Pós-Graduação em Ciência da Computação

Diego Ernesto Rosa Pessoa

Um Framework para Recomendação de Matchers de Ontologias baseado em Requisitos da Aplicação

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Diego Ernesto Rosa Pessoa

Um Framework para Recomendação de Matchers de Ontologias baseado em Requisitos da Aplicação

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Banco de Dados
Orientadora: Ana Carolina Salgado
Coorientadora: Bernadette Farias Lóscio

Recife
2019

Catalogação na fonte
Bibliotecária Mariana de Souza Alves, CRB4-2106

P475f Pessoa, Diego Ernesto Rosa
Um Framework para Recomendação de Matchers de Ontologias baseado em Requisitos da Aplicação/ Diego Ernesto Rosa Pessoa – 2019.
152 f.:il., fig., tab.

Orientadora: Ana Carolina Salgado.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências e apêndices.

1. Inteligência artificial. 2. Alinhamento de Ontologias. 3. Recomendação de Matchers. I. Salgado, Ana Carolina (orientadora). II. Título.

006.31 CDD (23. ed.) UFPE- MEI 2019-091

Diego Ernesto Rosa Pessoa

"Um Framework para Recomendação de Matchers de Ontologias baseado em Requisitos da Aplicação"

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Aprovado em: 14/03/2019.

_____

Orientadora: Profa. Dra. Ana Carolina Brandão Salgado

BANCA EXAMINADORA

_____

Profa. Dra. Patrícia Cabral de Azevedo Restelli Tedesco
Centro de Informática /UFPE

_____

Prof. Dr. Altigran Soares da Silva,
Instituto de Computação / UFAM

_____

Profa. Dra. Damires Yluska de Souza Fernandes
Unidade Acadêmica de Informática / IFPB

_____

Profa. Dra. Fernanda Araújo Baião Amorim
Departamento de Informática / PUC/RJ

_____

Profa. Dra. Maria Claudia Reis Cavalcanti
Departamento de Engenharia de Sistemas / IME

À minha esposa Nina, ao meu filho Murilo e à minha filha Marina. O amor de vocês é a minha fortaleza. A compreensão, paciência e o suporte foram fundamentais para que fosse possível alcançar os meus objetivos.

AGRADECIMENTOS

Meu coração se enche de gratidão ao concluir esta importante etapa na minha vida. A trajetória foi desafiadora, mas fico feliz em poder dar orgulho aos que me apoiaram e estiveram juntos comigo nessa jornada.

Presto os meus sinceros agradecimentos primeiramente a Deus, por ter iluminado o meu caminho e ter me dado saúde e força para seguir em frente.

Agradeço à minha amada esposa, Nina Rosa, que esteve incondicionalmente ao meu lado, com quem eu pude compartilhar de perto a alegria de cada etapa vencida, como também a tensão nos momentos mais difíceis. O seu apoio foi fundamental para me fazer conseguir atingir os objetivos. Quando entrei no doutorado éramos apenas namorados, hoje me sinto agraciado em chegar até aqui casado e com dois filhos maravilhosos. Se não fosse a tua força para conseguir resistir nos momentos em que estive distante, certamente não teria conseguido. Obrigado por tudo, meu amor!

Agradeço aos meus filhos, Murilo e Marina, que por mais que ainda sejam dois bebês, foram o principal combustível nesta reta final. Essa fonte inesgotável de amor e carinho foi responsável por revigorar as minhas forças. Espero que esta conquista possa servir de inspiração para o futuro de vocês.

Obrigado à minha mãe, Sonely, por ter abdicado de tanto em prol da minha criação e educação, sempre com muito carinho. Agradeço também ao meu pai, Joaquin, pelo apoio e incentivo ao longo de todos esses anos. Fico feliz em poder deixar vocês orgulhosos dessa conquista. Espero agora poder estar com vocês mais frequentemente.

Agradeço especialmente à minha orientadora Ana Carolina Salgado, pela sua lucidez, sabedoria e paciência. As lições aprendidas durante esses anos eu levarei para sempre em minha vida. Obrigado por me aceitar como orientando e acreditar na minha capacidade. As suas colocações sempre foram muito pertinentes, sendo fundamentais para a construção desta tese.

Agradeço à minha coorientadora Bernadette Farias Lóscio, pelas contribuições e *insights* que foram fundamentais para que eu pudesse encontrar o caminho para a minha pesquisa. A sua capacidade técnica e a forma como você lida com os desafios tão naturalmente me serviram de inspiração durante esta caminhada.

Sou grato aos meus colegas da Toca dos Dados, que o doutorado me proporcionou conhecer. Em especial Gabrielle, Neto, Marcelo e Priscilla. Juntos pudemos compartilhar desde as lamúrias até os momentos de descontrações, que por muitas vezes ajudaram a aliviar a tensão. As palavras de incentivo e a ajuda mútua durante o convívio também foram fundamentais.

Deixo minha gratidão também às professoras Fernanda Baião, Damires Souza e Patrícia Tedesco, pela disposição em participar do meu exame de qualificação e da minha defesa.

Muito obrigado pela dedicação e atenção ao meu trabalho. As observações feitas ainda na qualificação foram primordiais para a construção da versão final desta tese. À professora Fernanda Baião, destaco ainda o agradecimento pelas valiosas contribuições para o trabalho durante o ISWC em 2017. Agradeço também ao professor Altigran da Silva e à professora Maria Cláudia Cavalcanti (Yoko), por terem aceito o convite para participar da minha defesa. Ao professor Altigran, estendo meus agradecimentos por ter se disposto a ouvir a minha proposta de tese, durante o AMW de 2016, contribuindo para o direcionamento da minha pesquisa em um momento decisivo. Agradeço também ao professor Ricardo Baeza-Yates, que pude conhecer no AMW e me deu orientações importantes a respeito do meu trabalho.

Agradeço ao IFPB e à Unidade de Informática por ter me concedido o afastamento. Agradeço também aos colegas do IFPB, em especial Gabriela, Aristofânio, Paulo e ao coordenador de Unidade Ricardo Job, que contribuíram para que eu pudesse focar exclusivamente na tese nos momentos finais. Agradeço também ao colega Daladier, com quem pude dividir a estadia em Recife nos primeiros anos de doutorado.

No mais, deixo o meu profundo agradecimento a todas as pessoas que contribuíram de alguma forma para a concretização desta tese.

RESUMO

Alinhamento de ontologias é o processo de busca por correspondências entre elementos de ontologias, geralmente, disponibilizadas por fontes de dados autônomas, heterogêneas e distribuídas. Apesar de diversos sistemas (denominados *matchers*) terem sido propostos nos últimos anos, não há um que forneça melhores resultados para todos os cenários possíveis. A utilização de *matchers* mais adequados é uma forma de melhorar a qualidade dos alinhamentos ao lidar com conjuntos de dados diversificados. Neste sentido, este trabalho propõe um framework com o objetivo de viabilizar a recomendação de *matchers* para aplicação específicas. Para tal, utiliza-se a extração de segmentos de ontologias com base em requisitos da aplicação, como forma de possibilitar a redução do espaço de busca e favorecer a combinação de *matchers*, favorecendo a uma maior precisão na recomendação. Para a avaliação das soluções propostas, foram constituídos catálogos contendo ontologias de diferentes conjuntos de dados e várias combinações de *matchers* existentes. Os resultados obtidos através de experimentos indicam uma melhoria na qualidade dos alinhamentos obtidos, em comparação com o estado da arte.

Palavras-chaves: Alinhamento de Ontologias. Segmentação de Ontologias. Recomendação de *Matchers*. Web Semântica.

ABSTRACT

Ontology matching is the task of finding relations between elements of ontologies provided by distributed and heterogeneous data sources. Despite the emergence of many matching tools (or matchers) in the last years, no single matcher performs better than all others in a wide range of scenarios. Using a set of suitable matchers is a way to improve the quality of alignments, especially for diversified datasets. In this sense, this work proposes a framework to provide the recommendation of matchers for a particular application setting. For such, we use the extraction of ontology segments based on application requirements, so that it reduces the search space of alignments, enabling the combination of matchers, which consequently makes more accurate the recommendation process. For evaluating the proposed solutions, we built two data catalogs, one containing ontologies from different knowledge domains and other containing several combinations of existing matchers. The results obtained from the experiments indicate that the proposed solutions provide alignments with better quality, in comparison with state-of-the-art techniques.

Keywords: Ontology Matching. Ontology Segmentation. Matchers Recommendation. Semantic Web.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CPR Censor Production Rules.

DOLCE Descriptive Ontology for Linguistic and Cognitive Engineering.

GFO General Formal Ontology.

KNN K-Nearest Neighbour.

KSMS Knowledge-based Schema Matching System for Ontology Matching.

MCMA Multilevel Characteristic for Matching Approaches.

OAEI Ontology Alignment Evaluation Initiative.

SF Similarity Flooding.

SMB Schema Matcher Boosting.

SUMO Upper Merged Ontology.

UFO Unified Foundational Ontology.

WFOV Well-founded Ontology View.

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo é descrito o contexto em que se insere a pesquisa proposta. Inicialmente, são apresentadas as principais motivações para a realização da pesquisa. Em seguida, define-se o problema investigado, os objetivos da pesquisa e as hipóteses. Por fim, detalha-se a organização geral do documento.

1.1 MOTIVAÇÃO E DEFINIÇÃO DO PROBLEMA

O crescimento da publicação de dados por diferentes fontes tem resultado numa ampla gama de ontologias sendo desenvolvidas para representar domínios ou aplicações específicas. Tendo em vista que aplicações são independentes, e que os usuários humanos, geralmente, se encarregam de modelar as ontologias, é comum encontrar termos e estruturas diferentes sendo usados para escrever os mesmos conceitos de mundo-real. Tais diferenças podem criar dificuldades na integração de fontes com dados modelados a partir de diferentes ontologias. Devido a este problema, é necessário comparar ontologias no intuito de descobrir correspondências entre os seus elementos - este processo é conhecido na literatura como Alinhamento de Ontologias (*Ontology Matching*). Um exemplo de tipo de correspondência é a equivalência, que ocorre quando dois elementos de ontologias diferentes representam o mesmo conceito. A Figura 1 ilustra a representação de um alinhamento para duas ontologias. As setas em azul indicam a o ocorrência de elementos correspondentes (e.g., name e title ou Science e Essay).

Mais especificamente, Alinhamento de Ontologias é o processo que visa encontrar correspondências entre elementos presentes em diferentes ontologias. Para tal, recebe-se como entrada um par de ontologias a serem alinhadas (O_1, O_2) e gera-se como resultado um alinhamento (A). Opcionalmente, também pode ser fornecido como entrada um alinhamento de referência (A'). O processo pode ser afinado por meio de parâmetros de configuração, que podem indicar, por exemplo, o valor mínimo de similaridade para considerar um determinado par de elementos como correspondentes. Além disso, como forma de melhorar a eficiência na geração dos alinhamentos, também pode-se usar como referência recursos externos, tais como ontologias de domínio e repositórios de conhecimento (e.g., Wordnet[1]). A Figura 2 ilustra esquematicamente a definição do processo de alinhamento de ontologias.

Recentemente, o problema de alinhamento de ontologias tem se tornado ainda mais desafiador, levando em consideração o aumento do número de fontes e de vocabulários disponíveis para descrever os dados publicados. Isto resulta em um aumento da quantidade de elementos a comparar, além da possibilidade de obtenção de um alto número de

[1] Site do Wordnet acessível pelo endereço: <http://wordnetweb.princeton.edu/perl/webwn?s=repository>

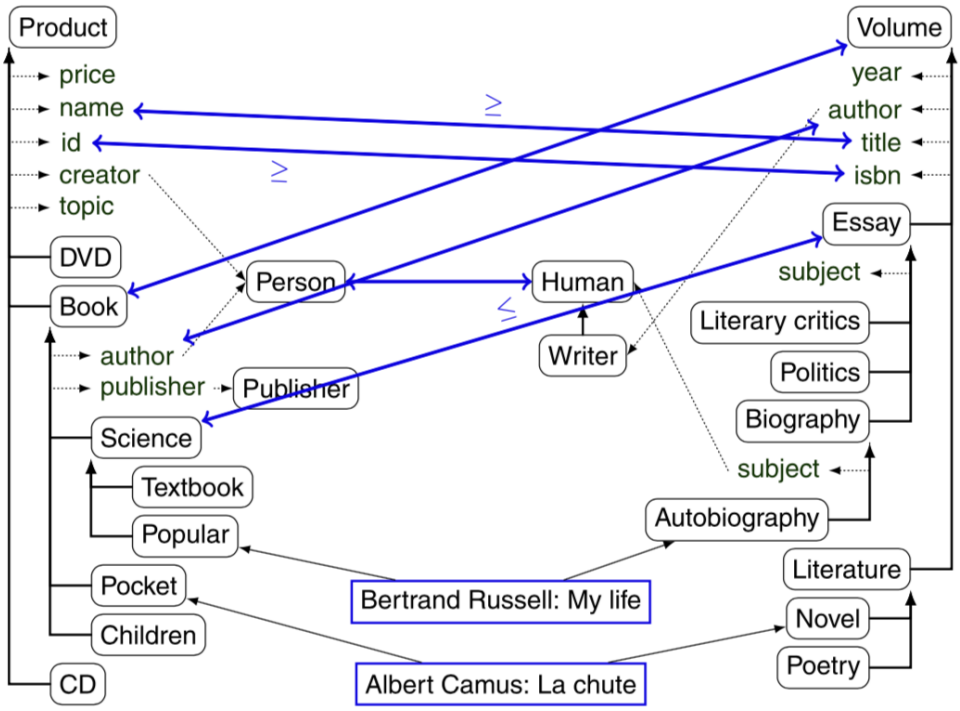


Figura 1 – Exemplo do Processo de Alinhamento de Ontologias

Fonte – (EUZENAT; SHVAIKO, 2013)

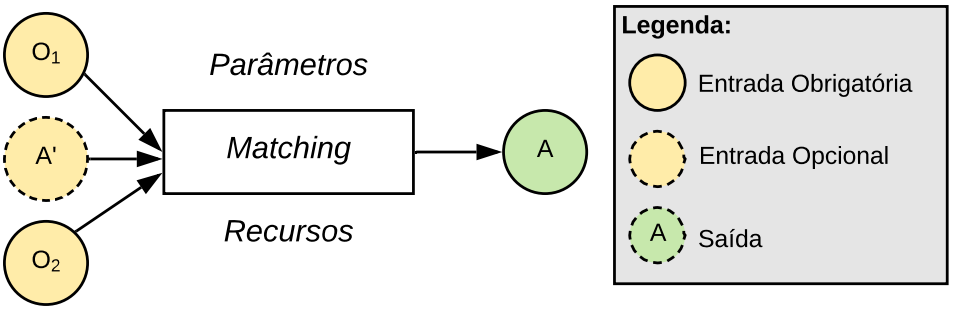


Figura 2 – Estrutura básica do processo de Alinhamento de Ontologias

Fonte – Adaptado de Euzenat e Shvaiko (2013)

correspondências, o que tornaria a verificação manual um processo demorado e custoso ao usuário.

A aplicabilidade desta área de pesquisa está relacionada tanto a aplicações de integração de dados tradicionais, quanto a aplicações modernas. Tipicamente, aplicações tradicionais, como as que lidam com integração de ontologias, integração de esquemas ou *data warehouses*, são modeladas usando estruturas heterogêneas. Geralmente, para integrá-las, seus elementos são analisados e comparados manualmente ou semi-automaticamente ainda na etapa de projeto. Ou seja, em tais aplicações, realizar os alinhamentos é um pré-requisito para a execução. Por outro lado, aplicações mais recentes requerem a geração dos alinhamentos em tempo de execução, demandando modelos conceituais mais comple-

xos. Alguns exemplos dessas aplicações são *Web Service Composition* (LEMOS; DANIEL; BENATALLAH, 2015), *Linked Data*, navegação na Web Semântica e resposta de consultas na Web (GANDON, 2018). O alinhamento de ontologias também pode beneficiar outros processos provenientes da integração de dados, tais como a reescrita de consultas em fontes heterogêneas, resolução de dados e fusão de dados (DONG; SRIVASTAVA, 2015), visto que o foco desses processos concentra-se nas instâncias de dados, frequentemente assumindo que as ontologias consideradas já estão integradas.

Como parte dos estudos desenvolvidos ao longo dos últimos anos, diversas técnicas e ferramentas foram propostas com o objetivo de automatizar etapas do processo de Alinhamento de Ontologias (SHVAIKO; EUZENAT, 2013; OTERO-CERDEIRA; RODRÍGUEZ-MARTÍNEZ; GÓMEZ-RODRÍGUEZ, 2014; OCHIENG; KYANDA, 2018c). Dada a diversidade de soluções, a avaliação dos sistemas de *Ontology Matching* (também conhecidos como *matchers*) é considerada uma tarefa bastante complexa. Como principal iniciativa neste sentido, pode-se citar a *Ontology Alignment Evaluation Initiative (OAEI)*[2], que realiza campanhas anuais para a avaliação de ferramentas de *Ontology Matching*. Para isso, são disponibilizados alguns conjuntos de dados heterogêneos de um mesmo domínio de conhecimento, juntamente com uma referência do que seria correspondente entre eles. As técnicas que mais se aproximarem do ideal, seguindo métricas já consolidadas (*precision*, *recall* e *f-measure*), atingindo o menor tempo para obter o resultado, serão mais bem classificadas. Das últimas edições do OAEI, podemos destacar uma evolução considerável nos sistemas participantes. Em 2009, o melhor sistema havia atingido apenas 63% de *f-measure* (EUZENAT et al., 2009). Já na edição de OAEI (2017), o melhor desempenho chegou a 94% de *f-measure* em algumas trilhas, o que é muito próximo do ideal (ACHICHI et al., 2017).

Conforme relatado por Achichi et al. (2017), em algumas modalidades da OAEI houve o caso de sistemas que levaram várias horas ou até mesmo não conseguiram concluir a sua execução. Além disso, os conjuntos de dados de avaliação considerados são fixos, de acordo com a modalidade. Mesmo assim, nota-se, a partir dos resultados obtidos, que não há um sistema de *ontology matching* dominante, ou seja, que obtenha melhores resultados do que os outros, independentemente do modelo de dados e do domínio de aplicação (OCHIENG; KYANDA, 2018c). Pode-se perceber também que o número de *matchers* cresce continuamente, e a diversidade por si só, já dificulta a escolha da ferramenta mais apropriada para um domínio de aplicação específico. Isso tem como consequência a motivação para abordar um problema que vem sendo discutido recentemente: a recomendação e combinação de *matchers*.

Apesar de diversas ferramentas (e.g. COMA (MASSMANN et al., 2011) e S-Match (GIUNCHIGLIA; AUTAYEU; PANE, 2012)) combinarem diferentes *matchers* para determinar a geração dos alinhamentos, parte dessas soluções são baseadas em heurísticas fixas e

[2] Site da OAEI acessível pelo endereço: <http://oaei.ontologymatching.org>

seleção manual, i.e., uma vez escolhida (geralmente por parâmetros de entrada), a mesma combinação é aplicada a qualquer cenário. Por outro lado, há trabalhos que aplicam técnicas de aprendizagem de máquina, o que pode favorecer a geração de um modelo diferente para cada cenário. YAM/YAM++ (DUCHATEAU; BELLAHSENE, 2015) e ALMa (RODRIGUES et al., 2015) são exemplos desse tipo de abordagem. Neste caso, normalmente é necessário estabelecer um conjunto de treinamento prévio (como é o caso das técnicas de aprendizagem supervisionada), o que pode ser bastante dispendioso em cenários de larga escala, visto que um pequeno conjunto de treinamento dificilmente conseguirá cobrir uma grande variedade de cenários.

Outra estratégia abordada na literatura é a utilização de metadados de um conjunto pré-determinado de *matchers* e das *ontologias* a serem alinhadas para recomendar os *matchers* mais adequados. O trabalho proposto por Mochol e Jentzsch (2008) segue essa linha, onde é especificado um *framework* baseado em regras que indicam relações entre metadados dos *matchers* e metadados das ontologias. Neste caso, o usuário precisa definir e priorizar um amplo conjunto de requisitos (através da atribuição manual de pesos); não é fornecida nenhuma classificação ou lista de características dos *matchers*, e não há o conhecimento de experimentos demonstrando a efetividade da abordagem proposta.

Para obter evidências mais claras quanto à relevância de um *matcher* para a geração dos alinhamentos, outros critérios podem ser utilizados, tais como a qualidade das correspondências geradas automaticamente e do custo necessário para a execução (e.g., tempo de execução, recursos de *hardware*). Porém, é importante ressaltar que a forma como esses critérios são avaliados pode variar de acordo com a aplicação interessada. Por exemplo, para algumas aplicações, pode ser mais interessante obter um grande número de correspondências, visando contemplar uma maior quantidade de elementos nos alinhamentos, mesmo que isso acarrete na obtenção de um maior número de falsos positivos. Por outro lado, outras aplicações podem optar por obter apenas um menor conjunto contendo as correspondências com maior nível de similaridade, minimizando a intervenção do usuário durante a validação dos alinhamentos gerados. Além disso, dependendo de alguns aspectos inerentes ao processo, como as heterogeneidades identificadas entre os elementos das ontologias comparadas, um determinado *matcher* pode apresentar variações de desempenho, o que dificulta a recomendação.

1.2 QUESTÕES DE PESQUISA E OBJETIVO

Este trabalho visa compreender os principais fatores relacionados às ontologias que impactam na geração de alinhamentos e, a partir disso, recomendar os *matchers* mais eficientes e combiná-los para a geração de alinhamentos entre pares de ontologias fornecidos por uma aplicação. Um exemplo de aplicação, considerando o domínio de conferências, seria um sistema que realize a integração de dados de eventos distintos (e.g., título, localização, data) para fins de divulgação em massa. Neste caso, seria requerida apenas a integração de dados

básicos relacionados à conferência, dispensando detalhes adicionais comumente presentes nos modelos de dados deste domínio, como artigos publicados, autores e revisores.

Nesta perspectiva, embasado nas colocações acima expostas, esta pesquisa cumpre questionar:

Q1 *Quais características das ontologias impactam no desempenho dos matchers, para que seja possível estabelecer uma correlação com amostras de alinhamentos gerados anteriormente?*

Q2 *Como extrair segmentos de ontologias a partir de requisitos de uma aplicação, visando viabilizar a combinação de matchers no intuito de obter uma maior eficiência na geração de alinhamentos?*

Q3 *Como medir a qualidade dos matchers para resolver alinhamentos em um cenário caracterizado por uma aplicação, como forma de dar suporte ao mecanismo de recomendação?*

Objetivando responder a essas questões, torna-se imprescindível o conhecimento das técnicas de alinhamento de ontologias propostas na literatura e dos principais aspectos que possam auxiliar no processo de decisão de como essas técnicas podem ser selecionadas com base em requisitos da aplicação. Assim, o objetivo principal do trabalho descrito nesta tese é:

Objetivo de Pesquisa *Prover a recomendação dos matchers mais adequados com base nos requisitos de uma aplicação, definidos no contexto de uma atividade de alinhamento de ontologias.*

1.3 VISÃO GERAL DA SOLUÇÃO PROPOSTA

Como forma de cumprir o objetivo de pesquisa, este trabalho apresenta o algoritmo de segmentação de ontologias SOMA (*Segmentation of Ontologies for Matching Applications*) e o framework Redoma (*Requirement-driven Ontology Matching*), que aborda o problema da recomendação de *matchers* através da correlação de alinhamentos a partir de métricas envolvendo as ontologias comparadas. Para reduzir o escopo de busca para a geração de alinhamentos, visando aumentar a precisão no cálculo dessas métricas, propõe-se trabalhar com segmentos de ontologias ao invés das ontologias inteiras. Na geração dos segmentos, introduz-se o conceito de requisitos da aplicação, que trata-se de um conjunto de conceitos e de parâmetros de configurações que refletem o interesse do usuário especialista naquele processo. Por fim, a partir da construção de uma base de conhecimento contendo diversas amostras da geração de alinhamentos por vários *matchers* existentes, chega-se à recomendação através da concepção de um perfil de qualidade para os *matchers*. Como diferencial, a construção do perfil é realizada sob-demanda, no qual os

matchers são avaliados quanto ao seu desempenho para gerar alinhamentos entre ontologias correlacionadas às fornecidas, i.e., que possuem estrutura ou características próximas.

