540-44008-9.

BAADER, F. *The description logic handbook: Theory, implementation and applications*. 2nd. ed. [S.l.]: Cambridge university press, 2010. ISBN 9780521150118.

BACHLECHNER, D.; FINK, K. Semantic Web Service research: Current challenges and proximate achievements. *International Journal of Computer Science and Applications*, IJCSA, v. 5, n. 3, p. 117–140, 2008.

BARROS, H. et al. Steps, Techniques, and Technologies for the Development of Intelligent Applications Based on Semantic Web Services: A Case Study in e-Learning Systems. *Engineering Applications of Artificial Intelligence*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 24, n. 8, p. 1355–1367, 2011. ISSN 0952-1976.

BARTALOS, P.; BIELIKOVA, M. Automatic Dynamic Web Service Composition: A Survey and Problem Formalization. *Computing and Informatics*, v. 30, n. 4, p. 793–827, 2011.

BERNERS-LEE, T. *Linked-data design issues*. 2006. W3C Design Issue Document. Disponível em: <http://www.w3.org/DesignIssues/LinkedData.html>. Acesso em: 27 abr. 2017.

BERNERS-LEE, T.; FIELDING, R. T.; MASINTER, L. M. *Uniform Resource Identifier (URI): Generic Syntax*. RFC Editor, 2005. RFC 3986. (Request for Comments, 3986). Disponível em: <https://rfc-editor.org/rfc/rfc3986.txt>.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. *Scientific American*, v. 284, n. 5, p. 34–43, maio 2001.

BIANCHINI, D.; ANTONELLIS, V. D.; MELCHIORI, M. Developers' Networks Contribution to Web Application Design. In: *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*. New York, NY, USA: ACM, 2015. (iiWAS '15), p. 55:1–55:10. ISBN 978-1-4503-3491-4.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked Data - The Story So Far. *International journal on Semantic Web and Information Systems*, IJSWIS, v. 5, n. 3, p. 1–22, 2009. ISSN 15526283.

BIZER, C. et al. Linked Data on the Web (LDOW2008). In: *Proceedings of the 17th International Conference on World Wide Web*. New York, NY, USA: ACM, 2008. (WWW '08), p. 1265–1266. ISBN 978-1-60558-085-2.

BLAKE, M. B. et al. *Semantic Web Services: Advancement through Evaluation*. [S.l.]: Springer, 2012. ISBN 978-3-642-28734-3.

BORST, W. N. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. Tese (Doutorado) — University of Twente, Enschede, Netherlands, 1997.

BOUGUETTAYA, A. et al. A Service Computing Manifesto: The Next 10 Years. *Communications of the ACM*, ACM, New York, USA, v. 60, n. 4, p. 64–72, mar 2017. ISSN 0001-0782.

BRAVO, M.; RODRÍGUEZ, J.; PASCUAL, J. SDWS: Semantic description of web services. *International Journal of Web Services Research*, v. 11, n. 2, p. 1–23, 2014. ISSN 1545-7362.

BRUIJN, J. d. et al. Using the Web Service Modeling Ontology to Enable Semantic e-Business. *Commun. ACM*, ACM, New York, NY, USA, v. 48, n. 12, 2005. ISSN 0001-0782.

BUITELAAR, P.; CIMIANO, P. *Ontology Learning and Population: Bridging the Gap Between Text and Knowledge - Volume 167 Frontiers in Artificial Intelligence and Applications*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2008.

CAMPOS, G. M. M.; ROSA, N. S.; PIRES, L. F. A Survey of Formalization Approaches to Service Composition. In: *Services Computing (SCC), 2014 IEEE International Conference on*. [S.l.: s.n.], 2014. p. 179–186.

CAO, H.; FALLERI, J.-R.; BLANC, X. Automated Generation of REST API Specification from Plain HTML Documentation. In: MAXIMILIEN, M. et al. (Ed.). *Service-Oriented Computing*. Cham: Springer International Publishing, 2017. p. 453–461. ISBN 978-3-319-69035-3.

CIMIANO, P.; VÖLKER, J. Text2Onto: A Framework for Ontology Learning and Data-driven Change Discovery. In: *Proc. of the 10th Internat. Conf. on Natural Language Processing and Information Systems*. Berlin, Heidelberg: Springer-Verlag, 2005. (NLDB'05), p. 227–238. ISBN 3-540-26031-5, 978-3-540-26031-8.

CREMASCHI, M.; PAOLI, F. D. Toward Automatic Semantic API Descriptions to Support Services Composition. In: PAOLI, F. D.; SCHULTE, S.; JOHNSEN, E. B. (Ed.). *Service-Oriented and Cloud Computing*. [S.l.]: Springer International Publishing, 2017. p. 159–167.

DAVID, J. et al. The Alignment API 4.0. *Semantic Web*, IOS Press, v. 2, n. 1, p. 3–10, jan. 2011. ISSN 1570-0844.

DAVID, J.; GUILLET, F.; BRIAND, H. Association Rule Ontology Matching Approach. *International Journal on Semantic Web and Information Systems*, IGI Global, v. 3, n. 2, p. 27–49, 2007. ISSN 1552-6283.

DING, L.; PERISTERAS, V.; HAUSENBLAS, M. Linked Open Government Data. *IEEE Intelligent Systems*, v. 27, n. 3, p. 11–15, 2012. ISSN 1541-1672.

DONG, H.; HUSSAIN, F. K.; CHANG, E. Semantic Web Service matchmakers: state of the art and challenges. *Concurrency and Computation: Practice and Experience*, v. 25, n. 7, p. 961–988, may 2013. ISSN 15320626.

DUAN, Q.; YAN, Y.; VASILAKOS, A. V. A survey on service-oriented network virtualization toward convergence of networking and cloud computing. *IEEE Transactions on Network and Service Management*, v. 9, n. 4, p. 373–392, 2012.

DUO, Z.; JUAN-ZI, L.; BIN, X. Web Service Annotation Using Ontology Mapping. In: *Proceedings of the IEEE International Workshop*. Washington, DC, USA: IEEE Computer Society, 2005. (SOSE '05), p. 243–250. ISBN 0-7695-2438-9.

ELSAYED, D. H.; SALAH, A. Semantic Web Service discovery: A systematic survey. In: *2015 11th International Computer Engineering Conference: Today Information Society What's Next?, ICENCO 2015*. [S.l.: s.n.], 2015. p. 131–136.

EUZENAT, J.; EHRIG, M.; CASTRO, R. G. *Towards a methodology for evaluating alignment and matching algorithms.* Ontology Alignment Evaluation Initiative (OAEI), 2005.

EUZENAT, J.; SHVAIKO, P. *Ontology Matching*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. ISBN 3540496114.

FALBO, R. A. *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*. Tese (Doctoral dissertation) — COPPE/UFRJ, Rio de Janeiro, 1998.

FALBO, R. d. A. SABiO: Systematic Approach for Building Ontologies. In: *Proc. of the 1st Joint Workshop ONTO.COM /*

ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering, co-located with 8th International Conference on Formal Ontology in Information Systems (FOIS 2014). [S.l.]: CEUR-WS, 2014. p. 16–29.

FELLAH, A.; MALKI, M.; ELÇI, A. Web services matchmaking based on a partial ontology alignment. *International Journal of Information Technology and Computer Science (IJITCS)*, MECS Publisher, v. 8, n. 6, p. 9–20, 2016. ISSN 2074-9007.

FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ, A.; JURISTO, N. Methontology: from ontological art towards ontological engineering. In: *Proc. Symposium on Ontological Engineering of AAAI*. [S.l.: s.n.], 1997.

FIELDING, R. T. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doctoral dissertation) — University of California, Irvine, 2000.

FREIRE, F.; FREIRE, C.; SOUZA, D. Enhancing JSON to RDF data conversion with entity type recognition. In: *Proceedings of the 13th International Conference on Web Information Systems and Technologies, WEBIST, Porto, Portugal*. [S.l.: s.n.], 2017. p. 97–106.

GAO, L.; URBAN, S. D.; RAMACHANDRAN, J. A survey of transactional issues for Web Service composition and recovery. *International Journal of Web and Grid Services*, v. 7, n. 4, p. 331–356, 2011.

GARRIGA, M. et al. RESTful service composition at a glance: A survey. *Journal of Network and Computer Applications*, v. 60, p. 32–53, 2016.

GIROLAMI, M.; CHESSA, S.; CARUSO, A. On service discovery in mobile social networks: Survey and perspectives. *Computer Networks*, v. 88, p. 51–71, 2015.

GÓMEZ-PÉREZ, A.; FERNÁNDEZ-LÓPEZ, M.; CORCHO, O. *Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. (Advanced Information and Knowledge Processing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. ISBN 1846283965.

GREGORIO, J. et al. *URI Template*. 2012. RFC 6570. Disponível em: <https://tools.ietf.org/html/rfc6570>. Acesso em: 02 out. 2017.

GROLINGER, K. et al. Integration of business process modeling and Web services: A survey. *Service Oriented Computing and Applications*, v. 8, n. 2, p. 105–128, 2014.

GRUBER, T. R. A translation approach to portable ontology specifications. *Knowledge Acquisition*, Academic Press Ltd., London, UK, v. 5, n. 2, p. 199–220, jun. 1993. ISSN 1042-8143.

GRÜNINGER, M.; FOX, M. S. *Methodology for the Design and Evaluation of Ontologies*. University of Toronto, Canada, 1995.

GUARINO, N. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In: _____. *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology: International Summer School, SCIE-97 Frascati, Italy, July*. [S.l.]: Springer Berlin Heidelberg, 1997. p. 139–170. ISBN 978-3-540-69548-6.

GUARINO, N. *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*. 1st. ed. Amsterdam, The Netherlands, The Netherlands: IOS Press, 1998. ISBN 9051993994.

GUIZZARDI, G. Summary for Policymakers. In: Intergovernmental Panel on Climate Change (Ed.). *Climate Change 2013 - The Physical Science Basis*. [S.l.]: Cambridge University Press, 2007. v. 155, p. 1–30. ISBN 978-1-58603-715-4.

HALL, W. et al. Linked open government data: Lessons from data.gov.uk. *IEEE Intelligent Systems*, IEEE Computer Society, Los Alamitos, CA, USA, v. 27, p. 16–24, 2012. ISSN 1541-1672.

HANG, F.; ZHAO, L. Supporting End-User Service Composition: A Systematic Review of Current Activities and Tools. In: *Proceedings of the IEEE International Conference on Web Services, ICWS 2015*. [S.l.: s.n.], 2015. p. 479–486.

HEATH, T.; BIZER, C. *Linked Data: Evolving the Web into a Global Data Space*. 1st. ed. Morgan & Claypool, 2011. ISBN 9781608454303. Disponível em: <http://linkeddatabook.com/>.

HOHMANN, L. *Beyond Software Architecture: Creating and Sustaining Winning Solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0201775948.

HOPCROFT, J. E.; ULLMAN, J. D. *Introduction To Automata Theory, Languages, And Computation*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990. ISBN 020102988X.

IMMONEN, A.; PAKKALA, D. A survey of methods and approaches for reliable dynamic service compositions. *Service Oriented Computing and Applications*, v. 8, n. 2, p. 129–158, jun 2014. ISSN 1863-2386.

ISOTANI, S.; BITTENCOURT, I. I. *Dados Abertos Conectados*. São Paulo: Novatec, 2015. ISBN 978-85-7522-449-6.

ISSARNY, V. et al. Service-oriented middleware for the Future Internet: State of the art and research directions. *Journal of Internet Services and Applications*, v. 2, n. 1, p. 23–45, 2011.

JULA, A.; SUNDARARAJAN, E.; OTHMAN, Z. Cloud computing service composition: A systematic literature review. *Expert Systems with Applications*, v. 41, n. 8, p. 3809–3824, jun 2014.

KASHYAP, V.; BUSSLER, C.; MORAN, M. *The Semantic Web, Semantics for Data and Services on the Web*. Berlin, Heidelberg: Springer-Verlag, 2008. ISBN 978-3-540-76451-9.

KITCHENHAM, B. *Procedures for Performing Systematic Reviews*. Department of Computer Science, Keele University, UK, 2004.

KOPECKÝ, J.; GOMADAM, K.; VITVAR, T. hRESTS: An HTML Microformat for Describing RESTful Web Services. In: *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT'08*. Washington, USA: IEEE, 2008. p. 619–625.

KOPECKÝ, J. et al. SAWSDL: Semantic Annotations for WSDL and XML Schema. *IEEE Internet Comput.*, v. 11, n. 6, p. 60–67, 2007. ISSN 1089-7801.

KÜSTER, U.; LAUSEN, H.; KÖNIG-RIES, B. Evaluation of semantic service discovery—a survey and directions for future research. In: *Emerging Web Services Technology, Volume II*. [S.l.]: Birkhäuser Basel, 2008. p. 41–58.

LANTHALER, M. Creating 3rd generation web apis with hydra. In: *Proceedings of the 22Nd International Conference on World Wide Web*. New York, NY, USA: ACM, 2013. (WWW '13 Companion), p. 35–38. ISBN 978-1-4503-2038-2.

LANTHALER, M.; GÜTL, C. Hydra: A vocabulary for hypermedia-driven web apis. In: BIZER, C. et al. (Ed.). *Proceedings of the 6th Workshop on Linked Data on the Web (LDOW2013) at the 22nd International World Wide Web Conference (WWW2013)*. [S.l.]: CEUR-WS.org, 2013. (CEUR Workshop Proceedings, v. 996).

LANTHALER, M.; SPORNY, M.; KELLOGG, G. *JSON-LD 1.0*. [S.l.], 2014. Http://www.w3.org/TR/2014/REC-json-ld-20140116/.

LEE, C.-S. et al. Automated ontology construction for unstructured text documents. *Data and Knowledge Engineering*, v. 60, n. 3, p. 547 – 566, 2007.

LEITE, L. A. F. et al. A systematic literature review of service choreography adaptation. *Service Oriented Computing and Applications*, v. 7, n. 3, p. 199–216, 2013.

LEMOS, A. L.; DANIEL, F.; BENATALLAH, B. Web service composition: A survey of techniques and tools. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 48, n. 3, p. 33:1–33:41, dez. 2015. ISSN 0360-0300.

LIEPINS, R.; GRASMANIS, M.; BOJARS, U. Owlgred ontology visualizer. In: *Proceedings of the 2014 International Conference on Developers - Volume 1268*. Aachen, Germany, Germany: CEUR-WS.org, 2014. (ISWC-DEV'14), p. 37–42.

LIN, S.-Y. et al. Automated knowledge discovery and semantic annotation for network and web services. *International Journal of Distributed Sensor Networks*, v. 12, n. 7, p. 1–11, 2016.

LIRA, C.; CAETANO, P. REST-Based Semantic Annotation of Web Services. In: *Information Technology: New Generations*. [S.l.]: Springer, 2016. v. 448, p. 269–279. ISBN 978-3-319-32466-1. ISSN 21945357.

LIRA, H. A. et al. Semantic Data Services: An approach to access and manipulate Linked Data. In: *XL Latin American Computing Conference (CLEI)*. [S.l.]: IEEE, 2014. p. 1–12. ISBN 978-1-4799-6130-6.

LUO, C. C. et al. Automated structural semantic annotation for RESTful services. *International Journal of Web and Grid Services*, v. 12, n. 1, p. 26–41, 2016. ISSN 1741-1106.

MAEDCHE, A.; STAAB, S. Ontology learning for the semantic web. *IEEE Intelligent Systems*, IEEE Educational Activities Department, Piscataway, USA, v. 16, n. 2, mar. 2001. ISSN 1541-1672.

MALESHKOVA, M.; PEDRINACI, C.; DOMINGUE, J. Supporting the Creation of Semantic RESTful Service Descriptions. In: *8th International Semantic Web Conference (ISWC'09)*. Washington D.C., USA: [s.n.], 2009.

MÁRMOL, F. G.; KUHNEN, M. Q. Reputation-based Web service orchestration in cloud computing: A survey. *Concurrency Computation*, v. 27, n. 9, p. 2390–2412, 2015.

MARTIN, D. et al. *OWL-S: Semantic Markup for Web Services*. [S.l.], 2007. Http://www.daml.org/services/owl-s/1.2/overview/.

MCGUINNESS, D. L. et al. Daml+oil: An ontology language for the semantic web. *IEEE Intelligent Systems*, v. 17, n. 5, p. 72–80, 2002. ISSN 1541-1672.

MCILRAITH, S.; SON, T.; ZENG, H. Z. H. Semantic Web services. *IEEE Intelligent Systems*, v. 16, n. 2, p. 46–53, mar. 2001. ISSN 1541-1672.

MOHR, F.; WALTHER, S. Template-based generation of semantic services. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 8919, p. 188–203, 2014.

MURGUZUR, A. et al. Process flexibility in service orchestration: A systematic literature review. *International Journal of Cooperative Information Systems*, v. 23, n. 3, 2014.

NACER, H.; AISSANI, D. Review: Semantic Web Services: Standards, Applications, Challenges and Solutions. *Journal*

of Network and Computer Applications, Academic Press Ltd., London, UK, v. 44, p. 134–151, 2014. ISSN 1084-8045.

NAZMUDEEN, M. S.; BUHARI, S. M. A survey on distributed service discovery mechanisms with the focus on topology awareness. *Advances in Intelligent Systems and Computing*, v. 331, p. 315–326, 2015.

NGAN, L. D.; KANAGASABAI, R. Semantic Web service discovery: State-of-the-art and research challenges. *Personal and Ubiquitous Computing*, v. 17, n. 8, p. 1741–1752, 2013.

NGUYEN, T. T. S.; LU, H. Domain ontology construction using web usage data. In: *Advances in Artificial Intelligence*. [S.l.]: Springer, 2016. p. 338–344.

NUNES, B. P. et al. Complex Matching of RDF Datatype Properties. In: DECKER, H. et al. (Ed.). *Database and Expert Systems Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 195–208. ISBN 978-3-642-40285-2.

NUZZOLESE, A. G. et al. Conference Linked Data: the ScholarlyData Project. In: *Proceedings of ISWC 2016 - Resource Track*. [S.l.]: Springer, 2016. (Lecture Notes in Computer Science, v. 9982), p. 150–158.

OLIVEIRA, B. C. N. et al. Automatic semantic enrichment of data services. In: *iiWAS '17: The 19th International Conference on Information Integration and Web-based Applications & Services, December 4–6, 2017, Salzburg, Austria.* New York, USA: ACM, 2017. p. 415–424.

OLIVEIRA, B. C. N. et al. A Platform to Enrich, Expand and Publish Linked Data of Police Reports. In: *Proceedings of the IADIS International Conference on WWW/Internet (ICWI)*. Manheim, Germany: [s.n.], 2016. p. 111–118. ISBN 978-989-8533-57-9.

OLIVEIRA, B. C. N.; SIQUEIRA, F. OntoGenesis: Uma Arquitetura para Enriquecimento Semântico de Web Data Services. In: *Anais do XXIII Simpósio Brasileiro de Sistemas Multimídia e Web: XVII Workshop de Teses de Dissertações (WTD 2017)*. Gramado, Brazil: [s.n.], 2017. p. 29–34. ISBN 978-85-7669-380-2.

OPEN DEFINITION. *Open Definition*. 2014. Disponível em: <http://opendefinition.org/>. Acesso em: 27 abr. 2017.

ORDÓNEZ, A. et al. Towards automated composition of convergent services: A survey. *Computer Communications*, v. 69, p. 1–21, 2015.

PAPAZOGLOU, M. P. Service-oriented computing: concepts, characteristics and directions. In: *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.* [S.l.: s.n.], 2003. p. 3–12.

PAVEL, S.; EUZENAT, J. Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 25, n. 1, p. 158–176, 2013. ISSN 1041-4347.

PERRY, D. E.; WOLF, A. L. Foundations for the Study of Software Architecture. *SIGSOFT Softw. Eng. Notes*, ACM, New York, USA, v. 17, n. 4, p. 40–52, october 1992. ISSN 0163-5948.

PLATENIUS, M. C. et al. A survey of fuzzy service matching approaches in the context of on-the-fly computing. In: *CBSE 2013 - Proceedings of the 16th ACM SIGSOFT Symposium on Component Based Software Engineering.* [S.l.: s.n.], 2013. p. 143–152.

RATHINAM, N. E.; PARTHIBAN, L.; MARIAKALAVATHY, G. Automatic discovery of relevant web services with semantic ranking. *Research Journal of Applied Sciences, Engineering and Technology*, v. 8, n. 22, p. 2240–2247, 2014. ISSN 20407467.

ROCHA, R. dos S.; DEGROSSI, L. C.; ALBUQUERQUE, J. P. de. A systematic literature review of geospatial web service composition. In: *CIBSE 2015 - XVIII Ibero-American Conference on Software Engineering.* [S.l.: s.n.], 2015. p. 502–515.

ROMAN, D. et al. Web service modeling ontology. *Applied Ontology*, IOS Press, v. 1, n. 1, p. 77–106, 2005. ISSN 1570-5838.

SALEM, S.; ABDELRAHMAN, S. A multiple-domain ontology builder. In: *Proc. of the 23rd Internat. Conf. on Computational Linguistics.* Stroudsburg, USA: Association for Computational Linguistics, 2010. p. 967–975.

SALVADORI, I. et al. Improving Entity Linking with Ontology Alignment for Semantic Microservices Composition. *International Journal of Web Information Systems*, v. 13, n. 3, p. 302–323, 2017. ISSN 1744-0084.

SALVADORI, I. et al. An ontology alignment framework for data-driven microservices. In: *iiWAS '17: The 19th International Conference on Information Integration and Web-based Applications & Services, December 4–6, 2017, Salzburg, Austria.* New York, USA: ACM, 2017. p. 425–433.