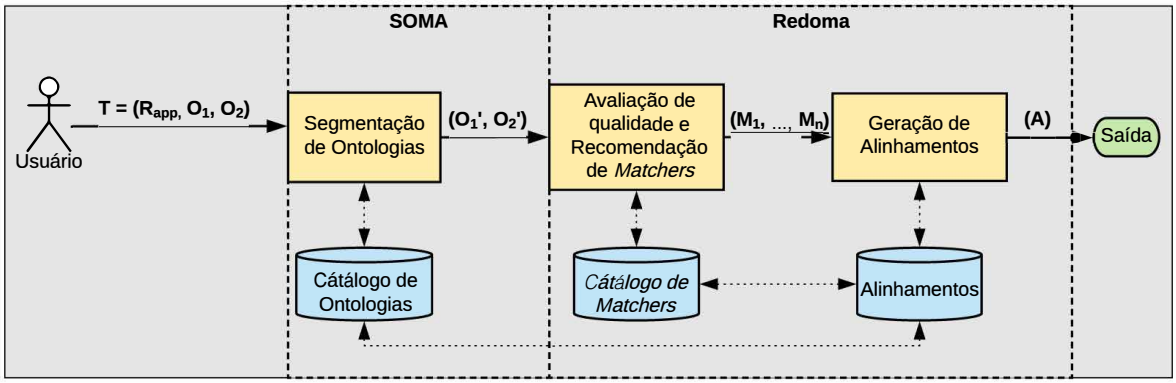


Figura 3 – Fluxo de trabalho envolvendo o algoritmo SOMA e o *framework* Redoma

A Figura 3 ilustra o fluxo de trabalho do processo de geração de alinhamentos utilizando o algoritmo SOMA e o *framework* Redoma, o que envolve a extração de segmentos de ontologias e a recomendação de *matchers*. Para iniciar, o usuário fornece como entrada um conjunto de requisitos da aplicação (R_{app}) e um par de ontologias (O_1, O_2). Na primeira etapa, são extraídos os elementos das ontologias que estão relacionados aos requisitos, resultando nos segmentos (O_1', O_2'). Em seguida, um conjunto de métricas é calculado em cima desses segmentos (denominado Perfil da Ontologia)) e o resultado é armazenado numa base de dados local (denominado catálogo de ontologias). Na segunda etapa, constrói-se, em tempo de execução, um perfil de qualidade para os *matchers* disponíveis numa base local (denominada catálogo de *matchers*), levando em consideração o histórico de alinhamentos envolvendo ontologias que se correlacionam (a partir do perfil das ontologias) às fornecidas como entrada. Retorna-se então como resultado a lista com os melhores avaliados (M_1, ... ,M_n) para resolver o alinhamento em questão. Por fim, na última etapa, é realizada a execução dos recomendados, gerando alinhamentos (A) que são fornecidos como saída final para o usuário, como também são armazenados na base local para servir de referência para próximas execuções.

1.4 CONTRIBUIÇÕES

Levando em conta a problemática exposta e a solução proposta, enumeram-se as seguintes contribuições desta pesquisa:

i. Levantamento de métricas que podem ser utilizadas para classificar ontologias durante a geração de alinhamentos;

ii. Definição de um modelo para especificação de requisitos da aplicação no que concerne o processo de alinhamento de ontologias;

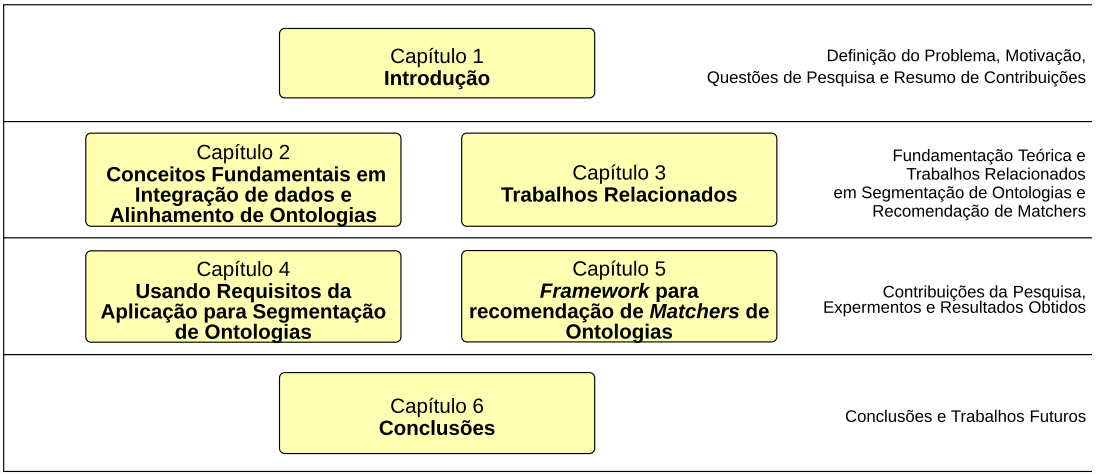
iii. Especificação e implementação do algoritmo SOMA para geração de segmentos de ontologias a partir de um conjunto de requisitos;

iv. Especificação e implementação do *framework* Redoma para recomendação de *matchers* adequados a resolver alinhamentos no contexto de uma aplicação específica.

1.5 ORGANIZAÇÃO DO DOCUMENTO

No que concerne a organização desta tese, a Figura 4 descreve, esquematicamente, os capítulos organizados por temática. Em seguida, apresenta-se de maneira sucinta o que é abordado em cada capítulo.

Figura 4 – Organização da Tese



O Capítulo 2 apresenta os conceitos fundamentais que envolvem a área de Alinhamento de Ontologias. Além da definição básica de ontologias e do problema de alinhamento como parte do processo de Integração de Dados, destaca-se a definição das métricas para avaliação de *matchers*, como também as métricas para classificação de ontologias.

O Capítulo 3 mostra um levantamento dos trabalhos relacionados nas áreas de segmentação de ontologias e recomendação de *matchers*, que se apresentam como o escopo central desta pesquisa. Ao final das seções respectivas à cada área, descrevem-se, a partir de um quadro comparativo, as principais características de cada trabalho, os seus benefícios e as possíveis desvantagens.

O Capítulo 4 descreve o algoritmo proposto para segmentação de ontologias com base em requisitos da aplicação. Neste, primeiramente, são formalizados alguns conceitos básicos ao contexto deste trabalho, tais como, segmento de ontologia e requisito da aplicação, como também são introduzidos os princípios de conservação utilizados durante a geração de segmentos. Em seguida, descreve-se cada etapa prevista no algoritmo, ilustrando os resultados obtidos através de exemplos. Além disso, é abordado o planejamento, a configuração e a execução dos experimentos conduzidos para avaliar o algoritmo proposto.

O Capítulo 5 descreve o *framework* proposto para a recomendação de *matchers*. Apresenta-se, inicialmente, uma visão geral do funcionamento do *framework*, para em seguida detalhar cada um dos componentes que compõem o seu fluxo principal. Neste contexto, os componentes do *framework* contemplam etapas como a construção do perfil de ontologias e a construção do perfil, avaliação, recomendação, seleção e execução de *matchers*, além da validação dos alinhamentos gerados. Ao término do capítulo, são descritos os experimentos realizados para a avaliação do *framework* e discutem-se os resultados obtidos, que são ilustrados por intermédio de gráficos.

Por fim, o Capítulo 6 apresenta as considerações finais sobre os resultados obtidos, além de apontar direções futuras para a pesquisa apresentada nesta tese.

2 CONCEITOS FUNDAMENTAIS EM ALINHAMENTO DE ONTOLOGIAS

Neste capítulo são apresentados os principais conceitos que fundamentam a pesquisa proposta, em que se abordam os elementos, as etapas e os desafios inerentes ao processo de Integração de Dados e Alinhamento de Ontologias.

2.1 INTEGRAÇÃO DE DADOS

O processo de Integração de Dados tem por objetivo prover acesso unificado a dados residentes em diversas fontes heterogêneas. Ao longo de várias pesquisas realizadas nos últimos anos, constatou-se que alcançar esse objetivo tem sido notoriamente complexo, mesmo quando é considerado um pequeno número de fontes que proveem dados estruturados - cenário conhecido como integração de dados tradicional (DOAN; HALEVY; IVES, 2012).

São alguns dos desafios inerentes a este processo a ambiguidade semântica, a ambiguidade de representação de instâncias e a inconsistência de dados. Dong e Srivastava (2015) destacam três etapas fundamentais do processo de Integração de Dados tradicional, são elas: Alinhamento de Esquemas (*Schema Alignment*), Resolução de Entidades (*Record Linkage*) e Fusão de Dados (*Data Fusion*), ilustrados na Figura 5.

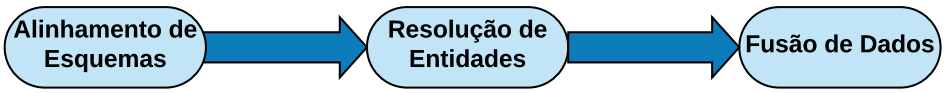


Figura 5 – Etapas do processo de Integração de Dados.

Fonte – Adaptado de Dong e Srivastava (2015)

Conforme a definição de Dong e Srivastava (2015), o processo de Integração de Dados é composto por três principais etapas. A primeira é o alinhamento de esquemas, que aborda o problema da ambiguidade semântica de elementos e visa compreender quais atributos representam o mesmo conceito de mundo-real. A segunda etapa é a fusão de dados, que aborda o desafio da ambiguidade na representação de instâncias, visando compreender quais registros representam a mesma entidade e quais não. Por fim, a terceira etapa é a fusão de dados, que aborda o desafio de avaliar a qualidade dos dados, no sentido de compreender quais instâncias de entidades integradas devem ser utilizadas, especialmente quando as fontes proveem valores conflitantes.

O escopo deste trabalho aborda como ponto central o desafio do alinhamento de esquemas. No entanto, nos últimos anos, devido à sua alta capacidade de abstração, as ontologias tornaram-se um modelo conceitual bastante utilizado para a representação de esquemas (SHVAIKO; SHVAIKO; EUZENAT, 2013). Por conta disso, nota-se que nos últimos anos (EUZENAT; SHVAIKO, 2013; OCHIENG; KYANDA, 2018c), o problema de alinhamento

de esquemas passou a ser tratado como alinhamento de ontologias, que é o termo adotado neste trabalho. Nas seções a seguir, são descritos em detalhes os conceitos fundamentais relacionados a essa área.

2.2 ONTOLOGIAS

Ontologias podem ser vistas como um conjunto de asserções para modelar um domínio particular. Além disso, podem representar diversos modelos conceituais ou de dados, tais como esquemas de bancos de dados, modelos UML ou XML. Dentre as diversas definições para ontologias na literatura, a mais referenciada é a proposta por Euzenat e Shvaiko (2013):

Ontologia é uma tupla $o = \langle C, I, R, T, V, \sqsubseteq . \perp, \in, = \rangle$, em que:

- C é um conjunto de classes;
- I é um conjunto de indivíduos;
- R é um conjunto de relacionamentos;
- T é um conjunto de tipos de dado;
- V é um conjunto de valores que pode estar relacionado aos demais elementos (sendo C, I, R, T, V disjuntos);
- $\sqsupseteq$ é um relacionamento entre $((C \times C) \cup (R \times R) \cup (T \times T))$ chamado especialização;
- $\perp$ é um relacionamento entre $((C \times C) \cup (R \times R) \cup (T \times T))$ chamado exclusão;
- $\in$ é um relacionamento entre $((I \times C) \cup (V \times T))$ chamado instanciação;
- $=$ é um relacionamento entre $I \times R \times (I \cup V)$ chamado atribuição.

Como para o alinhamento de ontologias geralmente não são considerados todos tipos de relacionamentos de uma ontologia (e.g., o relacionamento $\in$ não é considerado em sistemas que não utilizam instâncias), é comum que os trabalhos nesta área apresentem definições simplificadas. Um exemplo é o trabalho de Acampora et al. (2012), que propõe:

Uma ontologia é uma tripla $O = (C, P, I)$, em que:

- C é o conjunto de *classes*, i.e., os de conceitos que compõem um domínio de interesse;
- P é o conjunto de *propriedades*, i.e., os relacionamentos existentes entre os conceitos do domínio;
- I é um conjunto de *indivíduos*, i.e., os objetos de mundo real que representam as instâncias de um conceito.

Classes, propriedades e indivíduos são geralmente referenciados como elementos de ontologias. Propriedades que representam relacionamentos entre classes são denominadas *Propriedades de Objeto*. Propriedades que representam relacionamentos entre uma classe e algum valor literal (e.g., um número inteiro) são denominadas *Propriedades de Dado*.

Tendo em vista que o foco abordado nesta tese é alinhamento de ontologias, opta-se por seguir essa definição simplificada como referência ao longo do texto.

2.3 ALINHAMENTO DE ONTOLOGIAS

Alinhamento de Ontologias (*Ontology Matching*) pode ser definido como o processo de encontrar correspondências entre elementos de diferentes ontologias (DOAN; HALEVY; IVES, 2012). Esse conjunto de correspondências, denominado alinhamento, é a saída do processo e pode ser utilizado por outras áreas relacionadas, tais como: resolução de entidades (DONG; SRIVASTAVA, 2015), descoberta e composição de *Web Services* (SELLAMI; BOUCELMA, 2011), entre outros.

Nesta seção, são apresentadas algumas definições referentes a conceitos básicos que envolvem o processo de Alinhamento de Ontologias, seguindo como referência os trabalhos de Duchateau e Bellahsene (2015), Euzenat e Shvaiko (2013) e Guarino, Oberle e Staab (2009).

Processo de Alinhamento de Ontologias. O processo de Alinhamento de Ontologias pode ser definido como uma função f que gera um alinhamento A' para um par de ontologias O_1 e O_2. Alguns parâmetros complementares podem ser incluídos como entrada, tais como: (i) um alinhamento A, para ser completado ou estendido pelo processo, (ii) um conjunto de parâmetros p, que podem definir aspectos relacionados à execução do processo (e.g., pesos, limiares mínimos) e um conjunto de recursos r, que podem referenciar, por exemplo, uma base de conhecimento comum ou um dicionário de termos de um domínio específico, definido como:

$$A' = f(O_1, O_2, A, p, r) \tag{2.1}$$

Matcher. Um *matcher*, ou *Ontology Matcher* é um algoritmo ou uma função que recebe um conjunto de esquemas e fornece como saída um alinhamento (conjunto de correspondências) entre os elementos dos esquemas.

Medida de Similaridade. Uma medida de similaridade é definida como uma função que calcula o nível de similaridade existente entre uma tupla de elementos de ontologias $< e_1, e_2 >$, no qual o elemento $e_1 \in O_1$ e $e_2 \in E_2$. A função de similaridade é denotada por $sim(e_1, e_2)$ e indica o grau de semelhança entre os elementos. O resultado desta função (valor de similaridade) é um valor numérico no intervalo [0,1], no qual o valor 0 indica a ausência total de similaridade entre os elementos e o valor 1 representa a total de similaridade.

Correspondência. Uma correspondência C pode ser denotada como uma tupla $< e1, e2, k >$, na qual $e_1 \in O_1$, $e_2 \in O_2$ e $e_1 \equiv e_2$, sendo k o valor de similaridade existente entre os elementos que a compõem. Um conjunto de correspondências pode ser obtido a partir do resultado da execução de um *matcher* ou por indicação manual de usuários especialistas.

Alinhamento. Um alinhamento A consiste em um conjunto de correspondências identificadas entre dois esquemas. Desta forma, pode-se definir que $A = C\{O_1, O_2\}$, onde $C\{O_1, O_2\}$ representa o conjunto contendo as correspondências identificadas entre os elementos das ontologias O_1 e O_2.

2.3.1 Etapas do Processo de Alinhamento de Ontologias

Seguindo a definição genérica introduzida por Duchateau (2009) e referenciada por trabalhos posteriores (e.g., (XUE; CHEN, 2018), (OCHIENG; KYANDA, 2018c)), o processo de *Ontology Matching* consiste nas seguintes etapas: (i) preparação: processamento prévio de recursos recebidos e configurações iniciais; (ii) execução: descoberta de correspondências e (iii) refinamento: validação e consolidação das correspondências descobertas. Abaixo descreve-se o escopo de cada etapa.

Etapa de Preparação (*Pre-match*). Inclui o carregamento de recursos externos que serão usados no processo, como também a calibração de parâmetros e a realização de algum pré-processamento necessário, conforme detalha-se a seguir.

- Carregamento de recursos que serão utilizados no processo: importação dos esquemas, alinhamentos (fornecidos pelo usuário como referência) e recursos externos (e.g., dicionários, ontologias de domínio).
- Calibração: definição de valores para os parâmetros de configuração disponibilizados pelos *matchers* (e.g., pesos ou limiares). Esses valores podem ser definidos manualmente ou automaticamente, sendo a calibração automática tema de várias pesquisas (PEUKERT; EBERIUS; RAHM, 2012; GAL; SAGI, 2010; RAHM; PEUKERT; EBERIUS, 2011).
- Redução de ontologias: geração de segmentos a partir das ontologias originais, reduzindo o total de elementos a comparar no processo (e.g., (DRUMM et al., 2007)).
- Pré-processamento: processamento prévio de recursos, tais como a conversão das ontologias de entrada para uma estrutura de representação interna ou a geração de um conjunto de treinamento (útil para abordagens baseadas em Aprendizado de Máquina (DOAN et al., 2002)).

Etapa de Execução (*Matching*). Aplicação de medidas de similaridade aos elementos das ontologias visando estabelecer um conjunto de correspondências. As atividades

realizadas nesta etapa vão depender da estratégia utilizada. Em algumas abordagens, o usuário pode ser requisitado para interagir durante esta etapa, realizando desambiguações ou respondendo perguntas (e.g., (RODRIGUES et al., 2015)).

Etapa de Refinamento (*Post-matching*). Avaliação das correspondências geradas por parte do usuário, validando-as ou invalidando-as a partir da interação com uma interface gráfica (e.g., COMA++ (AUM"ULLER et al., 2005)) ou API. O resultado dessa avaliação pode ser armazenado e reutilizado como entrada para alinhamentos posteriores.

2.3.2 Heterogeneidade de Elementos de Ontologias

Um dos principais desafios durante a geração de alinhamentos é a necessidade de reduzir heterogeneidades presentes em diferentes elementos. Essas heterogeneidades podem se apresentar em múltiplas formas. Por exemplo, as ontologias podem referenciar um mesmo conceito utilizando elementos com: (i) diferentes nomes (e.g., Endereco.rua ≡ Endereco.logradouro), (ii) quantidades de atributos (e.g., Pessoa.nome_completo ≡ Pessoa.nome+Pessoa.Sobrenome) e (iii) representações de estrutura (entidade, propriedade) (e.g., Pessoa.endereco.rua ≡ Pessoa.endereco_rua). A principal causa dessas heterogeneidades é o fato de que as ontologias são criadas a partir do ponto de vista de quem as modelam, aplicando estruturas e terminologias de sua preferência.

Diversas classificações para essas heterogeneidades foram propostas na literatura (BENERECETTI; BOUQUET; GHIDINI, 2000; WACHE et al., 2001; KLEIN, 2001; EUZENAT, 2001; CORCHO, 2005; HAMEED; PREECE; SLEEMAN, 2003; BOUQUET et al., 2004), sendo a maioria focada em um tipo modelo ou num domínio específico. Neste trabalho, consideram-se três dos tipos mais comuns de heterogeneidades destacados por Euzenat e Shvaiko (2013):

Heterogeneidades Sintáticas. Ocorre quando duas ontologias não utilizam o mesmo modelo ou linguagem de representação. Normalmente, são tratadas no nível teórico e em alguns casos é possível oferecer tradutores para resolvê-las sem que haja perda de significado.

Heterogeneidades Terminológicas. Ocorre devido a variações na nomenclatura utilizada para descrever as entidades ou propriedades de uma ontologia, tais como o uso de derivações (e.g., Telefone e Fone), abreviações (End. e Endereço), sinônimos (Autor e Criador) ou termos em diferentes idiomas (e.g., Título e Title).

Heterogeneidades Conceituais ou Semânticas. Relaciona-se às variações no significado dos termos utilizados para representar um determinado conceito. Pode ocorrer nos níveis de (i) cobertura: relacionado à variação do significado de acordo com localização geográfica; (ii) granularidade: relacionado à utilização de diferentes estruturas

ou níveis de abstração e (iii) perspectiva: relacionado à mudança de significado de acordo com o contexto em que o termo é aplicado.

2.3.3 Alinhamento de Ontologias em Larga-Escala

Muitos dos problemas encontrados na construção dos sistemas de integração de dados no contexto de grandes volumes de dados, como na *Web*, são similares aos encontrados em sistemas de integração de bancos de dados tradicionais. Entretanto, existem algumas diferenças que precisam ser consideradas quando fontes são integradas em larga-escala, tais como: (i) o número de fontes de dados pode ser grande, o que dificulta os processos de integração de ontologias e resolução de conflitos; (ii) as fontes de dados são dinâmicas e assim a adição ou remoção de fontes de dados deve ser feita de maneira a minimizar o impacto na visão integrada; (iii) as fontes de dados são heterogêneas, podendo contemplar desde sistemas de gerenciamento de bancos de dados até dados provenientes de sensores e (iv) as fontes de dados podem ser não estruturadas ou semi-estruturadas e, em sua maioria, não fornecem informações suficientes para a integração das ontologias.

Em pesquisas mais recentes, tais cenários envolvendo ambientes de larga-escala serviram como principal motivação para a busca de uma maior automatização do processo de alinhamento de ontologias, que pode ser caracterizada desta forma quando: (i) são fornecidas duas ontologias grandes (e.g., contendo milhares de elementos) e (ii) é fornecido um largo conjunto de ontologias (contendo centenas de ontologias). Em ambos os casos, a intervenção manual do usuário para avaliar correspondências pode não ser viável, dada a possibilidade de uma grande quantidade de elementos a avaliar.

Neste cenário, somam-se novos desafios aos já conhecidos. Um dos principais problemas é o alto tempo de execução que a atividade pode consumir, dada a quantidade de comparações necessárias. Para lidar com isso, estratégias de paralelização podem ajudar a obter resultados em menos tempo (ARAÚJO et al., 2015), visto que a comparação entre cada par de ontologias pode ocorrer de maneira independente. Porém, aplicar este método simplesmente através de uma abordagem tradicional pode não ser suficiente, tendo como consequência um maior consumo de recursos, sem gerar ganhos significantes no que diz respeito ao tempo de execução. Uma alternativa seria combinar a paralelização com outras estratégias, como a segmentação de ontologias.

Ressalta-se que os desafios inerentes ao ambiente de larga-escala vão além do tempo de execução (DONG; SRIVASTAVA, 2015). No Quadro 1, apresentam-se alguns exemplos do impacto do fator de larga-escala em comparação com o processo tradicional, com base nos trabalhos de Do e Rahm (2007a), Mork et al. (2009), Rahm (2011), Ivanova, Lambrix e Åberg (2015).

Um dos itens citados como estratégia para execução do processo é a combinação de diferentes *matchers*. Neste caso, visa-se aproveitar os pontos positivos de cada abordagem, como forma de obter um maior número de resultados corretos. A partir disto, surge uma

Quadro 1 – Exemplos do impacto do fator de larga-escala no processo de alinhamento de ontologias

Aspecto	**Processo em Larga-escala**	**Processo Tradicional**
Granularidade de execução	Demanda várias iterações (divisão em sub-atividades)	Em alguns casos, pode ser resolvida numa única iteração.
Configuração/Calibração (tuning)	Indireta/automática (através da correlação de características do ambiente do solicitante)	Direta/manual (através do fornecimento de parâmetros)
Realização da etapa de preparação	Necessária	Opcional
Representação das correspondências	Redes de relacionamentos	Matriz de relacionamentos
Utilização de um único limiar mínimo	Impreciso (demanda aplicação de outras estratégias, como limiares múltiplos (e.g., Double thresholding)	Possível
Redução das correspondências apresentadas pelo matcher	Obrigatória	Desejável
Intervenções manuais	Envolve vários usuários especialistas, podendo demandar tratamento de conflitos	Envolve normalmente um usuário especialista
Redução de intervenções manuais	Necessário	Desejado
Estratégias para redução do Tempo de Execução	Necessidade de combinar várias estratégias de otimização como, por exemplo, paralelização e redução do espaço de busca, para se chegar a um tempo aceitável.	Geralmente, a aplicação de uma estratégia de otimização é suficiente.
Estratégia para execução do processo	Combinação de vários matchers	O uso de um matcher pode ser suficiente.

outra dificuldade, que é a grande quantidade de *matchers* disponíveis. Nesta perspectiva, é necessário estabelecer estratégias que proporcionem a seleção dos mais adequados a um cenário, e, posteriormente, indicar como combinar da melhor maneira os resultados obtidos por cada um. Como forma de indicar evidências para auxiliar nesta seleção, há algumas classificações para os *matchers* existentes na literatura, conforme é apresentado a seguir.