SAQUICELA, V.; VILCHES-BLAZQUEZ, L. M.; CORCHO, O. Lightweight Semantic Annotation of Geospatial RESTful Services. In: *Proceedings of the 8th Extended Semantic Web Conference on The Semanic Web: Research and Applications - Volume Part II.* Berlin, Heidelberg: Springer-Verlag, 2011. (ESWC'11), p. 330–344. ISBN 978-3-642-21063-1.

SCHULZ, K. U.; MIHOV, S. Fast string correction with Levenshtein automata. *International Journal on Document Analysis and Recognition*, v. 5, n. 1, p. 67–85, 2002. ISSN 1433-2833.

SCHWICHTENBERG, S.; GERTH, C.; ENGELS, G. From open api to semantic specifications and code adapters. In: *2017 IEEE International Conference on Web Services (ICWS).* [S.l.: s.n.], 2017. p. 484–491.

SEGEV, A.; SHENG, Q. Z. Bootstrapping ontologies for web services. *IEEE Transactions on Services Computing*, IEEE Computer Society, Washington, DC, USA, v. 5, n. 1, p. 33–44, jan. 2012. ISSN 1939-1374.

SPORNY, M. et al. *JSON-LD 1.0: A JSON-based Serialization for Linked Data.* 2014. Disponível em: <https://www.w3.org/TR/json-ld/>. Acesso em: 28 abr. 2017.

STRUNK, A. QoS-aware service composition: A survey. In: *Proceedings - 8th IEEE European Conference on Web Services, ECOWS 2010.* [S.l.: s.n.], 2010. p. 67–74.

SUBHASHINI, R.; AKILANDESWARI, J. A survey on ontology construction methodologies. *International Journal of Enterprise Computing and Business Systems*, v. 1, n. 1, p. 60–72, 2011.

SUCHANEK, F. M.; ABITEBOUL, S.; SENELLART, P. Paris: Probabilistic alignment of relations, instances, and schema. *Procedures of the VLDB Endowment*, VLDB Endowment, v. 5, n. 3, p. 157–168, 2011.

SUN, L.; DONG, H.; ASHRAF, J. Survey of service description languages and their issues in cloud computing. In: *Proceedings - 2012 8th International Conference on Semantics, Knowledge and Grids, SKG 2012*. [S.l.: s.n.], 2012. p. 128–135.

SUN, L. et al. Cloud service selection: State-of-the-art and future research directions. *Journal of Network and Computer Applications*, v. 45, p. 134–150, 2014.

SYU, Y.; FANJIANG, Y.-Y. A Survey to Service Composition Methods Using Aspects Classification. In: *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. [S.l.: s.n.], 2013. p. 170–181.

TALANTIKITE, H. N.; AISSANI, D.; BOUDJLIDA, N. Semantic annotations for web services discovery and composition. *Comput. Stand. Interfaces*, Elsevier Science Publishers B. V., v. 31, n. 6, p. 1108–1117, 2009. ISSN 0920-5489.

TEKA, A. Y.; FERNANDEZ, N. C.; SAPKOTA, B. A systematic literature review on service description methods. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 7195 LNCS, p. 239–255, 2012.

TOSI, D.; MORASCA, S. Supporting the semi-automatic semantic annotation of web services: A systematic literature review. *Information and Software Technology*, v. 61, p. 16–32, may 2015. ISSN 09505849.

TRAN, Q.-V.; ICHISE, R.; HO, B.-Q. Cluster-based similarity aggregation for ontology matching. In: *Proceedings of the 6th International Conference on Ontology Matching - Volume 814*. Aachen, Germany: CEUR-WS.org, 2011. (OM'11), p. 142–147.

USCHOLD, M. Building ontologies: Towards a unified methodology. In: *In 16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*. [S.l.: s.n.], 1996.

USCHOLD, M.; JASPER, R. A Framework for Understanding and Classifying Ontology Applications. In: *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*. Stockholm, Sweden: [s.n.], 1999.

VORUGANTI, S. Survey on Data-intensive Applications, Tools and Techniques for Mining Unstructured Data. *International Journal of Computer Applications*, Foundation of Computer Science (FCS), NY, USA, New York, USA, v. 146, n. 12, p. 23–27, Jul 2016. ISSN 0975-8887.

W3C. *OWL Web Ontology Language Reference*. W3C Recommendation, 2004. Disponível em: <https://www.w3.org/TR/owl-ref/>. Acesso em: 08 nov. 2017.

W3C. *Web Services Architecture*. W3C Working Group, 2004. Disponível em: <https://www.w3.org/TR/ws-arch/>. Acesso em: 08 nov. 2017.

W3C. *RDF Schema 1.1*. W3C Recommendation, 2014. Disponível em: <https://www.w3.org/TR/2014/REC-rdf-schema-20140225/>. Acesso em: 27 abr. 2017.

W3C. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation, 2014. Disponível em: <https://www.w3.org/TR/rdf11-concepts/>. Acesso em: 27 abr. 2017.

W3C. *Semantic Web - Vocabularies*. W3C Standards, 2014. Disponível em: <https://www.w3.org/standards/semanticweb/ontology>. Acesso em: 08 nov. 2017.

WANG, Y.; WANG, Y. A survey of change management in service-based environments. *Service Oriented Computing and Applications*, v. 7, n. 4, p. 259–273, 2013.

WU, X.; CHEN, C.; HUANG, H. A survey on web service composition: From service description, automatic process generation to process evaluation. *International Journal of Digital Content Technology and its Applications*, v. 6, n. 17, p. 483–495, 2012.

YAO, Y. et al. An automatic semantic extraction method for web data interchange. *2014 6th International Conference on Computer Science and Information Technology (CSIT)*, p. 148–152, 2014.

ZAPILKO, B.; MATHIAK, B. Object Property Matching Utilizing the Overlap between Imported Ontologies. In: PRESUTTI, V. et al. (Ed.). *The Semantic Web: Trends and Challenges*. Cham: Springer International Publishing, 2014. p. 737–751. ISBN 978-3-319-07443-6.

ZHANG, Z.; CHEN, S.; FENG, Z. Semantic Annotation for Web Services Based on DBpedia. In: *Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. Washington, DC, USA: IEEE Computer Society, 2013. (SOSE '13), p. 280–285.

ZILCI, B. I.; SLAWIK, M.; KUPPER, A. Cloud Service Matchmaking Approaches: A Systematic Literature Survey. In: *Proceedings of the 2015 26th International Workshop on Database and Expert Systems Applications*. Washington, DC, USA: IEEE Computer Society, 2015. (DEXA '15), p. 181–185. ISBN 978-1-4673-7582-5.

APÊNDICE A – PUBLICAÇÕES

No decorrer do desenvolvimento desta pesquisa foram produzidos alguns trabalhos no LaPeSD (Laboratório de Pesquisa em Sistemas Distribuídos), os quais foram publicados em veículos científicos. Seguem abaixo as publicações resultantes desta pesquisa:

1. Uma plataforma para publicação e enriquecimento de *Linked Data* permitiu realizar análises estatísticas interessantes sobre boletins de ocorrência e processos criminais. Tal pesquisa germinou a proposta desta dissertação e resultou na seguinte publicação:

    - OLIVEIRA, B. C. N. et al. A Platform to Enrich, Expand and Publish Linked Data of Police Reports. In: *Proceedings of the IADIS International Conference on WWW/Internet (ICWI)*. Manheim, Germany: [s.n.], 2016. p. 111–118. ISBN 978-989-8533-57-9.