2.4 CLASSIFICAÇÃO DOS *MATCHERS*

Nesta seção são apresentadas algumas classificações para os *matchers* existentes. Seguindo o que foi definido por Euzenat e Shvaiko (2013), conforme mostra a Figura 6, os *matchers* podem ser classificados em duas dimensões gerais: de acordo com a granularidade (e.g., opera no nível de elemento ou no nível estrutural) ou com base nas características da entrada recebida.

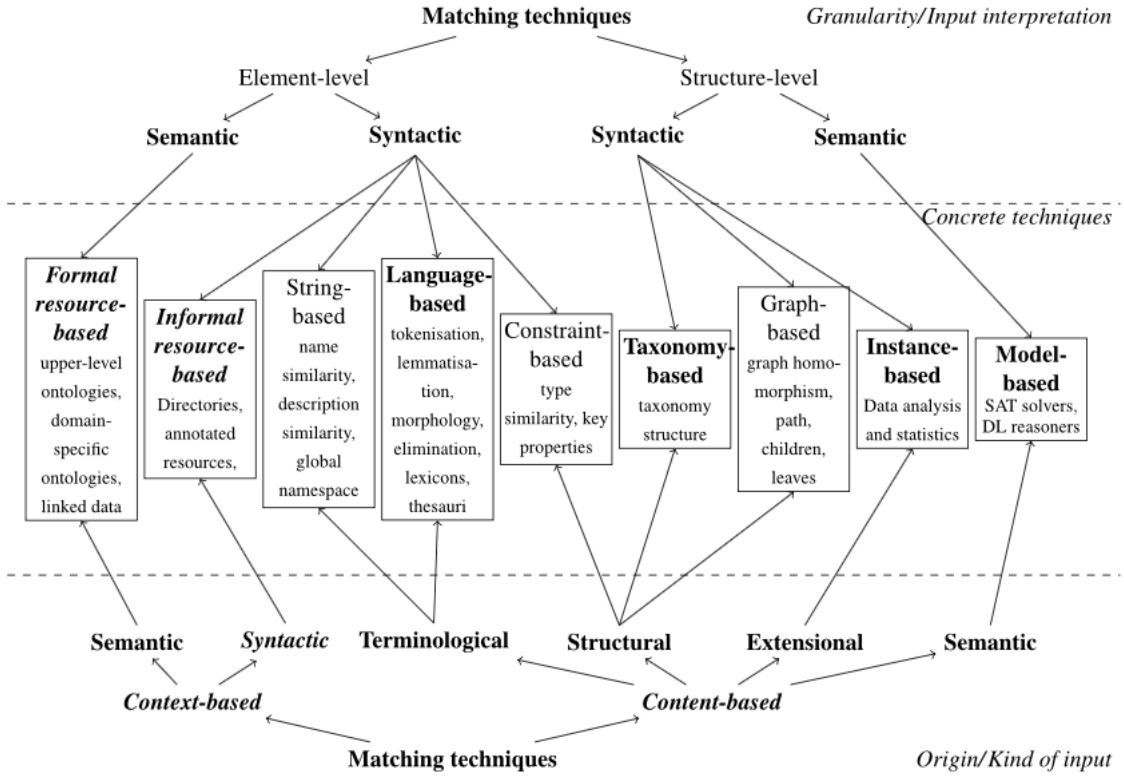


Figura 6 – Classificação dos *matchers*

Fonte – (EUZENAT; SHVAIKO, 2013)

Com base na granularidade, os *matchers* dividem-se em dois níveis: de elemento e estrutural. O nível de elemento contempla os *matchers* baseados em (i) comparação de *strings*; (ii) linguagens; (iii) restrições; (iv) recursos externos informais e (v) recursos externos formais. Os de nível estrutural contemplam os *matchers* baseados em (i) grafos; (ii) taxonomias; (iii) modelos e (iv) instâncias.

Já os com base nas características da entrada recebida, dividem-se em dois tipos: (i) baseados no contexto (e.g., o domínio ao qual a ontologia pertence) e (ii) no conteúdo de entrada (e.g., o conteúdo da ontologia em si). Os baseados no contexto contemplam os *matchers* baseados em: (i) recursos formais e (ii) recursos informais. Os que consideram o conteúdo da entrada contemplam os demais *matchers*, estabelecendo quatro categorias: terminológicos, estruturais, extensionais (de instâncias) e semânticos.

Baseados em *strings* (*String-based*). Baseiam-se na comparação dos nomes dos elementos. As *strings* mais similares são consideradas como tendo maior probabilidade de representar o mesmo termo. Alguns exemplos de técnicas utilizadas nesta perspectiva são a identificação de sub-palavras (prefixo, sufixo) ou as funções de similaridade *edit distance* e *n-gram*.

Baseados em linguagem (*Language-based*). Adotam técnicas de linguagem natural

para analisar morfologicamente os termos que descrevem os elementos. Neste caso, as similaridades são encontradas com base nas relações linguísticas que possam existir entre os termos comparados.

Baseados em restrições (*Constraint-based*). Exploram restrições internas que podem ser aplicadas durante a definição das entidades, tais como, tipos de dado, cardinalidade (ou multiplicidade) de atributos e chaves.

Baseados em recursos informais (*Informal resource-based*). Identificam similaridades através do acesso a recursos informais externos à ontologia. Neste caso, são consideradas fontes léxicas simples, como por exemplo, dicionários e tesauros, no intuito de explorar as relações linguísticas entre termos (e.g., sinônimos, antônimos).

Baseados em recursos formais (*Formal resource-based*). Utiliza referências externas formais, como as ontologias de domínio, para auxiliar na descoberta de novas correspondências.

Baseados em grafos (*Graph-based*). Identificam similaridades através da análise da posição dos elementos numa estrutura de grafo. Neste caso, cada elemento da ontologia é interpretado como um nó e os relacionamentos como arestas. A premissa desta técnica é que, se dois nós são similares, os vizinhos também devem apresentar similaridades.

Baseados em taxonomias (*Taxonomy-based*). Também são baseados em grafos, mas consideram apenas as relações de especialização e generalização. A premissa desta técnica é que, se os elementos estão envolvidos neste tipo de relação, eles são subconjuntos um do outro, logo, representam conceitos relacionados.

Baseados em modelos (*Model-based*). Realizam a interpretação semântica das ontologias fornecidos, utilizando como base um modelo abstrato. A intuição desta abordagem é que, se duas entidades representam o mesmo conceito, elas devem compartilhar a mesma interpretação. Exemplos de aplicação deste tipo de técnica é a adoção de modelos com base em lógica descritiva.

Baseados em instâncias (*Instance-based*). Compara conjuntos de instâncias das entidades para decidir se os elementos são correspondentes ou não. Podem considerar regras de inferência simples para analisar um valor ou utilizar técnicas mais elaboradas de análise estatística.

2.4.1 Critérios para avaliação de qualidade de *matchers*

O processo de avaliação de qualidade de um *matcher* depende da definição de quais critérios são mais relevantes ao usuário. Cada critério pode estar associado a uma ou mais métricas,

que irão fornecer os valores que serão considerados para estabelecê-los. Essas métricas podem ser objetivas (calculadas automaticamente) ou subjetivas (quando o cálculo para um cenário específico exige intervenção humana).

Estão no escopo do processo de avaliação de *matchers* definido neste trabalho apenas critérios objetivos. Isto é, critérios que dispensam a intervenção do usuário. Isso é justificado pela necessidade de automatizar o máximo possível. Sendo assim, critérios que dependem de uma resposta subjetiva do usuário, como satisfação e nível de esforço (em relação a intervenções manuais) não são abordados. Em resumo, os critérios de qualidade considerados neste trabalho são: (i) conformidade - calculado com base em métricas conhecidas, como precisão e cobertura; e (ii) desempenho - calculado com base em métricas que consideram o consumo de memória, velocidade e uso de rede. Esses critérios foram estabelecidos com base no que foi listado por Euzenat e Shvaiko (2013).

2.4.1.1 Métricas para Avaliação de Conformidade

Nesta seção, são apresentadas as métricas utilizadas para avaliar a qualidade dos *matchers*, baseada na proporção de correspondências geradas corretamente. As métricas consideradas nesta categoria utilizam como base os valores definidos na Tabela 2, que classificam a conformidade dos alinhamentos, i.e., a proporção de alinhamentos corretamente gerados pelos *matchers*. Tal classificação pode ser realizada a partir da intervenção de um usuário especialista ou pelo fornecimento de alinhamentos de referência (*gold standards*).

Quadro 2 – Elementos base para cálculo de métricas de avaliação de conformidade

	Corresp. corretas	**Corresp. incorretas**
Correspondências avaliadas como correta pelo *Matcher*	VP (Verdadeiro Positivo)	FP (Falso Positivo)
Correspondências avaliadas como incorreta pelo *Matcher*	FN (Falso Negativo)	VP (Verdadeiro Negativo)

Precisão (*Precision*). Calcula a proporção de resultados corretos descobertos pelo *matcher* entre todos os resultados obtidos. Usando as notações do Quadro 2, a precisão é dada pela Equação (2.2). A obtenção do valor máximo de precisão ($Precisão = 1$) significa que todos os mapeamentos descobertos pelo *matcher* estão corretos.

$$Precisão = \frac{VP}{VP + FP} \tag{2.2}$$

Cobertura/Revocação (*Recall*). Mede a proporção entre as correspondências indicadas como corretas pelo *matcher* e o total de correspondências corretas após avaliação. A cobertura é calculada pela Equação (2.3). A obtenção do valor máximo de cobertura ($Cobertura = 1$), significa que todas as correspondências existentes foram descobertas pelo *matcher*.

$$Cobertura = \frac{VP}{VP + FN} \tag{2.3}$$

Medida-F (*F-Measure*). Harmoniza as métricas de precisão e cobertura. Dado uma constante β com valor entre 0 e 1, a Medida-F é calculada conforme a Equação (2.4).

$$Medida - F(\beta) = \frac{(\beta^2 + 1)\ x\ Precisão\ x\ Cobertura}{(\beta^2\ x\ Precisão\ x\ Cobertura)} \tag{2.4}$$

Há ainda outras métricas para a avaliar os *matchers* quanto à sua conformidade (e.g., *fallout*, *overall*). No entanto, como a ideia deste trabalho é possibilitar que o usuário indique as métricas de sua preferência, a partir da definição de requisitos de qualidade, são consideradas apenas as métricas mais adotadas pelas iniciativas que fazem avaliações de *matchers* (e.g., OAEI), que é o caso das medidas de precisão, cobertura e Medida-F.

2.4.1.2 Métricas para Avaliação de Desempenho

As métricas para avaliação de desempenho consideram os recursos consumidos durante uma atividade de *matching*. Diferentemente das métricas de conformidade, as de desempenho dependem do ambiente de processamento. Desta forma, torna-se bastante desafiador obter avaliações confiáveis, visto que, para tal, seria necessária a comparação de diferentes *matchers*, sob as mesmas condições.

Tempo de Execução. A medida de tempo de execução leva em consideração quanto tempo um *matcher* gasta em média para concluir o seu processamento. Para obter uma maior confiabilidade na comparação entre diferentes *matchers*, a sua medição deve ser feita exatamente sob as mesmas condições (e.g., mesmo processador). Se a interação do usuário é requerida em algum momento pelo *matcher*, deve-se garantir que será computado para esta medida apenas o tempo de processamento interno do algoritmo. Nesta pesquisa, a métrica de velocidade é utilizada de maneira relativa, sendo fornecida da seguinte forma: a partir da obtenção do tempo de execução máximo e mínimo considerando todos os *matchers*, calcula-se a proporção do tempo médio de cada *matcher* com uma escala de valores de 1 a 3, de maneira a classificá-los como tendo velocidade baixa, média ou alta.

Uso de Rede. Alguns *matchers* utilizam a conectividade com a rede para obter os seus resultados. O consumo de rede pode ser medido em termos da vazão de banda passante utilizada por um *matcher* durante o processo. Isto é válido principalmente para *matchers* que utilizam consultas a recursos externos, como o acesso a dicionários ou a ontologias publicadas na *Web*.

Uso de Memória. O total de memória utilizado para realizar a geração de alinhamentos é outra métrica de desempenho. Para obter valores para esta métrica, há a dependência de componentes de baixo nível do sistema operacional.

2.5 MÉTRICAS PARA CLASSIFICAÇÃO DE ONTOLOGIAS

Nos últimos anos, foram propostas na literatura várias métricas com o intuito de quantificar as características de uma ontologia com respeito aos aspectos linguísticos, estruturais e de

instância. Especificamente, no contexto da geração de alinhamentos, essas métricas podem ser utilizadas para identificar a predominância de determinados aspectos nas ontologias, de maneira a considerar a escolha de *matchers* que os privilegie.

Neste trabalho, como parte do processo de recomendação de *matchers*, realiza-se a classificação dos alinhamentos com base nas características das ontologias comparadas, no intuito de estabelecer algum padrão que indique a compatibilidade com um determinado tipo de *matcher*. Para tal, utiliza-se o estado da arte das métricas para classificação de ontologias, introduzidas por Cruz et al. (2012) e estendidas por Sathiya, Geetha e Sugumaran (2017), conforme é descrito a seguir.

Antes de calcular as métricas para classificação de ontologias, é necessário identificar um conjunto de atributos que caracterizam a ontologia, listados abaixo: Esses atributos são utilizados como parâmetros para as métricas, detalhadas em:

- **Classes.** Quantidade de classes que uma ontologia possui;
- **Propriedades de Dado.** Quantidade de relacionamentos entre classes e tipos literais;
- **Propriedades de Objeto.** Quantidade de relacionamentos entre duas classes;
- **Número de instâncias.** Quantidade de instâncias presentes na ontologia;
- **Classes com Instâncias.** Quantidade de classes que possuem ao menos uma instância;
- **Rótulos nulos.** Número de elementos sem valor atribuído para os rótulos;
- **Comentários nulos.** Número de elementos sem valor atribuído para comentários;
- **Total de elementos.** Número total de elementos, incluindo classes e propriedades;
- **Total de subclasses.** Número total de relacionamentos de especialização entre classes;
- **Outros Relacionamentos.** Número total de outros relacionamentos além de especialização.
- **Profundidade total.** Profundidade total considerando todos os elementos da ontologia.

Uma vez identificados os atributos da ontologia a ser classificada, calcula-se os valores para cada métrica. Os valores obtidos para essas métricas são utilizados neste trabalho para avaliar um conjunto de ontologias quanto à sua riqueza ou fraqueza em relação a um determinado tipo de *matcher* (e.g., terminológico, estrutural, semântico, de instância). Essas métricas são definidas e divididas de acordo com a respectiva categoria, seguindo

a classificação proposta por Cruz et al. (2012) (Quadro 3). Em todas as definições de métricas, c_i representa o conceito número i pertencente a um conjunto de conceitos C de uma determinada ontologia.

Quadro 3 – Classificação das métricas do perfil de ontologias em categorias (CRUZ et al., 2012)

Métrica	**Sintática**	**Léxica**	**Estrutural**	**Instância**
Unidade de Rótulos	X			
Comentários e Rótulos Nulos	X			
Riqueza de Relacionamentos			X	
Riqueza de Atributos			X	
Riqueza de Herança			X	
Profundidade Média			X	
Riqueza de Atributos de Classe			X	
Riqueza de Estrutura Externa de Conceitos			X	
Cobertura do Wordnet		X		
Riqueza de Classe				X
População Média				X

2.5.1 Métricas de características sintáticas

As métricas de características sintáticas levam em consideração a presença de elementos terminológicos nos rótulos e comentários de ontologias. A partir dessa identificação, pode-se utilizar técnicas que levem em consideração também a comparação desses atributos, além do próprio nome dos elementos. Nesta categoria são definidas as métricas de Unicidade de Rótulos (UR), que identifica a porcentagem de elementos com rótulos diferentes do seu nome e a métrica de porcentagem de rótulos e comentários nulos, conforme é apresentado:

Unicidade de Rótulos. Um elemento de ontologia é descrito pelo nome (*local name*), que é sua principal identificação, e por um rótulo (*label*), que pode ser utilizado para melhorar a sua representação semântica, indicando alguma informação adicional. A métrica de Unicidade de Rótulos (UR) captura o número de termos pelos quais o nome e o rótulo de um elemento diferem, de maneira a determinar se podem ser encontradas informações adicionais. Define-se Unicidade de Rótulos (UR) como a proporção de termos que tiveram um rótulo que difere do seu nome (*diff*) em relação ao número total de termos ($|T|$).

$$UR = \frac{\mid diff \mid}{\mid T \mid}$$

Rótulos e comentários nulos. A métrica de Rótulos e Comentários Nulos (RCN) indica a porcentagem de termos que não possuem comentários ou rótulos (*labels*) m relação

ao número total de termos ($|T|$), nomeadas respectivamente por $N_{comment}$ e N_{label}.

$$RCN = N_{i \in \{label,comment\}} = \frac{\mid N_i \mid}{\mid T \mid}$$

2.5.2 Métricas de características estruturais

As métricas estruturais quantificam as características internas e externas de ontologias. As características externas (que envolve outros elementos), tais como a profundida de conceitos e relacionamentos, são quantificados por métricas como a Profundidade média, Riqueza de Relacionamentos, Riqueza de Herança e Riqueza Estrutural de Conceitos Externos. Já as características internas, como os atributos de um elemento, são quantificados pelas métricas de Riqueza de Atributos e Riqueza de Atributos de Classes. Essas métricas são definidas a seguir.

Riqueza de Relacionamentos. A métrica de Riqueza de Relacionamentos (RR) (CRUZ et al., 2012) de uma ontologia é definida pela porcentagem de relações no nível da ontologia (propriedades que referenciam uma entidade) que são diferentes das relações de uma subclasse (presentes na entidade especializada). Na Equação 2.5, P é o conjunto de tais relações e SC é o conjunto das relações de subclasses. Além de analisar aspectos terminológicos, esta métrica também é considerada uma métrica de elementos estruturais (2.5.2), por considerar as relações de especialização/realização.

$$RR = \frac{\mid P \mid}{\mid SC \mid + \mid P \mid} \tag{2.5}$$

Riqueza de Atributos. A Riqueza de Atributos (RA) (CRUZ et al., 2012) é definida como a média do número de propriedades de que apontam para um tipo (tipos complexos), por classe (entidade), computada pela proporção de atributos (att) por total de classes (C).

$$RA = \frac{\mid att \mid}{\mid C \mid}$$

Riqueza de Atributos de Classes. A Riqueza de Atributos de Classes (RAC) (SATHIYA; GEETHA; SUGUMARAN, 2017) é definida pela porcentagem do número de conceitos contendo atributos, em que os atributos são representados por propriedades de dado na ontologia. Na Equação 2.6, $P(c_i)$ é uma função booleana que retorna 1 se o conceito c_i contém um atributo, do contrário o valor 0 é retornado.

$$RAC = \frac{\sum_{c_i \in C} P(c_i)}{\mid C \mid} \tag{2.6}$$

Riqueza Estrutural de Conceitos Externos. A Riqueza Estrutural de Conceitos Externos (RECE) (SATHIYA; GEETHA; SUGUMARAN, 2017) é definida pela porcentagem de conceitos que contenham relacionamentos representados por propriedades de

objeto. Na Equação 2.7, $R(c_i)$ é uma função booleana que retorna 1 se o conceito c_i contém ao menos um desses relacionamentos associado a ele, do contrário o valor 0 é retornado. A soma dos valores dessa função para todos os conceitos irá fornecer o número de classes associadas a relacionamentos com outras classes, que é a porcentagem computada pela RECE.

$$RECE = \frac{\sum_{c_i \in C} R(c_i)}{\mid C \mid} \tag{2.7}$$

Riqueza de Herança. A Riqueza de Herança (RH) (CRUZ et al., 2012) descreve a estrutura de uma ontologia, sendo definida a partir da média do número de subclasses por classe. O número de subclasses para uma classe c_i é definido como $\mid H^c(c_l, c_i) \mid$, onde c_l é uma subclasse de c_i. Em resumo, isso significa que, uma ontologia com baixo RH tem poucos filhos por classe, mas muitos níveis de herança; enquanto que uma ontologia com alto RH tem muitos níveis hierárquicos, mas poucos filhos por classe.

$$RH = \frac{\sum c_i \in C \mid H^C(c_l, c_i)}{\mid C \mid} \tag{2.8}$$

Profundidade Média. Esta métrica descreve a profundidade média (PM) das classes (entidades) de uma ontologia, definidos como a média da profundidade sobre a obtida para todas as classes $c_i \in C$ ($PM'(c_i)$).

$$PM = \frac{\sum_{c_i \in C^{PM'(c_i)}}}{\mid C \mid}$$

2.5.3 Métricas de elementos léxicos

As métricas de elementos léxicos quantificam o relacionamento dos elementos das ontologias com termos existentes em um banco de dados léxico ou dicionário idiomático. O Wordnet (KILGARRIFF; FELLBAUM, 2000) é o serviço mais popular utilizado para este fim. Por esse motivo, a métrica léxica definida por Cruz et al. (2012) é a Cobertura do Wordnet, que calcula a porcentagem de termos que descrevem elementos das ontologias que se encontram no Wordnet, conforme define-se a seguir.

Cobertura do WordNet. A métrica de Cobertura do Wordnet (CW) avalia, considerando cada par de termos, se nenhum, um, ou ambos podem ser encontrados no Wordnet (KILGARRIFF; FELLBAUM, 2000). Neste caso, são computadas duas métricas, considerando o nome que identifica os elementos (id) e os rótulos presentes no WordNet ($covered$).

$$CW_{i \in \{label,id\}} = \frac{\mid covered_i \mid}{\mid C \mid}$$

2.5.4 Métricas de instâncias

As métricas de instâncias utilizam os dados armazenados (instâncias) para quantificar as características das ontologias. Para tal, Cruz et al. (2012) propõe duas métricas, uma baseada na proporção de instâncias por classe (Riqueza de Classe) e outra que indica a quantidade média de instâncias em todas as classes (População Média), conforme detalha-se a seguir.

Riqueza de Classe. A Riqueza de Classe (RC) é definida pela proporção do número de instâncias ($\mid C_i \mid$) para o total do número de classes definidas na ontologia ($|C|$).

$$RC = \frac{\mid C_i \mid}{\mid C \mid}$$

População Média. A População Média (PM) provê uma indicação da distribuição média de instâncias entre todas as classes. Na Equação 2.9, o valor de PM é obtido através do cálculo do número total de instâncias $\mid I \mid$ dividido pelo número de classes C.

$$PM = \frac{\mid I \mid}{\mid C \mid} \tag{2.9}$$

Todas essas métricas são representadas em números reais. O valor obtido para as métricas sempre variam no intervalo de 0 a 1. A obtenção de valores altos (próximos a 1) para as métrica indicam a riqueza das características correspondentes, enquanto que obtenção de valores baixos (próximos a 0) indicam a deficiência de tais características na ontologia. As métricas listadas acima formam um vetor numérico com 11 valores que quantificam as características da ontologia em termos de riqueza léxica, sintática, estrutural e de instância. Para cada processo de alinhamento de ontologias, dois vetores de métricas são computados, um para cada ontologia fornecida como entrada. Esses dois vetores são combinados em um vetor único através de uma função de agregação, conforme descrito na seção que segue.

2.5.5 Combinação de métricas de ontologias

Para que seja possível relacionar um *matcher* a um par de ontologias fornecidas como entrada para a geração de alinhamentos, torna-se necessário combinar as métricas calculadas para cada ontologia participante (fonte e alvo). Cruz et al. (2012) propuseram a métrica $FS - A$ para a realização dessa combinação, tomando como base duas funções: média aritmética (A) e uma nova função definida como *Feature Similarity* (FS). A função A calcula a média simples para os valores de uma métrica particular obtidos para cada ontologia e pode ser usada para detectar o quanto uma determinada característica se faz presente. Por exemplo, a média da métrica de Cobertura do Wordnet (CW) será um indicador de quão adequado será um *matcher* léxico para a geração de alinhamentos. Por

outro lado, a função FS avalia o quão uma característica é similar nas duas ontologias. Isto é, quando a CW média é alta, FS indica se as duas ontologias obtiveram valores similares ou se os valores são altos em uma e baixo na outra. Desta forma, pode-se dispensar o *matcher* léxico mesmo quando a média for alta, caso o valor seja muito baixo em uma ontologia.

Dada a obtenção de duas métricas de *Ontology Profiling*, sendo um para cada ontologia a ser alinhada, a função FS (Equação 2.5.5) é definida pela razão entre a métrica de menor valor (m_L) e a métrica de maior valor (m_H), dividido pelo logaritmo das suas diferenças. Quando os dois valores das métricas são iguais, o valor de FS é igual a um. Quanto mais os valores forem diferentes, mais o valor de FS é próximo a zero.

$$FS_M = \frac{M_L}{M_H[log(M_H - M_L + 1) + 1]}$$

Suponha que é calculada a métrica de Cobertura do Wordnet (CW) para o par de ontologias (O_1, O_2) e que os valores obtidos foram respectivamente $0,5$ e $0,8$. Logo, o cálculo da média (A_{CW}) e da *Feature Similarity* (FS_{CW}) se daria da seguinte maneira:

$$A_{CW} = \frac{0,5 + 0,8}{2} = 0,65 \qquad FS_{CW} = \frac{0,5}{0,8[log(0,8 - 0,5 + 1) + 1]} = 0,49$$

Segundo os experimentos realizados por Cruz et al. (2012), a concatenação das funções FS e A em uma função FS-A apresentou maior expressividade em detrimento ao uso de maneira isolada. Desta forma, torna-se possível levar em consideração aspectos particulares presentes nas duas ontologias, como também o quanto elas compartilham esses aspectos. Logo, considerando o exemplo anterior, o cálculo da função FS-A se daria desta forma:

$$FS\text{-}A_{CW} = A_{CW} + FS_{CW} = 0,65 + 0,49 = 1,14$$

Essas métricas podem ser aplicadas tanto em ontologias inteiras, como em segmentos. A vantagem de utilizar segmentos é possibilitar o cálculo do perfil em versões reduzidas de ontologias que contenham elementos que são primordiais para o usuário, o que possibilita uma maior eficiência em relação à definição do melhor *matcher* para processá-los, conforme é descrito nos Capítulos 4 e 5.

2.6 CONSIDERAÇÕES FINAIS

Neste capítulo foram abordados conceitos fundamentais a esta tese. Inicialmente, apresentou-se uma visão geral sobre o conceito de ontologias no contexto de integração de dados. Em seguida, foi abordado o problema de alinhamento de ontologias, de maneira a descrever as principais etapas consideradas por processos propostos na literatura. Destacam-se também as heterogeneidades que podem ocorrer entre elementos das ontologias, categorizando-as em sintáticas, terminológicas e conceituais/semânticas. Além disso, também são apresentadas

características específicas do problema de alinhamento de ontologias em larga-escala, em que realiza-se uma comparação com o problema tradicional.

Foram conceituadas também as principais métricas para classificação de *matchers* quanto à sua qualidade, levando em consideração os critérios para avaliação de conformidade e de desempenho. Essas métricas estão diretamente relacionadas à forma como o framework proposto neste trabalho aborda o processo de avaliação de matchers, através da concepção de um perfil de qualidade.

Por fim, levantou-se o estado da arte em relação às métricas para classificação de ontologias, em que descrevem-se as principais características que levam em consideração a predominância de aspectos sintáticos, estruturais, léxicos e de instância nos elementos de ontologias. Introduz-se ainda as estratégias existentes para combinação dessas métricas, no intuito possibilitar a classificação de pares de ontologias.

No próximo capítulo, apresenta-se a solução proposta para redução do escopo de ontologias durante a geração de alinhamentos, que baseia-se na extração de segmentos de ontologias a partir de um conjunto de requisitos fornecidos pelo usuário. No capítulo subsequente, descreve-se o framework proposto para a avaliação e recomendação de *matchers*, em que utilizam-se as métricas para classificação de ontologias para determinar as características de pares de segmentos. A avaliação da qualidade dos *matchers* é feita utilizando as métricas para classificação descritas neste capítulo, tomando como base a geração de alinhamentos prévios entre ontologias com características similares às das fornecidas como entrada.

3 TRABALHOS RELACIONADOS

Este capítulo tem como objetivo apresentar e discutir alguns trabalhos existentes na literatura relacionados com a proposta desta pesquisa, sendo divididos em duas categorias: a primeira elenca as abordagens que utilizam a estratégia de segmentação de ontologias para a redução do espaço de busca durante a geração de alinhamentos; a segunda apresenta técnicas e ferramentas existentes para a recomendação e seleção de *matchers*.

A Figura 7 mostra a linha cronológica dos trabalhos apresentados neste capítulo. No quadrante superior, estão os que utilizam a segmentação de ontologias e no inferior os que abordam a recomendação e seleção de *matchers*.

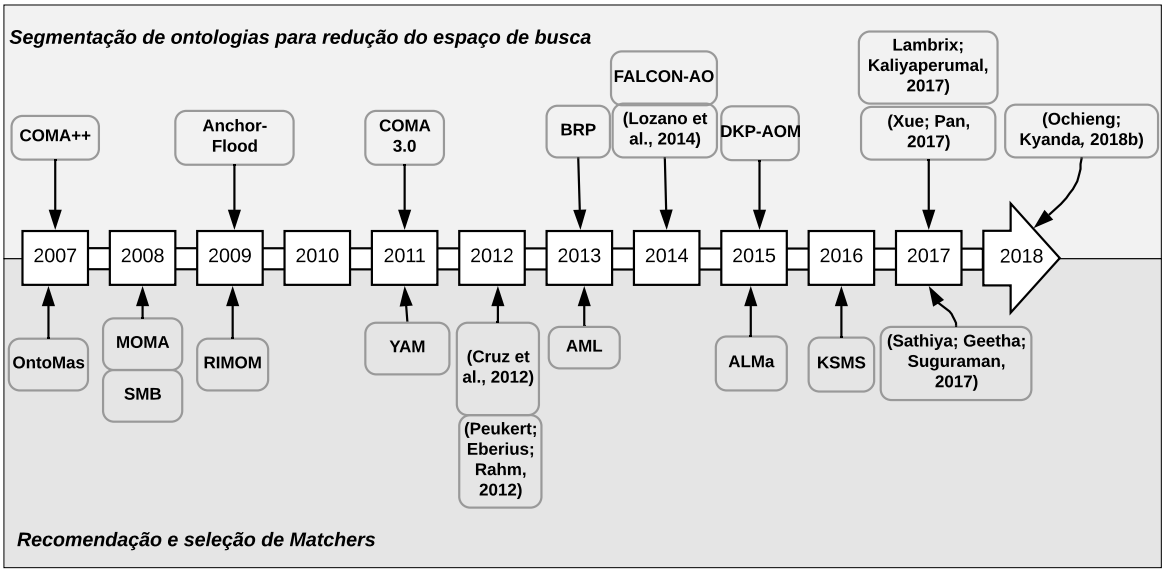


Figura 7 – Linha cronológica da publicação dos trabalhos relacionados.

A pesquisa dos trabalhos referenciados neste capítulo teve como base as principais fontes relacionadas aos trabalhos da área de alinhamento de ontologias, tais como o site ontologymatching.org[1] e anais de conferências e periódicos que abordam essa temática, com ênfase na segmentação de ontologias e recomendação de *matchers*. Foi definido como critério de seleção dos artigos o maior número de citações, de maneira a destacar ao menos um trabalho de referência publicado em cada ano, tomando como referência o período de 2007 a 2018.

[1] Acesso disponível no endereço: <http://ontologymatching.org>

3.1 SEGMENTAÇÃO DE ONTOLOGIAS PARA REDUÇÃO DO ESPAÇO DE BUSCA NO ALINHAMENTO DE ONTOLOGIAS

A redução do escopo de busca é um aspecto essencial ao considerar o problema de geração de alinhamentos de ontologias em larga-escala (OCHIENG; KYANDA, 2018a). A partir da redução dos elementos comparados, pode-se obter uma melhor eficiência na obtenção de correspondências, como também reduzir o número de validações requeridas ao usuário. Neste âmbito, a segmentação de ontologias é uma das estratégias que pode ser aplicada. Ela consiste em dividir uma ontologia em partes menores, podendo ser realizado de duas formas: i) extração de segmentos parciais - que obtém apenas uma parcela da ontologia com base em critérios pré-definidos e ii) segmentação da ontologia completa - que fornece conjuntos menores, porém contemplando todos os elementos da ontologia original.

A extração de segmentos parciais (também referenciada na literatura como extração de módulos) é utilizada quando se deseja considerar apenas um sub-conjunto de elementos de uma ontologia. Para tal, define-se um critério que é usado como referência para indicar os elementos que serão incluídos no segmento. Durante a sua construção, deve ser garantida a manutenção da semântica dos elementos extraídos. Alguns trabalhos exploram essa técnica para auxiliar no processo de construção ou representação de ontologias ou até mesmo possibilitar o reuso, tais como os propostos por Grau et al. (2006) e Bernardo Cuenca et al. (2008). No entanto, poucos exploram essa técnica para a geração de alinhamentos - Ela tem sido mais utilizada no processo de refinamento de resultados (e.g., reparo de alinhamentos), como é o caso do LogMap (JIMÉNEZ-RUIZ; GRAU, 2011) e do AML (FARIA et al., 2013), o que não caracteriza, neste caso, a redução do espaço de busca.

Já a segmentação da ontologia completa é abordada por uma quantidade maior de trabalhos e visa aplicar estratégias para quebrar uma ontologia em segmentos menores, contemplando todos os seus elementos. Num cenário ideal, a ontologia seria segmentada em blocos que representam os conceitos independentes, o que poderia facilitar também o reuso e a sua manutenção. Outras aplicações seriam o suporte a paralelização e a redução do tempo de execução dos *matchers* (OCHIENG; KYANDA, 2018a).

No contexto de geração de alinhamentos de ontologias, a segmentação envolve subdividir cada ontologia fornecida como entrada, de maneira que os segmentos produzidos mantenham a estrutura da original. Diferentemente de algoritmos de particionamento que produzem segmentos para outros propósitos (e.g., reuso, manutenibilidade), no âmbito da geração de alinhamentos, tal algoritmo teria uma rigorosa exigência de cumprir os seguintes aspectos (OCHIENG; KYANDA, 2018a):

1. Deve ser altamente escalável, i.e., deve ter baixa complexidade de tempo e espaço;

2. Deve permitir a definição de um tamanho ideal para os segmentos que são produzidos, de maneira integrada à ontologia, no sentido de evitar segmentos muito pequenos, que introduzem uma nova complexidade de espaço ao processo; ou segmentos muito

grandes, o que faria com que os benefícios em utiliza-los não fossem aproveitados ao máximo.

3. Deve posicionar as entidades mais similares em uma mesma partição.

A seguir, são apresentados os trabalhos que envolvem a segmentação de ontologias (parcial e completa), no contexto da geração de alinhamentos, seguindo a ordem cronológica de publicação.

3.1.1 COMA++ e COMA 3.0

Visando reduzir o número de comparações realizadas durante a geração de alinhamentos, o COMA++ (DO; RAHM, 2007b) propõe uma abordagem de "dividir e conquistar" baseada na decomposição de uma ontologia em menores partes, denominadas fragmentos. Neste caso, o alinhamento baseia-se em duas etapas: Na primeira etapa, os fragmentos são determinados e comparados uns com os outros para identificar os mais similares entre duas ontologias; na segunda, entidades dos fragmentos correspondentes são então alinhados.

Durante a geração dos fragmentos, o COMA++ usa regras heurísticas simples, baseadas, primordialmente, em técnicas de comparação de texto. No entanto, não são considerados métodos para computar a proximidade estrutural entre conceitos. Isso traz como consequência uma maior falta de controle na geração de segmentos. Neste caso, dependendo das ontologias apontadas, podem ser fornecidos poucos segmentos, i.e., contendo partes ainda com largo número de elementos, ou um grande conjunto de segmentos, porém de escopo tão reduzido que não haveria elementos suficientes para aproveitá-lo de maneira eficiente.

Posteriormente uma extensão do COMA++ foi proposta, denominada COMA 3.0 (MASSMANN et al., 2011). Nesta, foram incorporados algoritmos de *clustering*, utilizando a hierarquia de relacionamentos das ontologias. Assim, elementos com estrutura similar são agrupados no mesmo *cluster*, o que limitaria o escopo de busca para a geração de alinhamentos. Apesar disso, a estratégia utilizada para a clusterização é baseada em heurísticas fixas. Ou seja, independentemente da entrada, a estratégia utilizada sempre será a mesma. Isto pode resultar na separação de elementos que seriam essenciais em conjunto para a representação de um determinado conceito, gerando perda semântica.

3.1.2 Anchor-Flood

Visando reduzir o problema da perda semântica na extração de elementos de ontologias, o Anchor-Flood (SEDDIQUI; AONO, 2009) propõe uma abordagem que evita a segmentação das mesmas *a priori*. Ele inicia com a definição de um par de conceitos similares que estejam contido nas duas ontologias, denominado ponto de ancoragem (*anchor point*). A partir da região de referência inicial, prossegue-se rumo aos nós vizinhos e, dependendo do

sucesso no processo de descoberta de correspondências para a âncora, podem ser fornecidos segmentos de diferentes tamanhos.

A estratégia de segmentação do Anchor-flood é baseada em duas etapas. Os primeiros segmentos são gerados a partir dos relacionamentos taxonômicos e disjuntos dos conceitos na hierarquia da ontologia. São produzidos então *clusters* no nível conceitual, através da agregação de tipos similares e do distanciamento dos conceitos disjuntos. O alinhamento final é estabelecido pela comparação dos elementos em cada *cluster*.

O tamanho dos segmentos é definido dinamicamente a partir de uma âncora. De maneira que é explorada toda a vizinhança de elementos até todos os conceitos serem coletados ou até nenhum par de correspondência ser encontrado. O sistema foca em comparações segmento a segmento, não considerando a ontologia, em busca de melhorar a escalabilidade. O resultado fornecido contém conceitos e propriedades conectados semanticamente em diferentes segmentos. Para determinar as correspondências, o sistema utiliza métricas terminológicas (e.g., Wordnet e métricas de comparação de texto) e de similaridade estrutural. O cálculo final de similaridade é determinado pela proporção do número de superconceitos terminologicamente similares pelo seu número total. Pares de correspondência obtidos, anteriormente, podem ser consideradas como âncoras para processamentos futuros. Esse processamento é repetido até que não haja mais elementos a comparar.

3.1.3 BRP

Queiroz-Sousa, Salgado e Pires (2013) apresentam um algoritmo para construir sumários de ontologias baseado em parâmetros definidos pelo usuário, denominado BRP (*Broaden Relevant Paths*). Um sumário é um excerto contendo apenas os conceitos mais relevantes da ontologia, respeitando os relacionamentos e propriedades originais. Nesse pesquisa, os autores definem duas medidas de relevância: centralidade e proximidade. A medida de centralidade considera o número de relacionamentos entre os conceitos da ontologia e os seus tipos. Já a medida de proximidade indica o quão um conceito é referenciado por outros. Através da aplicação do algoritmo BRP, é extraída uma ontologia reduzida, contendo apenas os elementos mais relevantes.

Por ser um algoritmo de sumarização, o objetivo do BRP é criar uma versão resumida da ontologia fornecida, em que são destacados os seus principais elementos. Para o caso da geração de alinhamentos, pode ser de interesse do usuário comparar elementos que não sejam amplamente referenciados, o que não seria contemplado por esse algoritmo.

3.1.4 Abordagem Proposta por Lozano et al. (2014)

Na abordagem proposta por Lozano et al. (2014), os autores propõem a noção de visão de ontologia enriquecida (Well-founded Ontology View (WFOV)) e um conjunto de algoritmos capazes de extrair uma WFOV de uma ontologia base. Nessa teoria, as visões de ontologias são subconjuntos de uma ontologia extraídos de acordo com requisitos do

usuário e que podem sobrepor outras visões de ontologias. Por outro lado, na perspectiva dos autores, WFOV são visões que preservam propriedades ontológicas importantes, tais como: identidade, dependência existencial, entre outras.

As meta-propriedades de ontologia adotadas nesse trabalho são providas por uma meta-ontologia chamada Unified Foundational Ontology (UFO) (GUIZZARDI et al., 2015), que é um sistema de categorias bem fundamentado teoricamente e independente de domínio, cujos elementos podem ser usados para construir modelos de domínio específicos, como o descrito por Falbo et al. (2010).

O algoritmo proposto por Lozano et al. (2014), considera como entrada um conjunto de conceitos, que são selecionados pelo usuário, e extrai a visão mínima de ontologia que não viole as restrições filosóficas e cognitivas que devem ser seguidas pelas ontologias capturadas através das meta-propriedades. Essa abordagem foi proposta para permitir uma redução consistente de uma ontologia, visando prover aos usuários apenas segmentos de ontologia que são relacionados à tarefa que se tem em mãos.

Uma desvantagem da abordagem apresentada é a falta de flexibilidade em relação às restrições que são consideradas para a inclusão de conceitos. Alguns destes conceitos, que não sejam de interesse essencial ao usuário, poderiam ter regras menos rígidas. Outro aspecto importante é que todo o processo de construção das visões e extração de segmentos depende de elementos advindos de meta-ontologias. Isto faz com que seja pré-requisito que todos os conceitos das ontologias comparadas referenciem propriedades de uma meta-ontologia (como a UFO). Do contrário, não seria possível observar as restrições de conservação propostas.

Como tipicamente ontologias que são submetidas para o processo de alinhamento dificilmente possuem mapeamentos complexos para meta-ontologias, torna-se difícil a aplicabilidade dessa abordagem neste contexto. Adiciona-se ainda o fato de existirem várias meta-ontologias (e.g., General Formal Ontology (GFO) (ARP; SMITH; SPEAR, 2015), Upper Merged Ontology (SUMO) (NILES; PEASE, 2001), a Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) (GANGEMI et al., 2002)). Logo, diferentes meta-ontologias poderiam ser usadas para o mapeamento de meta-propriedades, o que poderia exigir um alinhamento prévio de meta-ontologias. Tal processo seria ainda mais complexo do que o alinhamento de ontologias tradicionais, visto que os elementos e relacionamentos presentes em meta-ontologias não são representação direta de conceitos do mundo-real.

3.1.5 FALCON-AO

O FALCON-AO (JAURO; JUNAIDU; ABDULLAHI, 2014) opera em três fases para a geração de alinhamentos em larga-escala. Na primeira são agrupadas as entidades das ontologias fornecidas como entrada em diferentes *clusters*. Na segunda, os *clusters* das ontologias fonte/alvo são comparados baseados em âncoras pré-definidas. Por último, é realizada a

comparação entre conceitos, se restringindo aos conceitos que fazem parte dos *clusters* correspondentes.

A estratégia de segmentação utilizada é baseada na proximidade estrutural entre classes e propriedades, e.g., maior proximidade das classes em uma hierarquia considerando relações rdfs:subClassOf, tendo como resultado o agrupamento em blocos. Após isso, os blocos advindos de ontologias diferentes são comparados com base em âncoras (correspondências obtidas anteriormente). Logo, quanto mais âncoras forem encontradas entre dois blocos, mais similares eles são. As âncoras são descobertas com o auxilio da técnica de comparação de *strings*. Os pares de blocos com maiores similaridades são selecionados baseados num limiar mínimo. Assim, cada bloco representa um pequeno fragmento de uma ontologia. Finalmente, na terceira fase, são combinados os resultados de técnicas linguísticas e estruturais para descobrir alinhamentos entre os blocos correspondentes. A saída fornecida é então extraída a partir de um algoritmo guloso.

Ressalta-se que o objetivo desta ferramenta é a geração do alinhamento completo entre pares de ontologias, e não a extração de segmentos para serem utilizados de maneira independente em alinhamentos parciais. Além disso, a vinculação da estratégia de blocagem ao relacionamento de subclasse (subClassOf), tende a separar os elementos que serão comparados, inicialmente, a partir de uma hierarquia comum, podendo não ser tão eficiente em ontologias que possuem significativas diferenças estruturais.

3.1.6 DKP-AOM

O DKP-AOM (FAHAD, 2015) segmenta uma ontologia exclusivamente pelo uso dos axiomas owl:disjointWith que são modelados na ontologia. Em um cenário ideal, o uso desse axioma seria o mais indicado para a segmentação de ontologias, visto que ontologias disjuntas não seriam incluídas no mesmo segmento. Porém, a maior parte das ontologias são modeladas sem incluir explicitamente os axiomas owl:disjointWith, limitando a eficiência da segmentação com base nesse tipo de axioma. Isto acaba por explicar porque essa ferramenta não consegue completar a geração de alinhamentos na trilha de ontologias biomédicas de larga-escala da OAEI.

3.1.7 Ferramenta Proposta por Lambrix e Kaliyaperumal (2017)

A primeiro ferramenta introduzida pelos autores foi o SAMBO (LAMBRIX; TAN; LIU, 2008), um sistema para o alinhamento de ontologias biomédicas. Nele, utiliza-se a combinação de vários *matchers* baseados em texto para o processamento de alinhamentos. A falta de uma estratégia de otimização foi uma das desvantagens levantadas inicialmente, sendo inclusive um ponto que impediu a ferramenta de prover resultados para algumas trilhas da OAEI, como a de anatomia. A ausência de uma interface gráfica e o alto tempo de execução foram apresentados inicialment pela ferramenta.

Posteriormente, Lambrix e Kaliyaperumal (2017) adaptaram a estrutura inicial do SAMBO a partir da introdução do conceito de alinhamentos baseado em sessões (*session-based ontology alignment*). Visando a otimização, os autores propõem três estratégias baseadas na utilização de segmentos de ontologias. Na primeira, são gerados pares de segmentos baseado na correspondência exata de um conjunto de conceitos. Na segunda, considera-se um conjunto de correspondências sugeridas pelos *matchers*. Por fim, consideram-se apenas as correspondências validadas pelo usuário.

Nessa nova versão, também foi apresentada uma interface gráfica em que o usuário pode realizar incrementalmente a avaliação dos alinhamentos decorrentes da comparação de segmentos. Cada etapa é denominada sessão e o usuário pode escolher interromper o processo em determinado ponto e retomar quando for de seu interesse.

Como os critérios para a construção de segmentos são basicamente os níveis de similaridades entre elementos, essa técnica seria mais adequada ao processo de segmentação de ontologias completas, tendo em vista que os segmentos gerados por ela podem não fazer sentido se considerados individualmente, visto que são compostos apenas de pares de elementos similares.

3.1.8 Abordagem proposta por Xue e Pan (2017)

Xue e Pan (2017) propõem uma abordagem de alinhamento de ontologias baseada em segmentos em que o termo segmento é um fragmento de ontologia que faz sentido isoladamente (SEIDENBERG et al., 2006). Os autores aplicam algoritmos de particionamento de ontologias para transformar um problema de alinhamento em larga-escala em vários problemas menores de alinhamentos entre segmentos.

Na prática, o algoritmo proposto primeiro particiona a ontologia-fonte em vários segmentos disjuntos. Cada um deles é então utilizado para particionar o seu similar com base na ontologia alvo, a partir de uma medida de relevância de conceitos. Finalmente, os segmentos similares são comparados e agregados de uma vez em um alinhamento final gerado por um algoritmo evolucionário, detalhado em Xue e Wang (2015).

Essa estratégia mostra-se eficiente para cenários em que deseja-se utilizar a segmentação como forma de otimizar a comparação de ontologias inteiras, aplicando estratégias de otimização existentes, como a execução em paralelo. No entanto, haveria maior dificuldade para realizar a geração de alinhamentos parciais, visto que os segmentos extraídos são gerados sem que seja possível estabelecer algum critério para direcionar a escolha dos conceitos que serão agrupados. Outro ponto é que o algoritmo não prevê nenhuma estratégias mais robustas para garantir a manutenção semântica dos elementos, visto que são utilizadas como regra principal para a criação dos segmentos apenas as relações de disjunção.

3.1.9 Abordagem Proposta por Ochieng e Kyanda (2018a)

No trabalho proposto por Ochieng e Kyanda (2018a), as ontologias são representadas como um grafo tradicional, de maneira que algoritmos de particionamento de grafos existentes são utilizados para dividi-las em n partições, de maneira a minimizar o custo total para extração dos segmentos. A utilização de algoritmos de particionamento de grafos é motivada pelo trabalho proposto por (CURINO et al., 2010), que demonstrou que aplicar heurísticas de particionamento de grafos para dividir modelos de representação de dados tem um menor custo de execução e produção balanceada. Isso é verificado pelos autores, que demonstraram que a técnica de *spectral clustering* (CHAN; SCHLAG; ZIEN, 1994) gera partições de alta qualidade para ontologias inerentes ao processo de geração de alinhamentos. Essa é uma das heurísticas de maior sucesso para o particionamento de grafos e matrizes (SPIELMAN; TENG, 2007) e tem sido aplicada em vários domínios, como por exemplo, segmentação de imagens (MALIK et al., 2001).