2. Uma visão geral da proposta desta pesquisa, bem como os resultados parciais, foi apresentado no Workshop de Teses e Dissertações do WebMedia'17:

    - OLIVEIRA, B. C. N.; SIQUEIRA, F. OntoGenesis: Uma Arquitetura para Enriquecimento Semântico de Web Data Services. In: *Anais do XXIII Simpósio Brasileiro de Sistemas Multimídia e Web: XVII Workshop de Teses de Dissertações (WTD 2017)*. Gramado, Brazil: [s.n.], 2017. p. 29–34. ISBN 978-85-7669-380-2.

3. A abordagem para enriquecimento semântico automático das representações fornecidas por serviços de dados resultou na seguinte publicação:

    - OLIVEIRA, B. C. N. et al. Automatic semantic enrichment of data services. In: *iiWAS '17: The 19th International Conference on Information Integration and Web-based Applications & Services, December 4–6, 2017, Salzburg, Austria.* New York, USA: ACM, 2017. p. 415–424.

4. Um artigo incluindo uma discussão mais completa e detalhada do OntoGenesis com resultados experimentais adicionais, foi aceito para publicação em um periódico internacional:

    - Título: OntoGenesis: An Architecture for Automatic Semantic Enhancement of Data Services.
    - Veículo: IJWIS – International Journal of Web Information Systems (Qualis B1).
    - Autores: Bruno C. N. Oliveira, Alexis Huf, Ivan Salvadori e Frank Siqueira.
    - Aceito em maio de 2018.

Adicionalmente, outras publicações científicas também foram alcançadas em colaboração com outros integrantes do LaPeSD e se relacionam indiretamente com esta dissertação. Tais publicações permeiam as áreas de Alinhamento de Ontologias, Web Semântica e Composição de Serviços:

1. SALVADORI, I. et al. An ontology alignment framework for data-driven microservices. In: *iiWAS '17: The 19th International Conference on Information Integration and Web-based Applications & Services, December 4–6, 2017, Salzburg, Austria.* New York, USA: ACM, 2017. p. 425–433.

2. SALVADORI, I. et al. Improving Entity Linking with Ontology Alignment for Semantic Microservices Composition. *International Journal of Web Information Systems*, v. 13, n. 3, p. 302–323, 2017. ISSN 1744-0084.

APÊNDICE B – META RSL

Este apêndice fornece detalhes adicionais da meta revisão sistemática que auxiliou na análise do estado da arte. A RSL realizada é chamada de meta revisão pelo fato de ter sido feito um estudo de outras revisões já existentes na área. Diferentemente de uma RSL tradicional, a qual considera estudos primários, uma meta revisão é utilizada para analisar os trabalhos secundários. Em particular, ela é apropriada para verificar se a base de evidência atual é completa (ou incompleta), sintetizar revisões sistemáticas anteriores, identificar lacunas nestas revisões e fornecer direções para novas pesquisas. Este tipo de pesquisa geralmente contempla mais trabalhos e fornece mais evidências que estudos empíricos separados. Portanto, em virtude da grande quantidade de revisões existentes na área de composição de serviços Web, preferiu-se estudar estas revisões, o que permitiu também identificar os trabalhos primários mais relevantes citados pelos artigos.

Recorrendo à metodologia descrita por Kitchenham (2004), definir um protocolo de pesquisa é imprescindível para a validação e reprodução da RSL por outros pesquisadores. É preciso, portanto, conceber questões de pesquisa em consonância com uma estratégia de busca e que sejam adequadas aos objetivos e expectativas dos pesquisadores. É necessário também estabelecer em cada etapa da revisão sistemática critérios de inclusão e exclusão como crivo dos trabalhos obtidos na busca dos artigos.

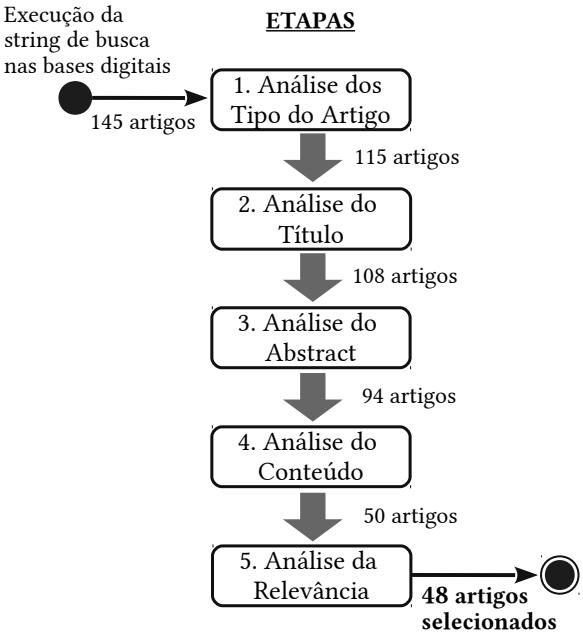
A meta revisão foi executada por 3 membros do LaPeSD (Laboratório de Pesquisa em Sistemas Distribuídos), no processo de revisão em pares seguindo a metodologia descrita por Kitchenham (2004). O escopo da RSL foi amplo e, como primeiro passo, foram definidas 11 questões de pesquisa que envolvessem os processos de composição, descoberta, seleção e descrição de serviços Web. Dentre algumas questões, pode-se citar: “qual a relação destes processos com dados conectados, de acordo com as revisões?”; “as revisões diferenciam serviços baseados em dados e baseados em ações?”; “como os trabalhos primários são classificados pelas revisões?”; “as revisões diferenciam tipos de serviços ou abordam algum tipo específico? Quais?”; “como a semântica é considerada pelas revisões?”, dentre outras. A motivação desta última pergunta apresentada é identificar como a Web Semântica é utilizada no contexto de *Web Services* e quais características são consideradas para diferenciar serviços semânticos de serviços não semânticos (tradicionais). O resultado das análises extraídas desta questão pôde ser incorporado a este traba-

lho e estimulou o desenvolvimento da proposta desta dissertação.

Após definidas as questões de pesquisa, foram consultadas revisões/*surveys* publicados entre 01 de janeiro de 2009 à 31 de dezembro de 2016 nas bases de dados científicas do *Scopus*, IEEE Xplore e ACM Digital Library. A seguinte *string* de busca foi utilizada nas bases de dados, considerando os termos em títulos, palavras-chave e *abstract*:

```
(("semantic service")   OR ("semantic web service") OR
("service composition") OR ("service description")  OR
("service discovery")   OR ("service selection"))  AND
(("survey") OR ("review") OR ("state of the art")   OR
("systematic mapping"))
```

Figura 28 – Etapas da meta revisão sistemática.



Fonte: Elaborado pelo autor (2018).

A Figura 28 ilustra as etapas realizadas na RSL, considerando os resultados da *string* de busca executada nas bases de dados. Vale ressaltar que em cada etapa os trabalhos passaram por uma análise

por pares, onde pelos menos dois pesquisadores analisaram o mesmo artigo. Em casos de divergência, o terceiro pesquisador participava da análise do artigo e dava o seu parecer para inclusão ou exclusão do artigo. Inicialmente, foram coletados 145 artigos, incluindo *surveys*, revisões sistemáticas da literatura, mapeamentos sistemáticos, dentre outras revisões. Na primeira etapa de análise dos artigos coletados, estabeleceu-se os seguintes critérios de exclusão:

- artigos curtos (*short papers*);
- trabalhos estendidos;
- capítulos de livros; e
- artigos em duplicidade.

Ao final da primeira etapa foram selecionados 115 artigos, dos quais foi realizada a análise dos títulos considerando alguns critérios de exclusão:

- não contém o termo “review”;
- não contém o termo “survey”;
- não contém o termo “state of the art”; e
- não contém o termo “systematic mapping”.

Dentre os 115 títulos analisados, 7 atenderam os critérios de exclusão e, como consequência, os seus respectivos artigos foram descartados. Com isso, a etapa 3 buscou analisar o *abstract* dos demais artigos levando em consideração critérios de inclusão e exclusão. Como critério de exclusão foi analisado se, pelo *abstract*, o artigo apresentava uma contribuição original além da revisão. A razão pela qual foi estabelecido esse critério de exclusão é que contribuições originais estavam tendo um peso maior no artigo do que a revisão em si. Como critérios de inclusão, foram considerados se, pelo *abstract*, a revisão tinha escopo em:

- descrição de serviços;
- descoberta de serviços;
- seleção de serviços;
- composição de serviços; ou

- conceitos explicitamente relacionados aos anteriores.

Ao final da etapa 3 foram selecionados 94 artigos, dos quais foram submetidos, para a etapa seguinte, a critérios de inclusão e exclusão com relação aos seus conteúdos. Como critério de exclusão da etapa 4 foi adotado o mesmo da etapa anterior: se o artigo apresentava uma contribuição original além da revisão. O mesmo critério de exclusão foi utilizado, pois nem sempre é possível obter tal informação somente pelo *abstract*. Como critério de inclusão, foi analisado se o corpo do trabalho apresentava de maneira explícita direções de pesquisa ou lacunas (questões de pesquisa em aberto, conhecidas também como *research gaps*). É importante frisar que muitas revisões e *surveys* naturalmente apresentam limitações dos seus trabalhos primários, portanto, tais discussões não foram consideradas como direções de pesquisa.

Por fim, foram obtidos, a partir da análise de conteúdo, 50 artigos para prosseguirem à última etapa da meta revisão. Nessa etapa, foi analisada a relevância dos 50 trabalhos selecionados em consonância com as questões de pesquisa elicitadas na RSL. Dentre os 50 trabalhos obtidos na quarta etapa, 2 não respondiam a nenhuma pergunta. Como resultado, foram selecionadas 48 revisões e *surveys* em torno do tema de composição de serviços, em adição de uma análise dos principais trabalhos primários citados pelas revisões/*surveys* selecionados. Os trabalhos que respondem a questão de pesquisa que tange esta dissertação são apresentados na Tabela 5.

Ngan e Kanagasabai (2013) afirmam que para facilitar a descoberta de serviços é fundamental ter descrições semânticas dos serviços. Garriga et al. (2016) apontam que fornecer dados associados a conceitos semânticos utilizando, por exemplo, o JSON-LD como representação, auxilia na integração dos dados e propicia o desenvolvimento de *mashups*. O trabalho de Syu e Fanjiang (2013) revisa métodos de composição de serviços e conclui que *workflows* orientados a dados são importantes para combinar serviços de forma automática, visto que as conexões entre os dados são os principais responsáveis pela compatibilidade entre serviços.

Pode-se perceber a importância da adoção da Web Semântica em serviços Web para prover a interoperabilidade e automatizar processos envolvendo composição, descoberta e seleção de serviços. As revisões, todavia, não definem o conceito de serviços semânticos de maneira clara e contundente. De acordo com as análises realizadas, é possível observar características existentes nos serviços para permitir a automatização dos processos e a integração de dados e

aplicações Web. Algumas características são: a publicação de descrições semânticas, o fornecimento de representações contendo dados conectados a conceitos semânticos, a capacidade de *reasoning* por meio de um motor de inferências, e a capacidade de lidar com ambientes e ontologias heterogêneas.

Suspeita-se que mais características podem ser exploradas e um conceito mais amplo de serviços semânticos possa ser idealizado. Pesquisas futuras podem identificar as principais características consideradas para construir serviços Web semânticos e classificá-los de acordo com o nível de suporte semântico que oferecem. Outro desafio é ampliar a adoção de implementações reais voltadas a serviços Web semânticos. Tosi e Morasca (2015) argumentam que muitos pesquisadores têm focado em criar novas ontologias e ferramentas, mantendo a discussão no nível abstrato, o que, segundo Nacer e Aissani (2014), compromete ainda mais o alcance da interoperabilidade. Segundo os autores, o alinhamento de ontologias é o principal desafio para interoperabilidade e integração de dados. Para tanto, faz-se necessário conceber abordagens e serviços capazes de lidar com a heterogeneidade de ontologias existentes para o mesmo domínio, consubstanciando, assim, as técnicas de *matching* de ontologias em serviços Web semânticos.

Tabela 5 – Comparativo dos trabalhos da RSL que consideram o uso da Web Semântica.

Trabalho	**Uso da Semântica**	**Aborda Ontologia**
Garriga et al. (2016)	Descrição de serviços e das representações	Sim
Lemos, Daniel e Benatallah (2015)	Descrição de serviços	Sim
Zilci, Slawik e Kupper (2015)	Descrição de serviços	Sim
Tosi e Morasca (2015)	Descrição de serviços e das representações	Sim
Elsayed e Salah (2015)	Descrição de serviços	Sim
Hang e Zhao (2015)	Descrição de serviços	Sim
Girolami, Chessa e Caruso (2015)	Descrição de serviços	Não
Mármol e Kuhnen (2015)	Descrição de serviços	Sim
Rocha, Degrossi e Albuquerque (2015)	Descrição de serviços	Sim
Ordónez et al. (2015)	Descrição de serviços	Sim
Nazmudeen e Buhari (2015)	Modelagem de clusters e sistemas P2P	Sim
Jula, Sundararajan e Othman (2014)	Descrição de serviços	Sim
Nacer e Aissani (2014)	Descrição de serviços	Sim
Sun et al. (2014)	Descrição de serviços	Sim
Grolinger et al. (2014)	Modelagem de processos	Sim
Murguzur et al. (2014)	Descrição de serviços	Não
Campos, Rosa e Pires (2014)	Descrição de serviços	Sim
Immonen e Pakkala (2014)	Descrição de serviços	Sim
Platenius et al. (2013)	Descrição de serviços	Sim
Syu e Fanjiang (2013)	Descrição de serviços e das representações	Sim
Dong, Hussain e Chang (2013)	Descrição de serviços	Sim
Leite et al. (2013)	Descrição de serviços e das representações	Sim
Ngan e Kanagasabai (2013)	Descrição de serviços	Sim
Wang e Wang (2013)	Descrição de serviços	Sim
Duan, Yan e Vasilakos (2012)	Descrição de serviços	Sim
Sun, Dong e Ashraf (2012)	Descrição de serviços	Sim
Teka, Fernandez e Sapkota (2012)	Descrição de serviços	Sim
Wu, Chen e Huang (2012)	Descrição de serviços	Sim
Gao, Urban e Ramachandran (2011)	Descrição de serviços	Não
Issarny et al. (2011)	Descrição de serviços	Sim
Bartalos e Bielikova (2011)	Descrição de serviços	Sim
Ahmed e Boutaba (2011)	Descrição de serviços	Sim
Al-Shargabi, Sheikh e Sabri (2010)	Descrição de serviços	Sim
Strunk (2010)	Descrição de serviços	Não