Com base nessa técnica, Ochieng e Kyanda (2018a) demonstram como modelar a segmentação de uma ontologia OWL como um problema de clusterização de dados usando um modelo de grafo. Para tal, as entidades de uma ontologia são consideradas como nós do grafo; as correspondências entre diferentes nós são utilizadas para representar ligações extras entre nós e os valores de similaridade são usados como pesos. Desta forma, o problema de particionamento de ontologias é transformado em um problema de particionamento de grafos, no qual as entidades das ontologias são divididas em diferentes *clusters*. Eles são criados de maneira a minimizar as similaridades entre *clusters* diferentes e maximizar as similaridades dentro de um mesmo *cluster*.

Nessa abordagem também é possível selecionar quais axiomas serão considerados para realizar a comparação entre os elementos de diferentes ontologias (e.g., owl:equivalentClass, hasExactSynonym), o que oferece um maior controle dos segmentos que são gerados. No entanto, em relação a isso, o usuário precisa indicar uma quantidade fixa k de segmentos que devem ser geradas. Como essa definição depende de diversos fatores relacionados às ontologias envolvidas, a decisão acaba sendo tomada por um processo de tentativa e erro. Mais recentemente, os autores propuseram a integração de modelos estatísticos como opção para a identificação do número ideal de segmentos (OCHIENG; KYANDA, 2018b).

3.1.10 Discussão

Nas seções anteriores, foram discutidos trabalhos que utilizam diferentes estratégias para a segmentação de ontologias, cujo objetivo principal é a redução do espaço de busca para a geração de alinhamentos. A Tabela 1 apresenta uma análise comparativa dos trabalhos supracitados.

Nota-se que a maior parte das ferramentas concentra esforços na geração de segmentos para a comparação de ontologias completas, não se atendo à conservação semântica dos

elementos que compõem os segmentos. Isto foi abordado com maior profundidade por Lozano et al. (2014), no entanto, a dependência do uso de anotações vinculadas a meta-ontologias, acaba limitando as ontologias que seriam aceitas na geração de alinhamentos.

Outro fator a destacar é o controle dos elementos que são incluídos no segmentos. Como boa parte dos trabalhos têm como premissa a geração de alinhamentos completos entre as ontologias, eles acabam não se preocupando em conceder ao usuário a opção de construir segmentos com base em conceitos de sua preferência. O trabalho de Lozano et al. (2014) é um dos que vão neste sentido, considerando um conjunto de conceitos como entrada.

A quantidade de segmentos gerada também é um fator que é tratado de maneira distinta nos trabalhos expostos. Boa parte dos trabalhos (e.g., (JAURO; JUNAIDU; ABDULLAHI, 2014), (MASSMANN et al., 2011), (SEDDIQUI; AONO, 2009)) escolhe por não oferecer nenhuma restrição quanto ao número de segmentos, fazendo com que a quantidade de similaridades descobertas seja o fator determinante. Por outro lado, o trabalho proposto por Ochieng e Kyanda (2018a) requer que seja definido um número fixo, o que acaba por ser uma tarefa complexa para o usuário, fazendo com que a decisão seja tomada via tentativa e erro. Na abordagem proposta, introduz-se o conceito de requisitos operacionais, que trata-se de um conjunto de diretrizes que influenciam na quantidade e no tamanho dos segmentos gerados, dando uma maior flexibilidade neste processo.

Outro detalhe comum a vários trabalhos (e.g., (XUE; PAN, 2017), (OCHIENG; KYANDA, 2018a), Jauro2014) é a utilização da estrutura de grafos para representar ontologias. Apesar de uma ontologia descrever uma estrutura hierárquica, ao considerar relacionamentos como por exemplo o rdf:subClassOf, ela pode ser representada também em outras estruturas, como índices estruturais (e.g., (DUCHATEAU; BELLAHSENE, 2015)), índices léxicos (JIMÉNEZ-RUIZ; GRAU, 2011) e mapas (*hashmaps*) (SANTOS et al., 2016). Porém, como demonstrado por Ochieng e Kyanda (2018a), quando se trata da segmentação de ontologias, a estrutura de grafo tem se mostrado mais eficiente. Na abordagem proposta, ao serem importadas as ontologias no catálogo, elas são convertidas em uma estrutura de grafos, de maneira que, possibilita o uso de algoritmos de grafos existentes para a manipulação e busca de elementos. Além disso, ainda utilizam-se bases de conhecimento externas para enriquecer o grafo de dados que serão úteis para segmentação com base em requisitos, como os sinônimos de cada elemento.

Por fim, destaca-se as abordagens apresentadas que focam majoritariamente na geração de alinhamentos completos entre as ontologias. Neste sentido, não foram encontradas abordagens que possibilitem a geração de alinhamentos apenas entre segmentos que estejam relacionados aos requisitos do usuário. Isto seria especialmente útil quando considera-se ontologias contendo muitos elementos ou até mesmo quando se deseja comparar um grande número de ontologias, trazendo como consequência a redução do número de correspondências geradas e que porventura venham a precisar passar por validação do usuário.

Ferramenta	Objetivos	Técnicas	Conservação Semântica	Escalável	Limitações
COMA++/ COMA 3.0	Segmentação da ontologia visando o alinhamento completo	Segmentação baseada em comparação léxicas - Utilização de heurísticas fixas.	Não	Sim	Falta de controle em relação ao tamanho dos segmentos gerados - Não é possível especificar os conceitos mais relevantes para a extração de segmentos.
Anchor-Flood	Extração de partes da ontologia com base em similaridades conhecidas entre as duas ontologias fornecidas (âncoras), visando o alinhamento completo em várias iterações	Definição de âncoras como referência básica para a geração do segmento - Estrutura baseada em gratos.	Não	Sim	Necessidade de correspondências iniciais verificadas entre as ontologias fornecidas (âncoras) - Menor controle do escopo dos segmentos (depende das similaridades com a âncora fornecida) - Pode exigir um número grande de iterações para chegar ao resultado final.
BRP	Sumarização de ontologias através da extração de elementos com maior relevância	Extração de elementos mais relevantes de uma ontologia - Utilização de métricas de centralidade e proximidade para calcular relevância de elementos.	Não	Não	Elementos com baixo número de relacionamentos são excluídos do segmento gerado - Foco é na extração de uma versão resumida da ontologia, não na redução do escopo para geração de alinhamentos.
Abordagem Proposta por Lozano et al. (2014)	Segmentação da ontologia parcial (extração de visões) com base em atributos de meta-ontologias	Construção de visões de ontologias a partir de conceitos fornecidos pelo usuário - Preservação de propriedades semânticas a partir de meta-propriedades.	Sim	Não	Falta de flexibilidade em relação às regras rígidas para a manutenção de conservação - Alto tempo de execução - Exigência de que cada elemento das ontologias fornecidas seja vinculado a propriedades de uma meta-ontologia (e.g., UFO)
FALCON-AO	Segmentação da ontologia completa a partir da blocagem de elementos com base na proximidade estrutural	Comparação léxica; Blocagem com base nas similaridades estruturais - Descoberta e comparação com base em âncoras.	Não	Sim	Vinculação exclusiva a hierarquia de relacionamentos rdfs:subClassOf para a blocagem inicial - Falta de controle no tamanho dos segmentos extraídos
DKP-AOM	Segmentação de ontologias com base nos relacionamentos de disjunção	Agrupamento de elementos das ontologias com base nos relacionamentos owl:disjointWith.	Não	Não	Vinculação exclusiva da estratégia de segmentação aos axiomas owl:disjointWith, que são pouco comumente usados em ontologias submetidas para a geração de alinhamentos.
Abordagem Proposta por Xue e Pan (2017)	Segmentação da ontologia alvo com base no segmento da ontologia fonte	Segmentação da ontologia fonte com base na análise de elementos disjuntos - Segmentação da ontologia alvo com base nos segmentos obtidos para a ontologia fonte.	Não	Sim	Falta de controle da quantidade de elementos e dos critérios que serão utilizados para a geração dos segmentos.
Abordagem Proposta por Ochieng e Kyanda (2018a)	Segmentação da ontologia completa com base em algoritmos de gratos	Segmentação com base na utilização de técnicas de particionamento de grafos	Não	Sim	Definição da quantidade de segmentos a serem gerados via tentativa e erro.
Ferramenta Proposta por Lambrix e Kaliyaperumal (2017)	Segmentação de ontologias com base no nível de similaridade para divisão da geração de alinhamentos em sessões	Segmentação de ontologias com base em três níveis de similaridade: i) exata, ii) recomendada por um matcher e iii) recomendada e validada por usuário	Não	Sim	Agrupa segmentos apenas por similaridade, não se atendo a agrupar um conjunto de elementos que mantenha o significado semântico.

Tabela 1 – Análise comparativa dos trabalhos de segmentação de ontologias

3.2 RECOMENDAÇÃO E SELEÇÃO DE MATCHERS

Nesta seção são descritas algumas abordagens que visam a recomendação e seleção de *matchers*, seguindo a ordem cronológica de publicação. Ao término da seção, é apresentado um quadro comparativo destacando os principais aspectos e limitações das abordagens existentes na literatura.

3.2.1 OntoMas

OntoMas (HUZA; HARZALLAH; TRICHET, 2007) aparece como uma primeira abordagem para auxiliar no processo de descoberta da melhor ferramenta para executar alinhamentos. O seu objetivo é auxiliar o usuário a encontrar o *matcher* mais adequado para determinada tarefa, i.e., par de ontologias para a geração de alinhamentos. Para a categorização, os autores usam uma classificação similar a proposta por (SHVAIKO; EUZENAT, 2005). A atribuição das ferramentas a cada categoria proposta é realizada através da aplicação de questionários.

As informações extraídas das ontologias fornecidas como entrada, que caracterizam o conceito de tarefa de alinhamento considerado no trabalho, leva em consideração apenas aspectos sintáticos dos seus elementos. Isto se deve ao fato de que a ferramenta foi desenvolvida para ser utilizada por usuários sem o conhecimento profundo da estrutura das ontologias fornecidas.

Para a recomendação de *matchers*, o modelo de decisão multi-critérios utilizado é baseado em regras *ad hoc*. Sendo alguma dessas filtros que eliminam sistemas considerados inadequados (com base nas categorias), enquanto outras aumentam ou diminuem a pontuação dos métodos considerados.

Como o número de ferramentas para a geração de alinhamentos foi crescendo ao longo dos anos, em um cenário atual, seria impraticável realizar aplicações de questionários com especialistas para tentar levantar e identificar quais aspectos se sobressaem de todos os possíveis *matchers* existentes. Na época, esse modelo chegou a ser utilizado e foi aperfeiçoado com a introdução do framework MOMA, descrito adiante.

3.2.2 MOMA

Visando fornecer maiores evidências para auxiliar na seleção de *matchers* adequados para uma tarefa específica, Mochol, Jentzsch e Euzenat (2006) identificaram um amplo conjunto de características, definido como MCMA, composto nos seguintes níveis:

i. Entrada (e.g., número de elementos, formato e modelo de representação dos esquemas fornecidos);

ii. Abordagem (e.g., tipo (individual ou composição), processamento (manual, semi-automático ou automático), tipo de relação de similaridade (semântico, sintático), nível de comparação (elemento ou estrutural));

iii. Uso (e.g., estatísticas de utilização em diferentes domínios, aplicações e tarefas de *matching*, utilizado por humanos ou máquinas);

iv. Saída (e.g., tipo de resultado gerado (alinhamentos, valor global, regras), cardinalidade dos relacionamentos identificados);

v. Documentação (e.g., clareza e qualidade da documentação disponibilizada pelos *matchers*) e

vi. Custo (e.g., preço da licença para uso comercial da ferramenta).

Para realizar a seleção dos *matchers* é necessário o fornecimento de um conjunto de requisitos que indiquem os valores desejados e os pesos considerados para cada característica correspondente a essas categorias. O método AHP (Analytic Hiearchical Process) é utilizado para fazer a comparação par a par, no sentido de avaliar quais são os mais adequados com base nos requisitos fornecidos.

Esse trabalho foi estendido por Mochol e Jentzsch (2008), no qual foi introduzido o *framework* MOMA, cujo principal diferencial é a concepção de regras com base nas características definidas no MCMA. Por exemplo, a regra noinstances define que em caso de não fornecimento de dados sobre as instâncias, apenas os *matchers* que operam no nível do esquema serão considerados. Essas regras são representadas no formato SWRL (HORROCKS et al., 2004) e são armazenadas em um repositório, que será acessado para determinar a compatibilidade de um esquema fornecido com um determinado *matcher*. No entanto, podem ser observados os seguintes pontos fracos quanto a esse trabalho: (i) o usuário precisa definir e priorizar manualmente um amplo conjunto de características por meio do preenchimento de um questionário; (ii) não é fornecida nenhuma classificação ou lista de características dos *matchers* existentes em relação ao MCMA, e (iii) a abordagem foi avaliada através de um estudo de caso e de questionários de satisfação aplicados com um conjunto de usuários, não havendo experimentos, demonstrando, na prática, a efetividade da abordagem proposta.

3.2.3 SMB

O Schema Matcher Boosting (SMB), introduzido por Marie e Gal (2008) e aperfeiçoado posteriormente por Gal e Sagi (2010), foi um dos primeiros trabalhos a discutir a seleção automática de *matchers*. A abordagem utiliza um algoritmo de Boosting para selecionar pares, no qual cada par contém um *matcher* de primeiro nível (os que recebem um conjunto de esquemas e fornecem alinhamento) e um *matcher* de segundo nível (os que recebem

alinhamentos e aplicam métodos para classificá-los, selecionando um conjunto final de correspondências).

No método definido pelo SMB, o *matcher* selecionado de primeiro nível gera correspondências que são classificadas pelo de segundo nível. A classificação é baseada no atributo de "monotonicidade", princípio introduzido por Gal, Anaby-Tavor e Trombetta (2003), que estabelece que um *matcher* é monotônico, se ele consegue classificar todas as correspondências conforme o nível de precisão delas. Mesmo considerando este princípio, os *matchers* considerados são originalmente independentes, podendo haver pouca ou nenhuma correlação entre as premissas que cada um utiliza para definir a classificação correta de uma correspondência. Neste ambiente, os *matchers* são considerados como classificadores "fracos"(*weak classifiers* (SCHAPIRE, 1990)), visto que não se pode garantir a sua confiabilidade.

Para lidar com esse problema, o SMB utiliza um algoritmo de *Boosting* (SCHAPIRE, 1990), que percorre em várias etapas os conjuntos de treinamento dos classificadores para ajustar a importância dos elementos presentes, concedendo valores positivos para os pares classificados corretamente e negativos para os incorretos. Após várias iterações, os classificadores conseguem obter desempenhos satisfatórios em alguns cenários, com o benefício de possibilitar a construção de um classificador adaptado, gerado sob-medida através do refinamento a cada iteração. Essa abordagem proporciona a sua aplicação também no processo de calibração (*tuning*) automática, a partir da descoberta dos valores mais adequados para os parâmetros de configuração dos *matchers*.

O principal ponto fraco do SMB é a forte dependência de um classificador individual para o processo de *Boosting*. Apesar da abordagem considerar um conjunto de *matchers* durante a classificação, ao final do processo é selecionado apenas um, que esteja apto a analisar as ontologias e gerar novas correspondências. Desta forma, as iterações para atualizar os classificadores baseiam-se nos resultados gerados por *matchers* executados individualmente. Essa confiança em um único classificador é, de certa forma, arriscada, pois pode enviesar o processo. Para minimizar esse problema, podem ser adotados classificadores múltiplos, como, por exemplo, as implementações AdaBoost.M1 e AdaBoost.M2 (FREUND; SCHAPIRE, 1997) do próprio algoritmo de *Boosting*. Um exemplo de abordagem que agrega vários classificadores durante a etapa de seleção é o YAM (DUCHATEAU; BELLAHSENE, 2015), descrito a seguir.

3.2.4 RIMOM

A abordagem proposta por Li et al. (2009), denominada RiMOM, avalia um conjunto de *matchers* com base em dois fatores de similaridade: de rótulos (*(labels)*) e estrutural, que são computadas a partir das estrutura conceitual e estrutural das ontologias fornecidas no processo. A classificação de cada *matcher* ocorre pela definição dos pesos corretos para cada fator. Além disso, uma estratégia é aplicada para escolher qual *matcher* usar de

acordo com um limiar (*threshold*) ideal, obtido empiricamente.

O processo proposto pelo RiMOM consiste em cinco etapas:

i. Pré-processamento: identificação das similaridades entre as entidades e propriedades dos esquemas;

ii. Alinhamento Baseado em Técnicas Linguísticas: execução de múltiplas estratégias baseadas na resolução das heterogeneidades sintáticas presentes nos esquemas;

iii. Combinação de Similaridades: utilização dos pesos calculados na primeira etapa como parâmetro de entrada para a combinação dos valores obtidos pelas estratégias selecionadas;

iv. Propagação de Similaridades: execução da técnica de SF (MELNIK; GARCIA-MOLINA; RAHM, 2002) para realizar a propagação de três estratégias de similaridade estrutural: conceito-conceito, propriedade-propriedade e conceito-propriedade;

v. Geração do Alinhamento e Refinamento: são realizados ajustes finos nos resultados obtidos e, por fim, é gerado o alinhamento resultante do processo.

Como utiliza métricas anteriores à execução dos alinhamentos para definir os *matchers* mais adequados, em alguns domínios de conhecimento cuja a estrutura das ontologias é complexa (e.g., bio-medicina), as informações utilizadas previamente seriam insuficientes para a recomendação. Isso foi demonstrado nos resultados da campanha da OAEI de 2013 (ZHENG et al., 2013), no qual a ferramenta não rodou no conjunto de dados de anatomia.

3.2.5 YAM

O YAM (BELLAHSENE; NGO; BELLAHSENE, 2012) é o primeiro trabalho a ser denominado como fábrica de *matchers*, projetado para produzi-los sob-demanda, baseado na configuração automática dos parâmetros e na integração opcional de preferências do usuário. Enquanto que os *matchers* tradicionais produzem correspondências entre ontologias, o YAM produz um *matcher* dedicado a um cenário, podendo ser utilizado depois para descobrir correspondências.

Para atingir esse objetivo, o YAM considera o processo de alinhamento de ontologias como um problema de classificação (*Machine Learning* (MITCHELL, 1997)), definindo-o da seguinte maneira: dada algumas características (i.e., valores de similaridades computadas usando várias métricas distintas), um *matcher* deve prever a correspondência entre um par de elementos de ontologias. Nesta abordagem, qualquer tipo de classificador pode ser treinado para tal, visto que há inúmeros métodos disponíveis para combinar medidas de similaridade.

Neste caso, cada par de elementos das ontologias é considerado como um objeto, no qual seus atributos são os valores de similaridade computados pelo conjunto de medidas

selecionadas para estes elementos. Dado os valores de similaridades de um par, um *matcher* rotula este como sendo correspondente ou não. Um *matcher* pode usar qualquer algoritmo pra computar seu resultado - classificação, clusterização, agregação de medidas de similaridade ou quaisquer outros métodos *ad hoc*. No YAM, é usada a biblioteca WEKA[2], que disponibiliza um conjunto de classificadores.

Outra característica do YAM é a integração de conhecimento prévio ou preferências do usuário, se disponíveis, durante a geração dos *matchers*. Assim, o usuário pode definir sua preferência por precisão ou cobetura como medida de avaliação de qualidade, sendo assim produzido um *matcher* dedicado que melhor atenda a essa necessidade.

A geração do *matcher* dedicado é dividida em duas etapas: 1) calibração (*tuning*), que visa estabelecer a melhor configuração possível e 2) seleção, que provê a combinação dos melhores *matchers*. Tal processo pode ser afetada pelos parâmetros e preferências fornecidos pelo usuário. Neste aspecto, o YAM disponibiliza 5 opções: (i) preferência por precisão ou cobertura; (ii) correspondências de referencia; (iii) cenários de um mesmo domínio; (iv) número de conjunto de treinamento a utilizar e (v) estratégia para selecionar um *matcher* dedicado. Essas opções são utilizadas para direcionar a seleção do *matcher*. Ressalta-se que o usuário não possui a obrigação de prover essas opções, especialmente, no que se refere ao conjunto de treinamento e a estratégia que será selecionada automaticamente pelo YAM quando necessário.

Destaque-se que, neste caso, o usuário é designado a selecionar um classificador apropriado ou usar um padrão definido através da interpretação de informações provenientes de uma base de conhecimento. No entanto, os classificadores padrões não são suficientes para lidar com conjuntos de dados de domínios diversificados, o que levaria a obtenção de resultados insatisfatórios em cenários não conhecidos previamente. Para minimizar este aspecto, o YAM realiza várias iterações para refinar os valores de similaridade entre esquemas. Este processo até consegue eliminar algumas correspondências incorretas, mas acaba gerando um alto custo em relação ao tempo de execução.

3.2.6 Abordagem Proposta por Peukert, Eberius e Rahm (2012)

Peukert, Eberius e Rahm (2012) propõem uma abordagem adaptativa e auto-configurável. O sistema baseia-se na análise dos dados fornecidos como entrada em conjunto com resultados provisórios obtidos por diferentes componentes internos: *matchers* básicos, métodos de agregação de correspondências e métodos para geração de alinhamentos finais. Na etapa de pré-processamento, antes de iniciar o processo de *matching*, o sistema decide quais os *matchers* básicos serão utilizados. Para isto, são analisadas as entidades contidas nas ontologias fornecias como entrada. Caso elas contenham, por exemplo, rótulos expressivos, o sistema pode decidir pela adoção de um *matcher* básico que determine as correspondências com base nos rótulos das entidades.

[2] Disponível em: <https://weka.wikispaces.com>

Depois da execução dos *matchers* básicos selecionados, o sistema avalia os resultados obtidos, excluindo os que não obtiveram qualidade satisfatória, apesar de terem sido selecionados na etapa anterior. Da mesma forma, o sistema escolhe qual dos métodos de agregação (e.g., valor mínimo/máximo, média, média ponderada (PEUKERT; MASSMANN; KOENIG, 2010)) será aplicado, como também os métodos para a geração dos alinhamentos finais (e.g., limiar mínimo para o valor de similaridade [*threshold*], número máximo).

Para guiar o processo de recomendação, os autores usam uma matriz de características, que são calculadas durante o fornecimento dos alinhamentos, em conjunto com características da ontologia que possam ter alguma influência no processo. Na época, o sistema alcançou bons resultados nas campanhas da OAEI, mas os resultados dependiam primordialmente dos métodos utilizados pelos *matchers* disponíveis no sistema. Por exemplo, se nenhum dos métodos de agregação pudessem melhorar significativamente os resultados obtidos, ou se uma coleção de *matchers* básicos não fosse capaz de usar toda informação disponível contida nas ontologias, logo, o resultado fornecido não seria o melhor possível.

Outro aspecto é que as características levantadas para as ontologias (e.g., a existência de nome, número de elementos repetidos) representam apenas atributos básicos. O trabalho proposto por Cruz et al. (2012), discutido na sequência, propõe um conjunto de métricas mais abrangentes para caracterizar as ontologias fornecidas.

3.2.7 Abordagem Proposta por Cruz et al. (2012)

Na abordagem proposta por Cruz et al. (2012), os *matchers* são escolhidos a partir de uma determinada categoria, levando em consideração se as características da tarefa de *matching* possuem informação quantitativa suficiente para a referida categoria. Para quantificar as características de um par de ontologias fornecida, utiliza-se um conjunto de nove métricas de qualidade para ontologias, categorizados como léxicos, estruturais, sintáticos e de instância (conforme detalhado na Seção 2.5).

Na etapa seguinte, um conjunto pré-definido de *matchers* fornecidos pela ferramenta AgreementMaker (CRUZ; ANTONELLI; STROE, 2009) são combinados, sendo denominados "*matchers* configurados". Neste âmbito, cinco combinações são apresentadas, cada uma correspondendo a um conjunto único de características. Assim, qualquer nova demanda por geração de alinhamentos é quantificada usando as métricas propostas. Para efetuar a recomendação, os autores utilizam um algoritmo de aprendizado supervisionado K-Nearest Neighbour (KNN). O modelo construído leva em consideração a quantificação das métricas propostas, em relação ao par de ontologias fornecido, para escolher o melhor dos cinco "matchers configurados".