APÊNDICE C – PARÂMETROS DOS *ENDPOINTS* (JAX-RS)

Há duas maneiras de se obter os parâmetros do URI de um serviço que adota o estilo arquitetural REST: por meio de *Path Parameters* e *Query Parameters* (BERNERS-LEE; FIELDING; MASINTER, 2005). Ambos os conceitos têm como objetivo fornecer um meio de consumidores passarem informações acerca do recurso solicitado. A diferença está na forma como as informações são passadas pelo URI. Os *Query Parameters* são adicionados no URI após o símbolo "?" (interrogação) e são utilizadas, em geral, para filtragem de recursos. Já os *Path Parameters* são associados aos segmentos do URI e são recomendados quando há necessidade de identificar um único recurso. A Figura 29 apresenta um trecho de código Java com suporte à especificação JAX-RS[1] que implementa *endpoints* de acesso e manipulação a dados de pessoas.

Para exemplificar um *Query Parameter* suponha o seguinte URI: `http://servico-exemplo/boletim/pessoa?rg=0123456789&profissao=professor%20titular`. Neste exemplo, há dois *Query Parameter*: um com o identificador `rg` e valor "0123456789", e outro identificado por `profissao` com o valor "professor titular"[2]. O método `getPessoaPorQuery` (linhas 8-12 da Figura 29) ilustra a implementação utilizando *Query Parameters* com JAX-RS. Para melhor exemplificar um *Path Parameter*, suponha o seguinte URI: `http://servico-exemplo/boletim/pessoa/0123456789/santa%20catarina`. Neste exemplo, os valores que anteriormente faziam parte de um *Query Parameter*, estão postos diretamente nos caminhos do URI. Vale ressaltar que "pessoa" pode ser tanto um valor fixo no URI (linha 3 da Figura 29), quanto uma variável (outro parâmetro). Neste último caso, o URI seria composto de três *Path Parameters*. Um exemplo de implementação de *Path Parameters* está ilustrado nas linhas 14-21 da Figura 29.

O *Semantic Adapter* (ver seção 4.2.3) busca no código do serviço pelas anotações JAX-RS visando extrair os *endpoints* necessários para a construção da descrição semântica. É com base nas anotações que o *Semantic Adapter* é capaz de gerar corretamente os *URITemplates* e os mapeamentos da descrição Hydra. O método HTTP definido pelo `supportedOperation` do Hydra também

[1] Disponível em: <https://docs.oracle.com/cd/E19798-01/821-1841/giepu/index.html>.

[2] Codificação que representa o espaço em branco em URIs: `%20`

é possível obter por meio das anotações. Os métodos `addPessoa` e `delPessoaPorParam` da Figura 29 ilustram exemplos de *endpoints* invocados pelos métodos PUT e DELETE, respectivamente. É importante notar que é possível definir operações de CRUD (*Create, Read, Update and Delete*) utilizando apenas um *endpoint* padrão. O que diferencia a operação é o método HTTP, como é o caso do GET (linhas 14-21) e DELETE (linhas 35-41), em que o mesmo URI é utilizado para operações distintas.

Figura 29 – Trecho de código utilizando JAX-RS definindo *endpoints* com *Query* e *Path Parameters*.

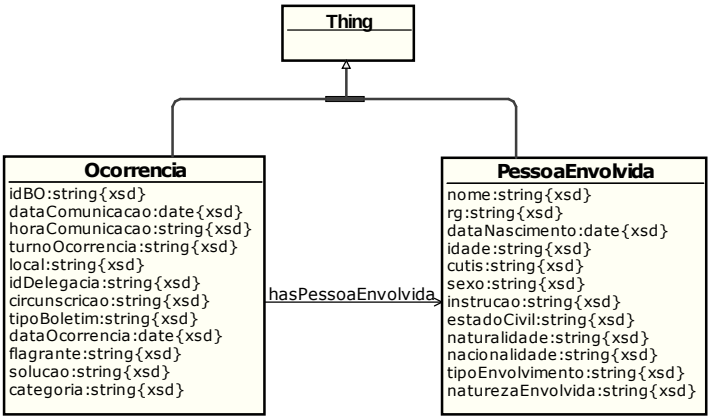
```
1  import javax.ws.rs.*;
2
3  @Path("/pessoa")
4  public class RercursoPessoa {
5   // ...
6          @GET
7          @Produces("text/json")
8          public Response getPessoasPorQuery(@QueryParam("estado") String estado,
9                                             @QueryParam("profissao") String profissao) {
10          /* Código para buscar pessoas de determinada profissao
11          **/
12         }
13
14         @GET
15         @Path("{rg}/{estado}")
16         @Produces("text/json")
17         public Response getPessoaPorParam(@PathParam("rg") Integer rg,
18                                           @PathParam("estado") String estado) {
19          /* Código para buscar a pessoa
20          **/
21         }
22
23         @PUT
24         @Path("{rg}/{estado}/{nome}/{nascimento}")
25         @Produces("text/json")
26         public Response addPessoa(@PathParam("rg") Integer rg,
27                                   @PathParam("estado") String estado,
28                                   @PathParam("nome") String nome,
29                                   @PathParam("profissao") Integer profissao
30                                   /* ... */ ) {
31          /* Código para adicionar uma pessoa
32          **/
33         }
34
35         @DELETE
36         @Path("{rg}/{estado}")
37         public Boolean delPessoaPorParam(@PathParam("rg") Integer rg,
38                                          @PathParam("estado") String estado) {
39          /* Código para remover uma pessoa
40          **/
41         }
42 }
```

Fonte: Elaborado pelo autor (2018).

APÊNDICE D – EXEMPLO DE ONTOLOGIA CONSTRUÍDA

A Figura 30 ilustra um diagrama baseado na UML, construído a partir da ferramenta OWLGrEd (LIEPINS; GRASMANIS; BOJARS, 2014), contendo as entidades (classes e propriedades) de uma ontologia de domínio. Este diagrama foi gerado com base na ontologia construída pelo OntoGenesis, após algumas requisições feitas para um determinado serviço de dados. A Figura 31 apresenta a ontologia gerada representada em RDF/XML. Tal ontologia inclui classes, propriedades de dados e de objetos, bem como as triplas de equivalência de propriedades. Exemplos de equivalências falso positivas incorporadas na ontologia podem ser observados nas linhas 56, 125 e 137 da Figura 31. Tais equivalências foram incluídas em decorrência da força do *match* ter sido superior ao *threshold* α definido para o experimento pelo qual a ontologia de domínio foi obtida.

Figura 30 – Diagrama de uma ontologia gerada pelo OntoGenesis.



Fonte: Elaborado pelo autor (2018).