No entanto, não foram disponibilizados conjuntos de dados de acesso aberto para construir o modelo KNN, dificultando a construção do conjunto de treinamento requerido para o funcionamento do sistema. Como alternativa, pode-se utilizar o conjunto fornecido pela OAEI, mas, mesmo considerando todas as trilhas, só seria possível construir apenas

cerca de 233 conjuntos de dados rotulados. Isto se deve ao fato de que para a construção de dados rotulados é requerido o uso de alinhamentos de referência, que por sua vez possuem disponibilidade limitada, tendo em vista que a sua criação é um processo dispendioso e depende da intervenção especialistas no domínio. Logo, uma importante desvantagem desse trabalho é o uso de um conjunto de treinamento bastante limitado para modelar um modelo supervisionado KNN. Geralmente, algoritmos de aprendizado supervisionado necessitam de um número amplo de dados rotulados para evitar problemas como o sobreajuste (*overfitting*) ou subajuste (*underfitting*) do modelo.

3.2.8 AML

AgreementMakerLight (AML) (FARIA et al., 2013) é um sistema de alinhamento de ontologias derivado do AgreementMaker (CRUZ; ANTONELLI; STROE, 2009), voltado à resolução do problema em larga-escala, com abrangência no domínio de biomedicina. Por ser focado em escalabilidade, ele inclui recursos para maximizar a efetividade na geração de alinhamentos, incluindo a redução da complexidade da execução. Essa ferramenta divide-se em três principais módulos: carregamento de ontologias, alinhamento de ontologias e seleção/reparo de alinhamentos.

O módulo de carregamento de ontologias é responsável por percorrer os elementos de ontologias para representá-los através de estruturas de dados internas do AML, que são concebidas para permitir a execução em uma complexidade linear. Dentre as estruturas, destaca-se a denominada por *Lexicon*. Trata-se de uma tabela de nomes de classes e sinônimos de uma ontologia, que utiliza um sistema de classificação para atribuir um peso a elas, de acordo com a pontuação obtida nas suas correspondências (PESQUITA et al., 2013).

O módulo de alinhamentos de ontologias contém um conjunto pre-determinado de *matchers* (derivados do AgreementMaker). Eles são divididos em primários e secundários. O primeiro tipo é formado por algoritmos de complexidade linear, que podem ser aplicados globalmente em quaisquer problemas de *matching*. Já o segundo, consiste de algoritmos de complexidade polinomial, que podem ser aplicados essencialmente em problemas de larga-escala. O AML contém a implementação de quatro *matchers* básicos, sendo três primários e um secundário - todos baseados em comparação de elementos textuais.

Por fim, a seleção e o reparo de alinhamentos garante que ao final, estes possuam a cardinalidade desejada, como também mantenha a sua coerência, i.e., não viole restrições presentes nas ontologias. O algoritmo de reparo de alinhamentos do AML apresenta um novo mecanismo de segmentação, que identifica um conjunto mínimo de classes que devem ser analisadas para garantir a coerência, reduzindo o escopo do problema de reparo (SANTOS et al., 2016). Note que neste caso, a segmentação é utilizada apenas para otimizar o processo de reparo e não para a redução do espaço de busca para a geração de alinhamentos, como é o caso dos trabalhos citados na Seção 3.1.

3.2.9 ALMa

O ALMa (RODRIGUES et al., 2015) concentra-se no treinamento de classificadores a partir de similaridades obtidas por *matchers* existentes e por um conjunto reduzido de exemplos fornecidos pelo usuário especialista. A ideia da abordagem é fazer um uso inteligente do esforço do usuário, fazendo com que a sua intervenção seja necessária apenas para avaliar instâncias que dividem a opinião de um comitê, possibilitando o re-treinamento de classificadores inicialmente fracos, devido ao número reduzido de exemplos.

O método baseado em aprendizado ativo definido pelo ALMa é composto por quatro etapas: (i) Seleção - um conjunto de correspondências são selecionadas para serem rotuladas por um usuário; (ii) Treinamento - o usuário rotula os pares selecionados que são usados para compor um conjunto de árvores de decisão, sendo atribuída a cada, uma pontuação que estima o seu grau de confiança. (iii) Eleição do comitê - um subconjunto de árvores de decisão é selecionado para compor um comitê de árvores, que será responsável por decidir quais pares (não-rotulados) são correspondências verdadeiras e (iv) Votação - todas as árvores de decisão anotam um voto para cada par. É importante ressaltar que apenas os votos das árvores que fazem parte do comitê são considerados para decidir se uma correspondência é verdadeira, enquanto que os pares rotulados pelo usuário são usados para gerar os pesos (confiabilidade) de cada árvore de decisão.

Desta forma, através da participação ativa do usuário, o ALMa consegue prover a confiabilidade de cada árvore de decisão. Como os valores que compõem cada árvore são gerados por um *matcher*, pode-se aplicar este mesmo método para avaliá-los quanto ao seu grau de confiança para gerar correspondências corretas. Os *matchers* utilizadas pelo ALMa foram os citados no processo de avaliação do COMA (MASSMANN et al., 2011) e os disponibilizados pelo projeto SecondString[3].

Ressalta-se que, ao considerar um ambiente em larga-escala, o conjunto de treinamento, construído de maneira ativa e modelado como uma árvore de decisão, pode não ser suficiente para fornecer predições em um nível mais abrangente, dada a alta sensibilidade do modelo às classificações feitas pelo usuário. Neste caso, pode haver a necessidade de um maior número de iterações para que seja possível atingir resultados mais satisfatórios. Como forma de resolver essa limitação, recentemente os autores trabalharam para estender a abordagem no sentido de dar suporte a redes de esquemas (RODRIGUES, 2018), visando aproveitar modelos de restrições baseados em grafos para estabelecer uma solução mais abrangente.

3.2.10 KSMS

Outra alternativa para esse problema seria a extração mais eficiente do conhecimento do usuário, por meio da identificação de regras a partir do que é fornecido. Essa ideia é

[3] Disponível em: <http://secondstring.sourceforge.net>

aplicada no trabalho de Anam et al. (2016), que apresenta o KSMS, um sistema baseado na aquisição de conhecimento (knowledge-based) para realizar a atividade de *ontology/schema matching* em dois níveis: estrutural e de elemento.

No nível de elemento, é utilizada a abordagem Hybrid-RDR, que combina árvores de decisão J48 e a abordagem aquisição incremental de conhecimento, denominada CPR, baseada no conceito de *Ripple-Down Rules* (KIM; COMPTON; KANG, 2012). Ela inicia com a base de conhecimento vazia, sendo a primeira regra adicionada através da classificação de um conjunto de dados utilizando o modelo J48. Em seguida, as regras são adicionadas incrementalmente, visando resolver problemas de *matching* como a classificação de falsos positivos e falsos negativos, tomando como suporte o processo de aquisição de conhecimento.

No nível estrutural, a abordagem KSMS utiliza o Similarity Flooding (MELNIK; GARCIA-MOLINA; RAHM, 2002), um *matcher* baseado em grafos utilizado há décadas na literatura. Para gerar os alinhamentos finais, o KSMS combina os resultados obtidos nos dois níveis (estrutural e de elemento), usando a função de agregação de média harmônica (*harmonic mean*).

Os autores destacam os seguintes benefícios desta abordagem: (i) as regras não são pré-definidas (fixas), podendo ser geradas com base em diferentes características dependendo das métricas de similaridade utilizadas; (ii) podem ser reutilizados operadores de *maching* no formato regras, ao invés de reutilizar resultados de *matching*; (iii) as regras podem ser reutilizadas por outros domínios; (iv) a abordagem não demanda uma atividade extensa de aquisição de conhecimento (como o YAM), visto que as regras são criadas apenas para classificar corretamente os casos que foram indicados como incorretos pelas árvores decisão; (v) dispensa a utilização de um limiar mínimo (*threshold*) para filtrar os resultados; (vi) dispensa a seleção do melhor método de *matching*, visto que a abordagem híbrida combina as similaridades produzidas por todos os métodos básicos.

Por outro lado, ressalta-se que, como as regras são estabelecidas conforme o uso, pode-se construir um conhecimento baseada nos interesses de uma determinada aplicação. Isto não é uma desvantagem neste contexto, mas questiona-se o reuso para outros domínios, visto que as regras podem ser definidas com base em alguma restrição específica. Além disso, os critérios de avaliação de qualidade de uma correspondência, podem variar de acordo com a aplicação, podendo um mesmo *matcher* obter desempenho insatisfatório quando aplicado a domínios diferentes. Por isso, combinar as similaridades produzidas por todos os métodos básicos pode não ser a melhor estratégia, visto que a utilização de *matchers* irrelevantes leva à penalização dos relevantes, tendo como consequência a geração de menos correspondências corretas após realização da combinação.

3.2.11 Abordagem Proposta por Sathiya, Geetha e Sugumaran (2017)

Sathiya, Geetha e Sugumaran (2017) propõem uma abordagem para escolher automaticamente o melhor conjunto de *matchers* a partir de um conjunto pré-definido, baseado

nas características de uma tarefa de alinhamento. Para viabilizar a escolha automática, a abordagem é capaz de aprender qual combinação de características do par de ontologias fornecido é melhor contemplada pelos *matchers* disponíveis. Para tal, dois novos algoritmos de aprendizado de máquina semi-supervisionados são propostos: Bottom-up Bootstraping (BUB) e Top-down Bootstraping (TDB), no intuito de aprender os padrões para cada conjunto de *matchers*. Cada padrão consiste de um conjunto de métricas de qualidade de ontologias, que representam as características linguísticas, estruturais e de instância requeridas para um conjunto particular de *matchers*.

Após o processo de aprendizado, os conjuntos de padrões são usados para automaticamente selecionar o melhor conjunto de *matchers* para uma determinada tarefa de alinhamento, baseado nas suas características. Além disso, visando uma maior acurácia na seleção de *matchers*, o trabalho propõe duas novas métricas de qualidade de ontologias, além das propostas por Cruz et al. (2012), voltadas a computar a porcentagem de conceitos que possui atributos e relacionamentos.

A abordagem proposta por Sathiya, Geetha e Sugumaran (2017) aproxima-se bastante do que foi proposto por Cruz et al. (2012), com o diferencial de aplicar algoritmos semi-supervisionados em alternativa aos supervisionados. Isto se deve à disponibilidade limitada de conjuntos de treinamento nesta área. Ao utilizar algoritmos de *bootstraping* semi-supervisionados, são requeridos menos dados rotulados e mais dados não-rotulados. Neste sentido, os autores utilizaram como dados de treinamento 233 conjuntos rotulados 11.589 não-rotulados, aumentando o escopo em comparação com o que é considerado por Cruz et al. (2012).

3.2.12 Discussão

Na seção anterior foi apresentada uma série de trabalhos que visam a recomendação de *matchers* para o alinhamento de ontologias. Os Quadros 2 e 3 apresentam uma análise comparativa dos trabalhos, levando em consideração os seguintes aspectos: objetivo, estratégia adotada para recomendação de *matchers*, quantidade de *matchers* selecionados, se há a exigência do fornecimento de correspondências de referência, se são considerados requisitos fornecidos pelo usuário e limitações.

Dentre os trabalhos listados, os primeiros (OntoMas e MOMA) introduziram o conceito de recomendação de *matchers* com base na compatibilidade entre eles e as características das ontologias fornecidas como entrada. No entanto, toda o levantamento de características é realizado de maneira manual, por intermédio da aplicação de questionários com especialistas. O uso de tal estratégia, além de ser demorada, não garante que o *matcher* apontado por um especialista como pertencente a uma determinada categoria irá, de fato, fornecer os melhores resultados quando comparados a uma ontologia que possua elementos compatíveis com ela. Neste aspecto, no *framework* Redoma propõe-se a extração de todas as características necessárias das ontologias automaticamente, com base em um conjunto de métricas de

qualidade. As informações inerentes aos *matchers* que influenciam na recomendação também são obtidas de maneira automática.

Outra estratégia utilizada pelos trabalhos apresentados, no sentido de dispensar o fornecimento de quaisquer alinhamentos de referência, é o cálculo prévio de métricas que indicariam a compatibilidade das ontologias fornecidas com uma determinada categoria de *matchers* (e.g., léxico, estrutural), como é o caso da ferramenta RIMOM. Para ontologias em larga-escala, tal cálculo teria de considerar percorrer todos os elementos, o que levaria a um custo computacional muito alto. Além disso, em um domínio de conhecimento com características peculiares, i.e., ontologias descritas com termos não conhecidos e contendo estruturas incomuns, não haveria garantia de que apenas o fato de se enquadrar numa categoria de *matcher* resultaria no fornecimento de melhores resultados.

Os trabalho propostos por Cruz et al. (2012) e Sathiya, Geetha e Sugumaran (2017) trazem como contribuição a definição de um conjunto de métricas de qualidade que podem ser utilizadas para avaliar quantitativamente um par de ontologias fornecido, em relação à predominância de elementos que se enquadrariam melhor em uma determinada categoria de *matcher* (e.g., léxicos, sintático, estruturais). Essas métricas são utilizadas no Redoma, em que também são propostas novas métricas para o agrupamento das existentes, denominadas medidas de afinidade. Elas servem de base para a extração de informações (alinhamentos gerados anteriormente) de bases de conhecimento utilizada para a recomendação de *matchers*.

Em relação aos critérios para avaliação dos *matchers*, a maior parte utiliza apenas como referência, primordialmente, as medidas de similaridade obtidas para uma determinada característica (e.g., SMB, ALMa). O YAM oferece a opção de maximizar uma das métricas de conformidade, tais como precisão, cobertura e Medida-F.

Já a quantidade de *matchers* que é fornecida como saída pelas abordagens, varia entre os que só fornecem um *matcher*, e outros que fornecem um conjunto, cabendo ao usuário definir a estratégia que será utilizada durante a geração dos alinhamentos (e.g., execução em paralelo, sequencial, combinação de medidas de similaridade). No caso do YAM, o conjunto é selecionado de maneira transparente ao usuário, visto que ele trata a seleção como a criação de um novo *matcher* sob-demanda. Na prática isso pode limitar a escolha do usuário em relação as estratégias de execução múltipla e de agregação de resultados existentes.

No capítulo a seguir, introduz-se a primeira contribuição desta tese que envolve a proposta de um algoritmo que utiliza requisitos do usuário para a geração de segmentos de ontologias, no intuito de tornar possível a divisão de uma tarefa de alinhamento em várias menores, possibilitando, tanto que diferentes *matchers* venham a desempenhar cada etapa (se for mais adequado às características do segmento), como também que o resultado obtido em uma, possa ser utilizado para aperfeiçoar o processo numa próxima fase.

Tabela 2 – Análise comparativa dos trabalhos de recomendação de ontologias (parte 1)

Abordagem	**Objetivo**	**Estratégia para recomendação de Matchers**	**Quantidade de matchers selecionados**	**Exige corresp. de referência?**	**Considera requisitos?**	**Limitações**
OntoMas	Auxiliar o usuário no processo de descoberta do melhor matcher dado um conjunto categorizado previamente	Atribuição das características de cada matcher mediante aplicação de questionários com especialistas - Modelo de decisão multicritérios baseado em regras adhoc.	N que se enquadrem nas características desejadas	Não	Não	Necessita que as características dos matchers sejam atribuídas previamente de maneira manual por usuários especialistas.
MOMA	Seleção dos matchers com base em regras derivadas das características de ontologias	Aplicação de regras preestabelecidas (fixas) - Consideração de características de matchers em vários níveis: entrada, abordagem, uso, processo, documentação e custo.	N que se enquadrarem nas características desejadas	Não	Parcialmente	Os requisitos são relacionados às características dos matchers e não ao objetivo da aplicação quanto ao processo. Exigindo do usuário o conhecimento das propriedades de cada matcher - Necessita a atribuição manual das características dos matchers - ausência de experimentos demonstrando efetividade da abordagem proposta.
SMB	Classificação com base em poucos exemplos, utilizando Boosting para melhorar classificadores fracos	Aprendizado de Máquina (Boosting)	Um par de cada nível. Na pratica, uma medida de similaridade.	Poucos exemplos	Não	Forte dependência de um único classificador de Boosting
RIMOM	Seleção de matchers com base em fatores de similaridade	Baseada em índices calculados previamente por meio da análise das características das ontologias	N definido por um limiar mínimo	Não	Não	Em ontologias de larga-escala, a análise das características se torna um gargalo - Não leva em consideração resultados concretos obtidos pelos matchers.
YAM	Geração de Matchers Dedicados com base na seleção de classificadores	Aprendizado de Máquina (Supervisionado)	Ilimitado (um matcher dedicado pode ser construído a partir da combinação de várias medidas de similaridade)	Sim	Parcialmente	Permite a escolha das métricas de avaliação de preferência, mas utilizam apenas para alimentar o conjunto de treinamento e não para definir o escopo do matching - O usuário é designado a selecionar um classificador apropriado, do contrário será utilizado um padrão disposto na base de conhecimento, que pode não representar a escolha mais adequada.
Abordagem Proposta por Peukert, Eberius e Rahm (2012)	Utilização de relação dos matchers com atributos de ontologias para definir adequabilidade	Abordagem adaptativa, combinando matchers básicos e métodos de agregação distintos de acordo com a tarefa de matching em questão.	1, representando a melhor combinação	Não	Não	Seleção inicial dos matchers é feita baseada em atributos básicos das ontologias - Algumas características dos matchers precisam ser atribuídas manualmente - Foram considerados apenas matchers básicos.

Tabela 3 – Análise comparativa dos trabalhos de recomendação de ontologias (parte 2)

Abordagem	**Objetivo**	**Estratégia para recomendação de Matchers**	**Quantidade de matchers selecionados**	**Exige corresp. de referência?**	**Considera requisitos?**	**Limitações**
Abordagem Prooposta por Cruz et al. (2012)	Quantificação de medidas de qualidade para ontologias para recomendação de matchers	Aprendizado de Máquina Supervisionado (KNN)	1 dentre os 5 matchers configurados disponibilizados	Sim	Não	O uso de abordagem supervisionada requer um conjunto de treinamento amplo, o que não foi considerado.
AML	Recomendação de alinhamentos com base no escopo da tarefa de matching	Categorização de matchers em básicos (lineares) e para larga-escala (polinomiais), selecionados de acordo com o escopo da tarefa de matching	1 dentre 4 matchers básicos disponibilidados. Novas implementações são admitidas.	Não	Não	Considera primordialmente o escopo da tarefa de matching como critério para seleção dos matchers, não se atendo a outros aspectos, como a compatibilidade com determinado tipo de característica predominante nas ontologias.
ALMa	Classificação com base em poucos exemplos	Aprendizado de Máquina (Aprendizado Ativo)	Utilização eficiente da participação do usuário, selecionando as correspondências mais significativas para avaliação em tempo de execução	Poucos exemplos	Não	Alta sensibilidade do classificador baseado em árvore de decisão pode não fornecer predições em um nível mais abrangente
KSMS	Seleção do melhor matcher com base em regras construídas ao longo do tempo	Híbrida (Aprendizado de Máquina + Recuperação de informação de bases de conhecimento)	Todos os matchers disponibilizados pela abordagem Hybrid-RDR	Não	Parcialmente	Permite a definição de regras com base na aquisição do conhecimento, porém, as utiliza apenas para a classificação das correspondências como corretas/incorretas e não na definição de métricas de avaliação - Regras são definidas globalmente e podem não ser aplicáveis a todos os domínios de conhecimento.
Abordagem Proposta por Sathiya, Geetha e Sugumaran (2017)	Seleção automática do melhor matcher com base as características das ontologias fornecidas	Aprendizado de Máquina Semi-supervisionado (Boosting)	1, dentre um conjunto de 3 matchers pré-configurados	Poucos exemplos	Não	Não suporta a definição de múltiplos critérios para a avaliação de qualidade do matcher.

4 USANDO REQUISITOS DA APLICAÇÃO PARA SEGMENTAÇÃO DE ONTOLOGIAS

Este capítulo apresenta a solução proposta neste trabalho para a extração de segmentos de ontologias a partir de um conjunto de requisitos da aplicação. Primeiramente, na Seção 4.1, discutem-se as motivações para o estudo e a definição do problema de pesquisa. Na Seção 4.2, apresentam-se as definições dos conceitos fundamentais para facilitar o entendimento da solução proposta, incluindo a caracterização formal dos requisitos da aplicação e segmentos de ontologias. Em seguida, na Seção 4.3, descrevem-se os detalhes sobre o algoritmo proposto para a geração de segmentos com base em requisitos. A Seção 4.4 descreve os experimentos realizados para a avaliação do algoritmo e apresenta os resultados obtidos. Por fim, a Seção 4.5, resume a contribuição exposta no capítulo e indica como o resultado pode ser utilizado para beneficiar a geração de alinhamentos.

4.1 MOTIVAÇÕES E DEFINIÇÃO DO PROBLEMA

Com a disseminação da Web Semântica, buscou-se expandir o potencial da Internet em relação a possibilitar que os computadores possam entender e navegar sobre os dados (recursos) publicados na Web, de maneira a estabelecer um ambiente análogo a um grande banco de dados global (VIRGILIO, 2012). Para viabilizar esse ambiente, uma alternativa inicial seria a construção de uma ontologia global, capaz de descrever todos os termos de interesse disponíveis na Internet. Porém, uma abordagem mais plausível, seria trabalhar com várias ontologias de domínio específico que, quando alinhadas a outras correlacionadas, formariam uma Web Semântica de relações inter-ontologias (HEATH; BIZER, 2011). Em ambos os casos, trabalha-se com um largo escopo de elementos, o que dá origem a novos desafios.

Um desses desafios é a demanda por alinhamentos de ontologias em larga-escala, i.e., quando é necessário comparar ontologias contendo milhares de classes e propriedades ou simplesmente quando deseja-se comparar um grande número de ontologias (RAHM, 2011). Um exemplo é o esforço recente para fazer buscas na Web no intuito de extrair entidades, relacionamentos e ontologias com o propósito de construir bases de conhecimento gerais, como a DBPedia (LEHMANN et al., 2015) e o Google Knowledge Graph (DONG et al., 2014). Nestes casos, encontrar e validar correspondências contendo tamanha quantidade de elementos exigiria um alto custo computacional. Uma possível estratégia para mitigar esse problema seria reduzir o espaço de busca, dividindo as ontologias em segmentos menores (SEIDENBERG et al., 2006; RAHM, 2011).

Este trabalho segue a premissa que integrar todo conhecimento da Web seria um esforço demasiadamente grande ou até mesmo inviável (DONG; SRIVASTAVA, 2015). Portanto, para viabilizar o uso da Web Semântica por usuários e aplicações no geral, seria preciso encontrar

uma forma de limitar o seu escopo, tendo em vista que no contexto da Web, as aplicações utilizam apenas informações relacionadas ao seu modelo de negócio, diferentemente do processo de integração de dados tradicional, que objetiva a construção de uma visão completa de todos os conceitos que compõem um conjunto de determinado domínio de conhecimento. Com isso, surge a necessidade de mecanismos que possibilitem a integração apenas de partes de ontologia de acordo com algum critério específico. Para tal, apresenta-se a seguinte solução: ao invés de tentar alinhar um grande número de ontologias por inteiro, cada aplicação pode fornecer um conjunto de requisitos que serão utilizados para extrair segmentos que contenham apenas os conceitos mais relevantes. A ideia é que os segmentos tenham tamanho suficiente para serem úteis ao processo de alinhamento de ontologias para aquela aplicação, mas que não sejam tão grandes a ponto de gerar problemas de escalabilidade.

Neste sentido, propõe-se o SOMA, um algoritmo de segmentação de ontologias que utiliza requisitos da aplicação como forma de adquirir informações relevantes para a geração dos segmentos. Isto possibilita uma forma personalizada de reduzir o escopo das ontologias para a obtenção de alinhamentos. Com o intuito de viabilizar a solução proposta, são abordados os seguintes problemas:

i. Como modelar os requisitos de uma aplicação de domínio específico para viabilizar a especificação dos elementos relevantes que serão incluídos nos segmentos de ontologias?

ii. Como extrair de um par de ontologias os conceitos mais relevantes, mas de maneira a minimizar a perda semântica nos segmentos gerados?

iii. Como utilizar segmentos de ontologias para possibilitar a combinação de diferentes *matchers* na geração de alinhamentos, de maneira a reduzir o escopo de busca e apresentar apenas correspondências relacionadas aos conceitos de interesse do usuário?