Figura 31 – Exemplo de uma ontologia gerada em RDF/XML.

```
1  <rdf:RDF
2      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3      xmlns:owl="http://www.w3.org/2002/07/owl#"
4      xmlns="http://127.0.0.1:8081/criminal-report-service/ontology#"
5      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
6      xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
```

```
7     <owl:Ontology
       ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology"/>
8
9     <owl:Class rdf:about=
       ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida">
10      <rdfs:comment xml:lang="EN">Class PessoaEnvolvida</rdfs:comment>
11    </owl:Class>
12
13    <owl:Class rdf:about=
       ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia">
14      <rdfs:comment xml:lang="EN">Class Ocorrencia</rdfs:comment>
15    </owl:Class>
16
17    <owl:ObjectProperty rdf:about=
       ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#hasPessoaEnvolvida">
18      <rdfs:range rdf:resource=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
19      <rdfs:domain rdf:resource=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
20    </owl:ObjectProperty>
21
22
23    <owl:DatatypeProperty
       ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#idBO">
24      <rdfs:domain rdf:resource=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
25          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
26    </owl:DatatypeProperty>
27
28    <owl:DatatypeProperty rdf:about=
       ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#idDelegacia">
29      <rdfs:domain rdf:resource=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
30          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
31      <owl:equivalentProperty
         ↪   rdf:resource="http://purl.org/dc/elements/1.1/description"/>
32    </owl:DatatypeProperty>
33
34    <owl:DatatypeProperty
       ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#circunscricao">
35      <rdfs:domain rdf:resource=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
36          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
37      <owl:equivalentProperty rdf:resource="http://xmlns.com/foaf/0.1/name"/>
38      <owl:equivalentProperty
         ↪   rdf:resource="http://purl.org/dc/elements/1.1/description"/>
39    </owl:DatatypeProperty>
40
41    <owl:DatatypeProperty
       ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#tipoBoletim">
42      <rdfs:domain rdf:resource=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
43          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
44      <owl:equivalentProperty
         ↪   rdf:resource="http://purl.org/dc/elements/1.1/description"/>
45    </owl:DatatypeProperty>
46
47    <owl:DatatypeProperty
       ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#categoria">
48      <rdfs:domain rdf:resource=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
49          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
50      </owl:DatatypeProperty>
51
52      <owl:DatatypeProperty
         ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#local">
53        <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
54            <rdfs:range rdf:resource= "http://www.w3.org/2001/XMLSchema#string"/>
55        <owl:equivalentProperty rdf:resource="http://xmlns.com/foaf/0.1/name"/>
56        <owl:equivalentProperty
           ↪   rdf:resource="http://xmlns.com/foaf/0.1/surname"/>
57        <owl:equivalentProperty
           ↪   rdf:resource="http://www.geonames.org/ontology#name"/>
58      </owl:DatatypeProperty>
59
60      <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#turnoOcorrencia">
61        <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
62            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
63        <owl:equivalentProperty
           ↪   rdf:resource="http://purl.org/dc/elements/1.1/description"/>
64      </owl:DatatypeProperty>
65
66      <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#dataOcorrencia">
67        <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
68            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
69        <owl:equivalentProperty
           ↪   rdf:resource="http://dbpedia.org/ontology/deathDate"/>
70        <owl:equivalentProperty
           ↪   rdf:resource="http://dbpedia.org/ontology/birthDate"/>
71      </owl:DatatypeProperty>
72
73      <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#dataComunicacao">
74        <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
75            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
76        <owl:equivalentProperty
           ↪   rdf:resource="http://dbpedia.org/ontology/deathDate"/>
77        <owl:equivalentProperty
           ↪   rdf:resource="http://dbpedia.org/ontology/birthDate"/>
78      </owl:DatatypeProperty>
79
80      <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#horaComunicacao">
81        <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
82            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
83      </owl:DatatypeProperty>
84
85      <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#flagrante">
86        <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
87            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
88      </owl:DatatypeProperty>
89
90      <owl:DatatypeProperty
         ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#solucao">
91        <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#Ocorrencia"/>
```

```
92          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
93        <owl:equivalentProperty
           ↪   rdf:resource="http://purl.org/dc/elements/1.1/description"/>
94      </owl:DatatypeProperty>
95
96      <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#tipoEnvolvimento">
97        <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
98          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
99      </owl:DatatypeProperty>
100
101     <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#naturezaEnvolvida">
102       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
103         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
104       <owl:equivalentProperty
           ↪   rdf:resource="http://purl.org/dc/elements/1.1/description"/>
105     </owl:DatatypeProperty>
106
107     <owl:DatatypeProperty
         ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#nome">
108       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
109         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
110       <owl:equivalentProperty rdf:resource="http://xmlns.com/foaf/0.1/name"/>
111       <owl:equivalentProperty
           ↪   rdf:resource="http://xmlns.com/foaf/0.1/givenName"/>
112       <owl:equivalentProperty
           ↪   rdf:resource="http://www.geonames.org/ontology#name"/>
113     </owl:DatatypeProperty>
114
115      <owl:DatatypeProperty
          ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#rg">
116       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
117         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
118       <owl:equivalentProperty rdf:resource=
           ↪   "http://www.omg.org/spec/EDMC-FIBO/FND/AgentsAndPeople/People/
           ↪   NationalIdentificationNumber"/>
119     </owl:DatatypeProperty>
120
121     <owl:DatatypeProperty
         ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#dataNascimento">
122       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
123         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
124       <owl:equivalentProperty
           ↪   rdf:resource="http://dbpedia.org/ontology/deathDate"/>
125       <owl:equivalentProperty
           ↪   rdf:resource="http://dbpedia.org/ontology/birthDate"/>
126     </owl:DatatypeProperty>
127
128     <owl:DatatypeProperty
         ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#idade">
129       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
130         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
131     </owl:DatatypeProperty>
132
133     <owl:DatatypeProperty
         ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#naturalidade">
```

```
134       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
135           <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
136       <owl:equivalentProperty rdf:resource="http://xmlns.com/foaf/0.1/name"/>
137       <owl:equivalentProperty
           ↪   rdf:resource="http://www.geonames.org/ontology#name"/>
138     </owl:DatatypeProperty>
139
140     <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#nacionalidade">
141       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
142           <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
143       <owl:equivalentProperty
           ↪   rdf:resource="http://www.geonames.org/ontology#name"/>
144     </owl:DatatypeProperty>
145
146     <owl:DatatypeProperty
         ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#cutis">
147       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
148           <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
149       <owl:equivalentProperty
           ↪   rdf:resource="http://dbpedia.org/ontology/skinColor"/>
150     </owl:DatatypeProperty>
151
152     <owl:DatatypeProperty
         ↪   rdf:about="http://127.0.0.1:8081/criminal-report-service/ontology#sexo">
153       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
154           <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
155       <owl:equivalentProperty rdf:resource=
           ↪   "http://www.omg.org/spec/EDMC-FIBO/FND/AgentsAndPeople/People/hasGender"/>
156     </owl:DatatypeProperty>
157
158     <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#estadoCivil">
159       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
160           <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
161       <owl:equivalentProperty rdf:resource="http://xmlns.com/foaf/0.1/name"/>
162       <owl:equivalentProperty
           ↪   rdf:resource="http://purl.org/dc/elements/1.1/description"/>
163     </owl:DatatypeProperty>
164
165     <owl:DatatypeProperty rdf:about=
         ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#instrucao">
166       <rdfs:domain rdf:resource=
           ↪   "http://127.0.0.1:8081/criminal-report-service/ontology#PessoaEnvolvida"/>
167           <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
168       <owl:equivalentProperty
           ↪   rdf:resource="http://purl.org/dc/elements/1.1/description"/>
169     </owl:DatatypeProperty>
170   </rdf:RDF>
```

Fonte: Elaborado pelo autor (2018).