Para lidar com o problema (i), é formalizado o conceito de requisitos da aplicação, que são concebidos em dois tipos: Requisitos de Dado (RD) e Requisitos Operacionais (RO). RD indicam os conceitos mais relevantes que devem ser considerados ao gerar os segmentos de ontologia. RO são parâmetros de configuração que afetam como os segmentos são gerados. Em relação ao problema (ii), além de incluir nos segmentos elementos diretamente relacionados aos conceitos relevantes para a aplicação, busca-se preservar a semântica de cada conceito selecionado, através da consideração de um conjunto de princípios de conservação. Por fim, em relação ao problema (iii), propõe-se a utilização dos segmentos obtidos como entrada para um *framework* de recomendação de *matchers*, possibilitando uma maior eficiência na seleção e combinação dos *matchers* mais adequados, por intermédio da identificação dos que melhor contemplem as características dos pares de segmentos

fornecidos, ao invés de considerar as características das ontologias inteiras. As seções seguintes deste capítulo detalham as soluções propostas para os problemas (i) e (ii). A aplicação prática dos segmentos obtidos para a geração de alinhamentos, que aborda o problema (iii), é descrita em detalhes no Capítulo 5.

4.2 DEFINIÇÕES PRELIMINARES

Para a geração de segmentos de ontologias, faz-se necessário extrair um conjunto de elementos consistentes e que cumpram os requisitos definidos pela aplicação. Como forma de viabilizar esse processo, os elementos das ontologias são armazenados em um catálogo e posteriormente enriquecidos semanticamente, considerando, por exemplo, a adição de sinônimos a cada elemento através de consultas a um dicionário de termos (e.g., Wordnet).

Para tal, utiliza-se uma representação interna de ontologia, contendo os elementos importados de uma ontologia original, juntamente de um conjunto de anotações semânticas, que serão utilizadas para facilitar a busca e manipulação de conceitos.

Definição 4.1. No contexto deste trabalho, uma *Ontologia* (O) é definida pela tupla (C, R), onde $C = \{c_1, ..., c_n\}$ é um conjunto de conceitos e $R = \{r_1, ..., r_n\}$ é um conjunto de relações semânticas entre os conceitos. Cada $c_i \in C$ é uma tupla $c_i = (n, t, k)$, onde n é o *nome do conceito*, t é o tipo do conceito (e.g., classe, propriedade) e k é um *conjunto de palavras-chave* que pode alternativamente representar o conceito (e.g., sinônimos). Por outro lado, cada $r_i \in R$ é uma tupla $r_i = (rn, rt, c_s, c_t)$, onde rn é o nome da relação; rt é o tipo da relação (e.g., *domain*, *range*), $c_s \in C$ é o conceito que pertence ao domínio da relação (a fonte); e $c_t \in C$ é o conceito que pertence ao alcance (*range*) da relação (o alvo).

Essa representação de ontologias é utilizada neste trabalho para facilitar a extração de segmentos a partir da validação de requisitos. As próximas seções definem os conceitos relacionados a essas duas atividades.

4.2.1 Extração de Segmentos de Ontologias

Como introduzido no Capítulo 3, o estado da arte da geração de segmentos de ontologias apresenta duas principais estratégias: modularização e extração de segmentos. A modularização lida com a fragmentação de ontologias a partir da sua divisão sem admitir sobreposição, i.e., os segmentos se complementam até formar a ontologia inteira. Já a extração de segmentos representa a obtenção de porções de ontologias com base em requisitos do usuário, podendo haver sobreposição de conceitos presentes em diferentes segmentos. Esta última é a estratégia adotada neste trabalho. No entanto, como a presente proposta envolve a utilização dos segmentos no processo de alinhamento de ontologias, a extração de segmentos é construída a partir de duas ontologias (fonte e alvo) e não apenas uma (como ocorre tradicionalmente), conforme descrito a seguir.

Definição 4.2. Definem-se *Segmentos de Ontologias* pela tupla $(O_s.seg \subset O_s, O_t.seg \subset O_t)$ obtidos a partir dos elementos (R_d, R_o, O_s, O_t), onde R_d representa os requisitos de dado de uma aplicação APP, R_o representa os requisitos operacionais de uma aplicação APP, O_s é a ontologia fonte (*source*) e O_t é a ontologia alvo (*target*).

4.2.2 Requisitos da Aplicação

Nesta seção apresentam-se os conceitos de requisitos utilizados pelo algoritmo proposto visando a segmentação de ontologias.

O conceito de requisitos utilizado neste trabalho trata-se de uma especialização do conceito presente na literatura de engenharia de software, adaptando-o à área de integração de dados. Originalmente, de acordo com Sommerville (2010), requisitos da aplicação (ou de sistema) consistem de uma descrição abstrata das características, atributos ou parâmetros de qualidade que um sistema deve prover para ser útil aos seus usuários. Requisitos podem partir de características de alto nível de um serviço ou de restrições de um sistema até especificações matemáticas detalhadas. Neste contexto, requisitos funcionais são aqueles que descrevem funcionalidades ou serviços do sistema e requisitos não-funcionais os que definem propriedades do sistema ou restrições (e.g., tempo de resposta, confiabilidade).

Inspirado nesses conceitos, o modelo proposto neste trabalho considera dois tipos de requisitos: Requisitos de Dado (R_d) e Requisitos Operacionais (R_o).

Definição 4.3. Os *Requisitos de Dado* (R_d) relacionam-se aos requisitos funcionais de uma aplicação e são definidos como um conjunto de palavras-chave $\{k_1, k_2, \ldots, k_n\}$ que descrevem termos que representam conceitos relevantes para o usuário. De uma maneira geral, um determinado conceito pode estar relacionado a diferentes elementos de uma ontologia. Visando simplificar ao máximo a forma como o usuário define o conjunto de requisitos de dado, solicita-se ao usuário apenas a definição de um conjunto de termos que representam conceitos relevantes para a aplicação. Com isso, o algoritmo proposto irá se encarregar de obter automaticamente os elementos das ontologias que serão incluídos nos segmentos. Neste caso, por exemplo, uma aplicação do domínio de conferências que lida apenas com notificações aos autores, poderia definir como R_d os conceitos {author, email, conference}. Por outro lado, uma aplicação do mesmo domínio que vise integrar dados apenas sobre os artigos, poderia utilizar como requisito de dado os conceitos {paper, reviewer, title, abstract}. Assim, durante a geração do segmento, seriam incluídos os elementos que são relacionados aos definidos nos requisitos de dado.

Definição 4.4. Os *Requisitos Operacionais* (R_o) são relacionados aos requisitos não-funcionais da aplicação e refletem configurações em torno de como o processo de segmentação de ontologias será conduzido. Eles permitem que o usuário possa personalizar alguns aspectos, como o tamanho do segmento gerado e a tolerância para definir a similaridade entre conceitos. Por exemplo, caso a aplicação tenha limitações de recursos, poderá optar

por uma geração de segmentos menores, com menor tolerância nas métricas de similaridade entre as palavras-chave e os conceitos que serão considerados nos segmentos. Por outro lado, caso o intuito seja tentar reduzir ao máximo a intervenção do usuário para validar os segmentos gerados, pode-se optar por limitar o número de conceitos, mantendo apenas aqueles com maior probabilidade de serem correspondentes. Esse *trade-off* entre desempenho e qualidade é um parâmetro que pode ser afinado a partir da definição dos requisitos operacionais.

A Tabela 5 descreve os valores possíveis para os R_o: i) nível de extensão do segmento; ii) limiares de similaridade para comparação de palavras-chave e iii) limiares de similaridade para comparação de elementos de ontologias. Quando o nível de extensão do segmento é definido como "simples", segmentos consistem apenas dos conceitos cruciais aos definidos nos requisitos de dado. Alternativamente, quando o nível de extensão é definido como "expandido", os segmentos tendem a ser mais densamente populados, de maneira a conter um nível extra de relacionamento para os elementos selecionados. Os limiares de similaridade são valores no intervalo entre $[0, 1]$ que serão utilizados para definir o intervalo de tolerância do valor de similaridade para incluir um elemento como relacionado a um requisito de dado ou a um elemento de ontologia.

Ressalta-se que outros requisitos operacionais poderiam ainda ter sido considerados inicialmente, como, por exemplo, a restrição por um determinado tipo de relacionamento (e.g., isA), a limitação específica do número de elementos incluídos, entre outros. Porém, como o foco da geração de segmentos concentra-se na extração de conceitos relacionados aos requisitos de dado, optou-se por simplificar ao máximo a especificação dos requisitos operacionais, evitando demandar do usuário uma configuração mais complexa que requeira um conhecimento aprofundado da estrutura das ontologias envolvidas.

Requisitos Operacionais	**Valores possíveis**
Nível de Extensão do Segmento	simples, expandido
Limiares de Similaridade para comparação de palavras-chave	Números decimais no intervalo de $[0, 1]$
Limiares de Similaridade para comparação de conceitos de ontologias	Números decimais no intervalo de $[0, 1]$

Tabela 5 – Valoes possíveis para os Requisitos Operacionais

4.2.3 Princípios de conservação

Para que a extração de segmentos de ontologias seja consistente, é preciso ir além da recuperação isolada de cada conceito pela sua similaridade com determinado termo definido nos requisitos de dado. Selecionar conceitos individualmente pode acarretar em perda

semântica, já que muitas vezes o significado de uma informação depende de um conjunto de conceitos relacionados.

Para lidar com esse caso, poderia ser utilizada uma função recursiva para percorrer todos os conceitos em vários níveis de relacionamentos. No entanto, algum critério precisa ser estabelecido para determinar as fronteiras que irão permear a extração do segmento. Neste sentido, Lozano et al. (2014) introduz o conceito de Princípios de Conservação (*conservation principles*), que define restrições que podem ser seguidas para nortear a extração de segmentos a partir de um dado elemento. Esses princípios foram inspirados no conceito de meta-propriedades, utilizado, por exemplo, na ontologia Unified Foundational Ontology (UFO) (GUIZZARDI, 2005)), que provê uma classificação específica para descrever (de maneira abstrata) conceitos de ontologias, tais como identidade, rigidez, dependência existencial, dependência relacional, entre outros.

No entanto, a utilização de meta-propriedades requer que as ontologias sejam modeladas de maneira a contemplar os elementos que refletem essas informações. Neste caso, todos os conceitos das ontologias devem referenciar meta-propriedades definidas por uma ontologia base (*foundational ontology*), como a UFO, de maneira a estabelecer uma descrição semântica mais rica no nível de modelo.

Porém, como este trabalho visa a utilização de segmentos para melhorar a geração de alinhamentos, entende-se que seria demasiadamente restritivo impor como pré-requisito a existência dessas meta-propriedades nas ontologias. Apesar de haver um esforço recente neste sentido (e.g., (GUIZZARDI et al., 2015)), ainda não há uma adoção ampla de meta-propriedades em ontologias (PFAFF; NEUBIG; KRCMAR, 2018). Outro aspecto é que, além da UFO, existem outras meta-ontologias como, por exemplo, a Upper Merged Ontology (SUMO) (NILES; PEASE, 2001), a Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) (GANGEMI et al., 2002) e a General Formal Ontology (GFO) (ARP; SMITH; SPEAR, 2015). Neste caso, ao considerar meta-propriedades durante a geração de alinhamentos, seria necessário ficar restrito a uma alternativa específica ou fazer um alinhamento prévio entre as meta-ontologias utilizadas. Ressalta-se, no entanto, que a geração de alinhamentos nesse nível não seria trivial, já que um usuário especialista no seu domínio pode não ter o conhecimento suficiente para validar correspondências entre elementos de meta-ontologias.

Por tal motivo, visando manter a consistência da ontologia e evitar perda semântica, propõe-se princípios de conservação que utilizem apenas a estrutura comum da ontologia, sem requerer a existência de meta-propriedades. Logo, ao realizar a extração dos segmentos, além de considerar um determinado conceito alvo de comparação, incluem-se outros conceitos, de maneira a fazer cumprir os princípios de conservação listados a seguir:

- **Conservação de taxonomia:** Se o segmento $O.seg$ inclui o conceito c_1, ele deve incluir todos os conceitos que são hierarquicamente superiores a c_1.

- **Conservação de propriedades de dado:** Se um segmento $O.seg$ inclui um determinado conceito c_1, as propriedades de dado de c_1 (i.e., os relacionamentos para tipos literais) também devem ser incluídos no segmento.
- **Conservação de propriedades de objeto:** Se um segmento $O.seg$ inclui um determinado conceito c_1, as propriedades de objeto de c_1 (i.e., os relacionamento entre c_1 e outro conceito c_2) devem ser preservados, de maneira que ambos devem ser incluídos no segmento, mas sem haver propagação automática para conceitos relacionados a c_2.

A aplicação desses princípios como parte do fluxo de trabalho do algoritmo proposto é descrito em detalhes na próxima seção.

4.3 ALGORITMO PARA GERAÇÃO DE SEGMENTOS COM BASE EM REQUISITOS

Nesta seção apresentam-se os detalhes sobre o SOMA (*Segmentation of Ontologies for Matching Applications*), um algoritmo para geração de segmentos de ontologias com base em requisitos. Antes de apresentar o fluxo principal da execução do algoritmo proposto, são definidas algumas funções que são usadas no algoritmo.

4.3.1 Funções para segmentação de ontologias

Como parte da execução do SOMA, são utilizadas algumas funções envolvendo a manipulação de ontologias. São introduzidas as funções CBR, para recuperação de conceitos a partir de um determinado relacionamento com outro conceito e Sim para o cálculo de similaridade entre palavras-chave e conceitos. Nas definições a seguir, considere R um conjunto de relacionamentos, C um conjunto de conceitos e K um conjunto de palavras-chave.

4.3.1.1 Recuperação de conceitos por tipo de relacionamento

Definição 4.5. A função CBR (*Concept By Relationship*) recebe um relacionamento $r \in R$, um conceito fonte $c_1 \in C$ e fornece o conceito alvo $c_2 \in C$ que possui o relacionamento r com c_1.

$$CBR : R \times C \rightarrow C$$

Por exemplo, $CBR(subclass, c_1)$ fornece um conjunto de conceitos C que possuem relacionamento do tipo $subclass$ com o conceito c_1.

4.3.1.2 Similaridade entre palavras-chave e conceitos

Definição 4.6. A função Sim compara uma determinada palavra-chave $k \in K$ a um determinado conceito $c \in C$, resultando em um valor $\alpha \in [0, 1]$, obtido por uma medida

de similaridade $m \in M$.

$$Sim : K \times C \times M \rightarrow \alpha$$

Por exemplo, se em $Sim(k_1, c_1, =)$, k_1 é a palavra-chave pessoa e c_1 possui como nome ($c.n$) ou contém como palavras-chave ($c.k$) o valor pessoa, o retorno da função Sim será o valor 1, visto que a medida de similaridade escolhida foi a igualdade (=). Outras métricas conhecidas, como, Jaro-Winkler, Levenshtein (Edit Distance), ou mesmo uma combinação delas pode ser usada para definir a similaridade entre os termos comparados.

Especificamente se tratando do protótipo utilizado na avaliação do algoritmo, foi adotado como implementação da função de similaridade (Sim), o algoritmo S-Match-Wordnet[1]. Esse algoritmo se baseia nas informações semânticas obtidas do Wordnet para identificar se um par de termos fornecidos estão relacionados. A opção por esta ferramenta é justificada pela sua capacidade de fornecer resultados sintetizados de maneira rápida (BELLA; GIUNCHIGLIA; MCNEILL, 2017), o que é especialmente útil durante a etapa de geração de segmentos, considerando que trata-se de um alinhamento preliminar entre um subconjunto de elementos das ontologias.

4.3.2 Fluxo de trabalho do algoritmo

SOMA é um algoritmo que recebe como entrada os seguintes parâmetros: duas ontologias O_s e O_t que deseja-se comparar, um conjunto de requisitos de dado R_d e um conjunto de requisitos operacionais R_o. O retorno gerado é um par $(O_s.seg, O_t.seg)$, referente aos segmentos gerados para as ontologias fornecidas como entrada (O_s, O_t).

A Figura 8 apresenta as etapas do algoritmo SOMA. As setas sólidas indicam o fluxo principal do algoritmo, enquanto que as setas pontilhadas indicam o fluxo interno referente à cada etapa.

Seguindo o fluxo principal do algoritmo, na primeira etapa, as ontologias fornecidas são convertidas em uma representação interna (vide Definição 4.1) e são armazenadas em uma base de dados local denominada catálogo de ontologias. Em seguida, gera-se o segmento para a ontologia fonte (O_s), por meio da extração de elementos relacionados aos requisitos da aplicação. Posteriormente, é gerado o segmento para a ontologia alvo (O_t), extraindo dela os elementos relacionados à ontologia fonte. Nas seções seguintes, são descritas em detalhes cada uma dessas etapas que compreendem o funcionamento do SOMA.

4.3.3 Importando ontologias no catálogo e coletando sinônimos

A primeira etapa do algoritmo envolve a importação, o pré-processamento e o armazenamento de ontologias. Após receber a entrada do usuário, o algoritmo percorre as ontologias fornecidas no formato de arquivos como OWL ou RDF e os converte em uma representação interna, estruturada como grafo rotulado. Durante esse processo, o grafo é enriquecido

[1] Código-fonte do S-Match-wordnet disponível em: <https://github.com/s-match/s-match-wordnet>

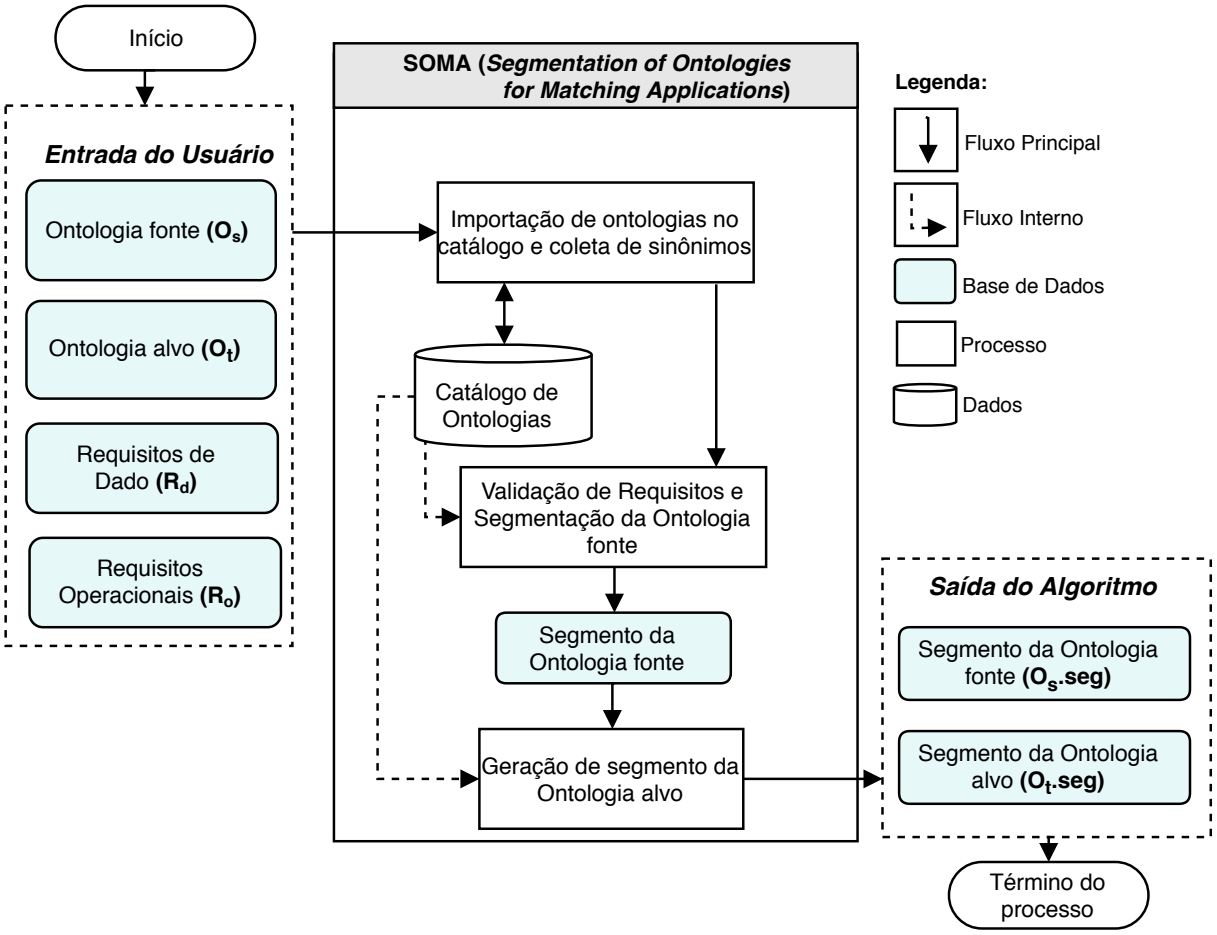


Figura 8 – Etapas de execução do algoritmo SOMA.

com a adição de sinônimos para cada nó (obtidos do Wordnet[2]). Para tal, os rótulos e nomes de cada classe ou propriedade são fornecidos como entrada de uma consulta ao Wordnet. Os sinônimos obtidos são então adicionados ao grafo como elementos relacionados à classe/propriedade comparada.

Para auxiliar na busca e possibilitar o armazenamento de ontologias para usos posteriores, o SOMA considera o uso de um *Catálogo de Ontologias*, que é um banco de dados de grafos que armazena metadados sobre as ontologias importadas pelo algoritmo, além de seus respectivos conceitos e relacionamentos. A implementação do protótipo do catálogo de ontologias utilizado nos experimentos é descrita em detalhes no Apêndice A.

A estrutura do grafo utilizado como representação interna para descrever ontologias é exemplificada na Figura 9. Os nós pintados em preto representam conceitos do tipo classe. As setas que ligam os nós indicam relacionamentos e o rótulo interno na seta indica o tipo do relacionamento. Os nós em cinza indicam propriedades de objeto, i.e., o relacionamento entre duas classes, (e.g., {Person, hasConflictOfInterest, Document} - uma pessoa com conflito de interesse em relação a um documento). Os nós em branco com borda sólida representam propriedades de dado, i.e., o relacionamento entre uma classe e

[2] Wordnet acessível pelo endereço: <https://wordnet.princeton.edu>

um tipo literal, (e.g., {Paper, title, xsd:string} - um artigo e um valor literal que representa o título). Por fim, os nós em branco com bordas tracejadas indicam as palavras-chave que podem ser utilizadas para identificar o elemento relacionado.

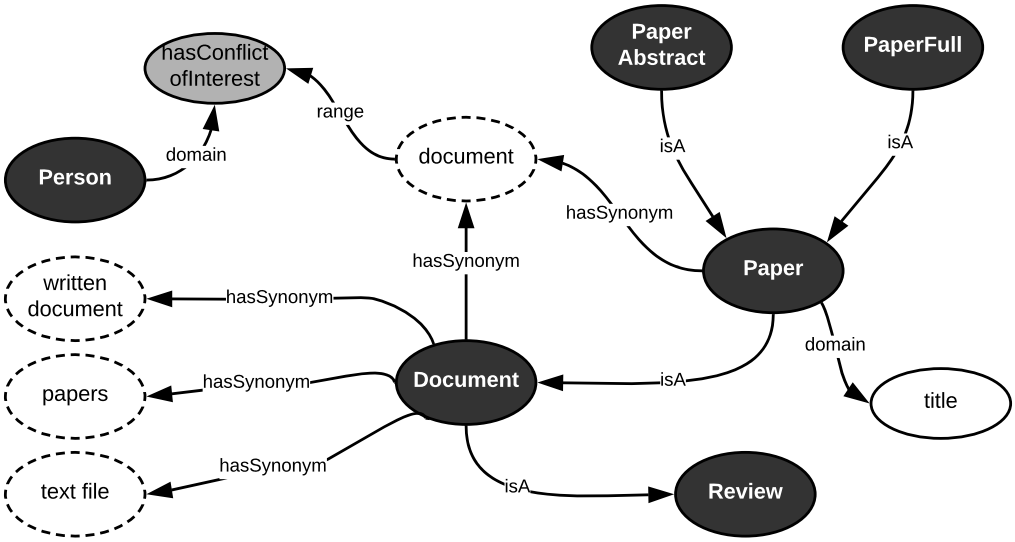


Figura 9 – Exemplo da representação de grafos adotada neste trabalho.

Uma vez que as ontologias envolvidas no processo foram importadas e preparadas para as demais etapas, elas serão percorridas com o intuito de analisar a relevância de cada conceito em relação aos requisitos que foram definidos. Caso a ontologia já tenha sido utilizada em um alinhamento anterior, utiliza-se a referência da ontologia já importada. Neste contexto, a geração de segmentos é feita em duas etapas. Primeiro gera-se o segmento para a ontologia fonte com base nos requisitos. Em seguida, utilizando como base o primeiro segmento obtido, gera-se o da ontologia alvo. Esse processo é detalhado nas próximas seções.

4.3.4 Validação de Requisitos e Segmentação da Ontologia fonte

A segmentação da ontologia fonte ocorre por meio da extração de conceitos que são relacionados aos termos que foram definidos nos Requisitos de Dado fornecidos. Para tal, a função $segO_f$ (Algoritmo 1) recebe como entrada, além da Ontologia Fonte O_s, um conjunto de Requisitos de Dado (R_d) e um conjunto de Requisitos Operacionais (R_o). O retorno é um segmento $segO_f$ contendo apenas os conceitos (i.e., classes, propriedades, relacionamentos) que são compatíveis com os R_d. Já o impacto dos R_o no processo, dependendo dos requisitos definidos, afetará a quantidade de elementos que será produzida como saída.

Para a geração do segmento, percorre-se cada termo t definido nos requisitos de dado R_d e buscam-se os conceitos que sejam similares. Para isso, utiliza-se a função Sim, que fornece um valor entre $[0, 1]$ indicando a similaridade entre t e os nomes dos conceitos

Algoritmo 1 Algoritmo para segmentação da ontologia fonte

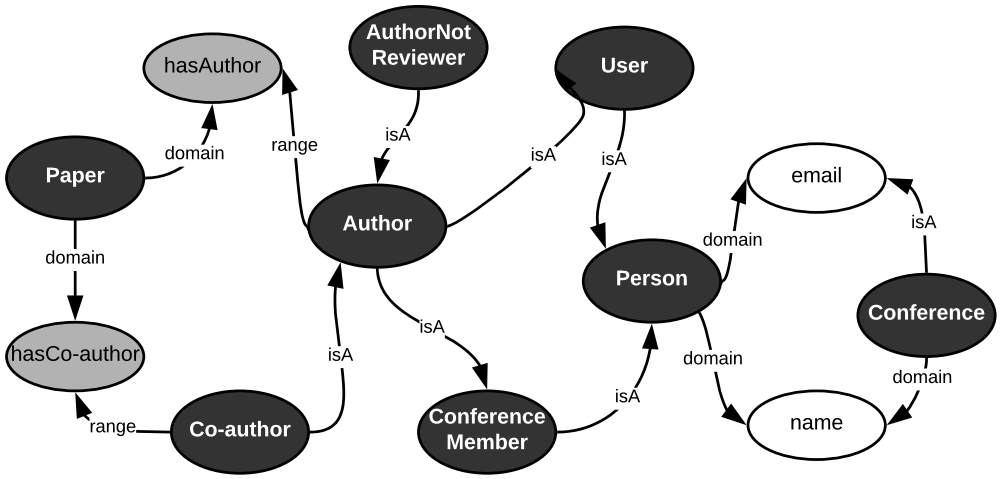
1: **function** $segO_s(O_s,\ R_d,\ R_o)$
▷ O_s corresponde ao grafo contendo os elementos da ontologia fonte;
▷ R_d (Requisitos de Dado) correspondem à lista de palavras-chave que representam os conceitos relevantes para a aplicação;
▷ R_o (Requisitos Operacionais) correspondem à lista de parâmetros de configuração usados para gerar o segmento;
2: $O_s.seg \leftarrow \{\}$ ▷ Inicializa o subgrafo $O_s.seg$ representando o segmento
3: **for all** $k_i \in R_d$ **do** ▷ Percorre cada palavra-chave k_i definida nos requisitos de dado R_d
4: **for all** $c_i \in O_s$ **do** ▷ Percorre cada conceito c_i da ontologia O_s
5: **if** $Sim(k_i, c_i) \cap R_o.limiaresSimilaridadePalavrasChave$ AND $c_i \notin O_s.seg$ **then**
6: $selectedConcept \leftarrow c_i$
7: $O_s.seg \leftarrow O_s.seg \cup selectedConcept$ ▷ Adiciona conceito c_i ao segmento
8: $superClasses \leftarrow CBR(superClass, c_i)$
9: $O_s.seg \leftarrow O_s.seg \cup superClasses$ ▷ Inclui as superclasses de c_i para cumprir o princípio de conservação de taxonomia
10: $dataProperties \leftarrow CBR(dataProperty, c_i)$
11: $O_s.seg \leftarrow O_s.seg \cup dataProperties$ ▷ Inclui as propriedades relacionadas a c_i para cumprir o princípio de conservação de propriedades de dado.
12: $objectProperties \leftarrow CBR(objectProperty, c_i)$
13: $O_s.seg \leftarrow O_s.seg \cup objectProperties$ ▷ Inclui as propriedades relacionadas a c_i para cumprir o princípio de conservação de propriedades de objeto.
14: **if** $RO.nivelExtensao ==$ EXPANDIDO **then**
15: $subclasses \leftarrow c_i \cup CBR(subclasses, c_i)$
16: $O_s.seg \leftarrow O_s.seg \cup subclasses$ ▷ No caso do requisito operacional nível de extensão como expandido, inclui as subclasses diretas de c_i.
17: **end if**
18: **end if**
19: **end for**
20: **end for**
21: **return** $O_s.seg$
22: **end function**

ou entre t e as palavras-chave adicionadas ao conceito durante a importação. O requisito operacional *valores de limiares de similaridade* (Seção 4.2.2) é utilizado para definir um intervalo de tolerância do que será considerado similar.

Durante a execução do algoritmo, além dos conceitos relacionados aos requisitos, são incluídos no segmento também os elementos que façam cumprir os princípios de conservação definidos na Seção 4.2.3. Para isso, é utilizada a função CBR, que recupera conceitos

relacionados a partir de um tipo (e.g., superclasses, propriedades de dado, propriedades de objeto). Por fim, caso o requisito operacional *nível de extensão* seja definido com o valor "EXPANDIDO", o segmento também irá incluir as subclasses ligadas diretamente aos elementos selecionados.

(a) Segmento para ontologia fonte considerando as palavras-chave k_1



(b) Segmento para ontologia fonte considerando as palavras-chave k_2

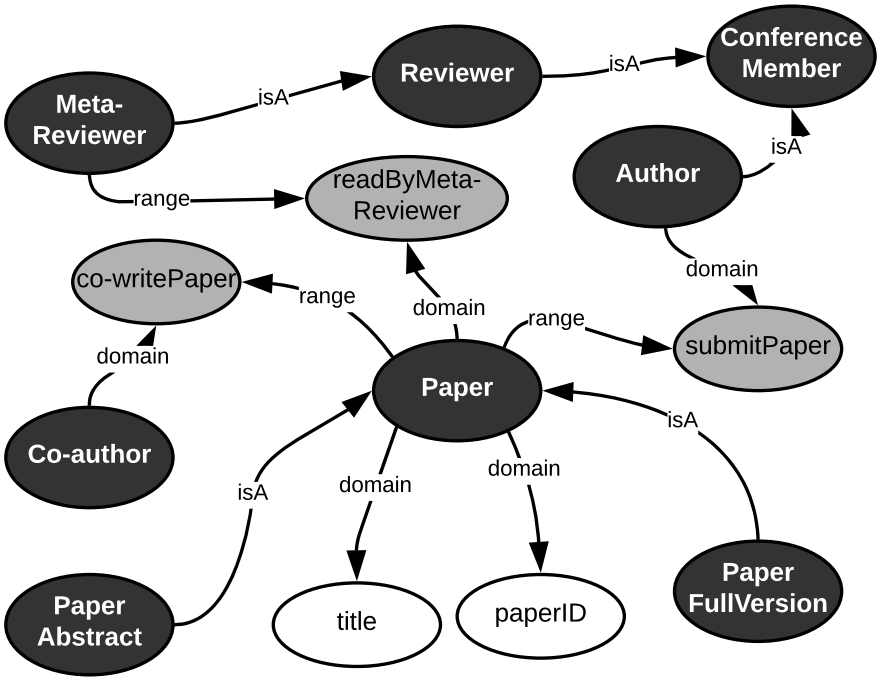


Figura 10 – Exemplos de segmentos gerados para a ontologia Cmt com base nos conjuntos de palavras-chave (person, name, author, email, university) e (paper, reviewer, title, abstract).

A Figura 10 mostra dois exemplos de segmentos gerados para a ontologia Cmt[3].

[3] Ontologia Cmt disponível no endereço: <http://oaei.ontologymatching.org/2017/conference/data/cmt.

No primeiro, ilustra-se como o segmento é gerado a partir das palavras-chave k_1 = {person, name author, email, university}. Na segunda, mostra-se como o segmento é gerado para as palavras-chave k_2 = {paper, reviewer, title, abstract}. Para ambos, foram utilizados como requisitos operacionais o nível de extensão "simples" e como limiares de similaridade o intervalo $[0, 7, 1]$.

Note que para o primeiro segmento (Figura 10a), o algoritmo SOMA inclui no segmento as classes Person', Author e seus relacionamentos, visto que a ontologia Cmt contém elementos relacionados a esses conceitos. O mesmo se aplica às propriedades email e name'. No entanto, como a ontologia Cmt não possui elementos relacionados a university, observa-se a ausência desse conceito. No segundo segmento (Figura 10b), o SOMA extrai da ontologia original as classes Paper, Reviewer e a propriedade de dados title, visto que correspondem exatamente às palavras-chave utilizadas. Já a classe Paper abstract, mesmo não tendo uma correspondência exata também é considerada.

4.3.5 Geração dos segmentos para a ontologia alvo

Após obter os segmentos para a ontologia fonte, é gerado o segmento para a ontologia alvo (O_t). Para tal, selecionam-se os conceitos de O_t com base na similaridade com os que foram considerados para o segmento da ontologia fonte ($O_s.seg$). Antes de chegar até esta solução, foram experimentadas outras estratégias, como a geração do segmento da ontologia alvo pela mesma estratégia utilizada para gerar o da ontologia fonte. No entanto, como uma ontologia é uma estrutura mais formal quando comparada a um conjunto de palavras-chave, foi possível alcançar melhores resultados pela segmentação da ontologia alvo a partir da comparação entre os elementos de ontologias, conforme descrito no Algoritmo 2.

Logo, para a geração do segmento para a ontologia alvo (O_t), primeiramente percorrem-se as palavras-chave associadas ($c_i.k_i$) aos conceitos presentes no segmento gerado para a ontologia fonte ($O_s.seg$). Em seguida, percorrem-se os conceitos da ontologia alvo ($O_t.c_j$), cuja similaridade é comparada com cada palavra-chave ($c_i.k_i$). Caso o valor obtido nesta comparação esteja dentro dos limiares de similaridade definidos como requisito operacional, o conceito (c_j) é selecionado para compor o segmento da ontologia alvo ($O_t.seg$).

Da mesma forma como ocorre com a geração dos segmentos para a ontologia fonte, além do conceito (c_j), são incluídos os conceitos relacionados que façam cumprir os princípios de conservação definidos na Seção 4.2.3. Para tal, utiliza-se a função CBR para buscar as superclasses, propriedades de dado e propriedades de objeto. Além disso, caso o o requisito operacional *nível de extensão* seja atribuído como "EXPANDIDO", serão consideradas também as subclasses de c_j.

A Figura 11 ilustra dois exemplos de segmentos para uma ontologia alvo. No primeiro (Figura 11a), o segmento apresentado é originado da comparação entre o segmento da ontologia fonte gerada a partir das palavras-chave k_1 (definidas na Seção 4.3.5). A ontologia

owl>

Algoritmo 2 Algoritmo para segmentação da ontologia alvo (O_t)

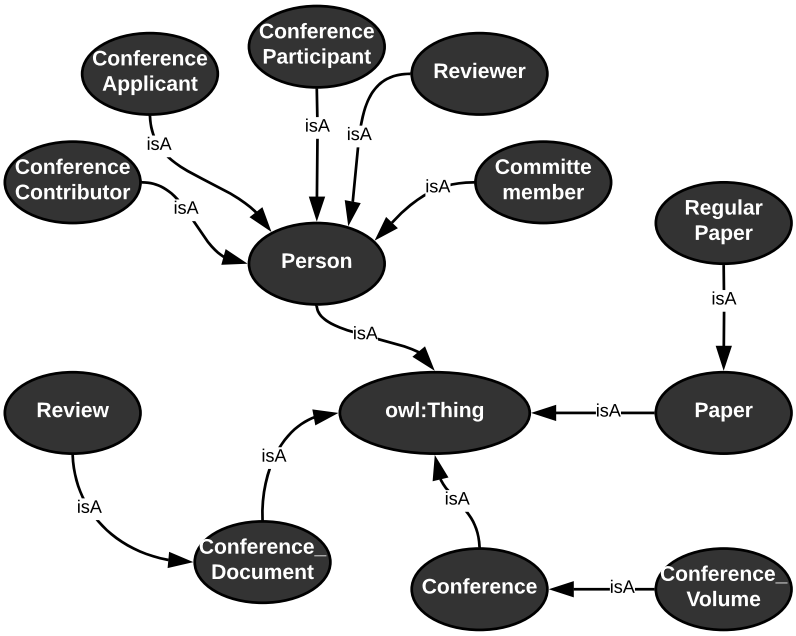
function $segO_t(O_t, O_s.seg, R_o)$
▷ O_t corresponde ao grafo representando elementos da ontologia alvo;
▷ $O_s.seg$ corresponde ao segmento da ontologia fonte obtido na etapa anterior;
▷ R_o (Requisitos Operacionais) corresponde à lista de parâmetros de configuração que são usados para gerar o segmento;
$O_t.seg \leftarrow \{\}$ ▷ Inicializa segmento da ontologia alvo
for all $c_i.k_i \in O_s.seg$ **do** ▷ Percorre cada palavra-chave relacionada a um conceito ($c_i.k_i$) em $O_s.seg$
for all $c_j \in O_t$ **do** ▷ Percorre cada elemento c_j em O_t
if $Sim(c_i.k_i, c_j) \cap RO.limiaresSimilaridadeConceitosOntologias$ AND $c_j \notin O_t.seg$ **then**
$selectedConcept \leftarrow c_j$
$O_t.seg \leftarrow O_t.seg \cup selectedConcept$ ▷ Adiciona conceito c_j ao segmento
$superClasses \leftarrow CBR(superClass, c_j)$
$O_s.seg \leftarrow O_t.seg \cup superClasses$ ▷ Inclui as superclasses do c_j para cumprir o princípio de conservação de taxonomia
$dataProperties \leftarrow CBR(dataProperty, c_j)$
$O_t.seg \leftarrow O_t.seg \cup dataProperties$ ▷ Inclui as propriedades relacionadas a c_j para cumprir o princípio de conservação de propriedades de dado.
$objectProperties \leftarrow CBR(objectProperty, c_j)$
$O_t.seg \leftarrow O_t.seg \cup objectProperties$ ▷ Inclui as propriedades relacionadas a c_j para cumprir o princípio de conservação de propriedades de objeto.
if $RO.nivelExtensao ==$ EXPANDIDO **then**
$subclasses \leftarrow c_j \cup CBR(subclasses, c_j)$
$O_t.seg \leftarrow O_t.seg \cup subclasses$ ▷ No caso do requisito operacional nível de extensão como expandido, inclui as subclasses diretas de c_j.
end if
end if
end for
end for
return $O_t.seg$
end function

considerada como alvo é a SofSem[4]. Essa mesma ontologia alvo é considerada no segundo caso (Figura 11b), que diferencia-se do primeiro apenas por considerar como entrada o segmento gerado para a ontologia fonte com base nas palavras-chave k_2.

Note que a apesar de a ontologia Sofsem não possuir uma classe com o nome Author, o SOMA incluiu classes no segmento que indicam algum tipo de correspondência com um autor, tais como Conference Contributor, Conference Participant e Conference Applicant.

[4] Ontologia Conference disponível no endereço: <http://oaei.ontologymatching.org/2017/conference/data/Conference.owl>

(a) Segmento para Ontologia alvo considerando as palavras-chave palavras-chave person, name, author, email, university.



(b) Segmento para Ontologia alvo considerando as palavras-chave (paper, reviewer, title, abstract).

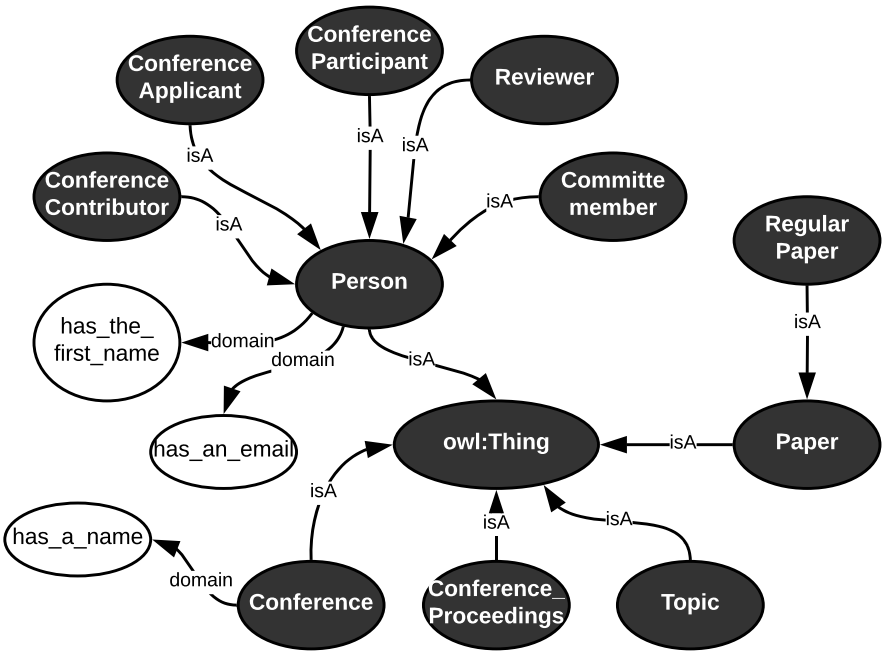


Figura 11 – Trecho do segmento gerados para a ontologia Sofsem, considerando como entrada o segmento da ontologia fonte ilustrado na Seção 4.3.4

Reforça-se que o intuito desta etapa de geração de segmentos é extrair um conjunto conciso de elementos relacionados aos conceitos definidos nos requisitos fornecidos pelo usuário e

não gerar alinhamentos completos entre as ontologias comparadas.

Portanto, uma vez gerados, ambos os segmentos estarão disponíveis no catálogo de ontologias e podem ser posteriormente exportados no formato OWL. Eventualmente, esses arquivos podem ser utilizados como entrada para algum sistema de alinhamento de ontologias (*matcher*). Inclusive, esta foi a estratégia usada nos experimentos realizados, como mostra a Seção 4.4. Neste caso, a premissa é que um *matcher* irá oferecer estratégias mais sofisticadas para identificar correspondências entre elementos presentes nos experimentos. Além disso, como evidenciado pelos experimentos, observou-se que a geração de alinhamentos com base nos segmentos possibilitou uma melhoria na qualidade dos resultados obtidos, através da combinação de diferentes *matchers* para realizar o alinhamento de ontologias com base em segmentos.

4.4 EXPERIMENTOS E RESULTADOS

Nesta seção são apresentados os experimentos realizados para a avaliação do SOMA. Para viabilizar a experimentação, foi construído um protótipo que implementa o algoritmo SOMA. Os detalhes de implementação e as formas de acesso a cada serviço disponibilizado pelo protótipo são descritos no Apêndice A. As seções seguintes abordam aspectos inerentes aos experimentos realizados, incluindo configurações utilizadas (e.g., os conjuntos de dados considerados), objetivos, metodologia e a discussão dos resultados obtidos.

4.4.1 Objetivo

O objetivo deste experimento é mostrar que a segmentação de ontologias com base em requisitos de usuário, introduzida pelo algoritmo SOMA, possibilita a geração eficiente de segmentos, de maneira que garanta a conservação semântica dos elementos extraídos e possibilite a sua utilização na execução de alinhamentos em várias etapas, comprovando as hipóteses **H1.1** e **H1.2** descritas abaixo.

H1.1 A segmentação de ontologias usando o SOMA garante a conservação semântica dos elementos extraídos.

H1.2 A utilização de segmentos de ontologias para a geração de alinhamentos em várias etapas provê resultados de melhor qualidade em comparação à geração de alinhamentos entre ontologias completas.

4.4.2 Configuração do Ambiente para Execução de Experimentos

Para a preparação do ambiente dos experimentos, o primeiro passo foi inicializar o Catálogo de Ontologias e o Catálogo de *Matchers* com um número suficiente de dados para dar suporte aos testes. Para a inicialização do Catálogo de Ontologias, foram importadas as ontologias do conjunto de dados de conferências fornecidas pela OAEI. Os *matchers*

existente utilizados para a comparação de resultados na geração de alinhamentos foram o LogMap, COMA e AML, utilizando configurações padrões recomendadas de cada um. Essa escolha é justificada pelo fato de serem ferramentas com boa participação nas campanhas da OAEI e que disponibilizam o código-fonte abertamente, o que facilitou a integração com o catálogo de *matchers*.

Para facilitar a execução de experimentos, todo o processo de configuração do ambiente foi automatizado, de maneira a possibilitar que o ambiente possa ser reiniciado quantas vezes for necessário. Para tal automatização, foram desenvolvidos *scripts* na linguagem Python [5]. Tal escolha se motivou por conta da leveza e simplicidade da linguagem, como também pela integração com bibliotecas que são úteis para a manipulação de dados e construção de gráficos (e.g., NumPy, pandas), o que é essencial para a coleta de resultados dos experimentos. Além disso, apesar dos protótipos terem sido desenvolvidos na linguagem Java, o acesso a todos os serviços oferecidos é feito mediante requisições HTTP REST, que são facilmente acessadas a partir da biblioteca *requests*, nativa ao Python.

Conjuntos de dados (Ontologias)

A Tabela 6 apresenta os detalhes dos conjuntos de dados considerados nos experimentos e os respectivos números de classes, propriedades de dado e propriedades de objeto. Apesar de a trilha de conferências da OAEI conter um total de 16 ontologias, foram selecionadas apenas as ontologias que contêm alinhamentos de referência (*gold standards*)[6].

Os alinhamentos de referência disponibilizados na trilha de conferências da OAEI descrevem as correspondências esperadas para um total de 22 tarefas de alinhamento entre as ontologias dispostas na Tabela 6. Para cada alinhamento, é definida uma ontologia fonte (O_s) e uma ontologia alvo (O_t), sendo a geração dos alinhamentos sempre baseada no sentido $O_s \rightarrow O_t$.

Ontologia	**Classes**	**Propriedades de Dado**	**Propriedades de Objeto**
Cmt	36	10	49
ConfTool	38	23	13
Edas	104	20	30
Ekaw	74	0	33
Iasted	140	3	38
Sigkdd	49	11	17

Tabela 6 – Informação detalhada das ontologias da trilha de conferências da OAEI utilizadas nos experimentos

Visando incrementar o número de ontologias utilizadas nos experimentos, foram gerados vários segmentos a partir do algoritmo SOMA, seguindo os seguintes cenários:

[5] Documentação da linguagem Python disponível em: <https://docs.python.org/3/>

[6] Alinhamentos de referência disponíveis em: <http://oaei.ontologymatching.org/2017/conference/data/reference-alignment.zip>

- **Cenário 1 (C1):** Geração de alinhamentos entre as ontologias completas;
- **Cenário 2 (C2):** Geração de alinhamentos entre segmentos gerados com base nos requisitos de dados: paper, reviewer, title, abstract;
- **Cenário 3 (C3):** Geração de alinhamentos entre segmentos gerados com base nos requisitos de dados: person, name, author, email, university.

Configurações de Hardware

A seguinte configuração de hardware foi utilizada durante a execução dos experimentos:

- Processador: Intel Core i5-3210M
- Velocidade da CUP: 2.50GHz x 2 core
- Capacidade de memória RAM: 16GB

Parâmetros de Configuração do Algoritmo

Para fins de facilitar a comparação dos resultados, foi utilizado um único conjunto de requisitos operacionais para os experimentos, definidos a seguir:

- Nível de Extensão do Segmento = EXPANDIDO
- Limiares de similaridade = [0,6, 1]

Na seção que segue, descrevem-se em detalhes os experimentos realizados, incluindo a discussão dos resultados obtidos em cada um.

4.4.3 Metodologia

Para avaliar a qualidade dos segmentos gerados pelo SOMA, foi utilizado como referência um conjunto de segmentos de referência construídos através da consulta a um especialista no domínio. Neste caso, considerando os pares de ontologias considerados nos experimentos, foram especificados segmentos de referência para cada uma das duas etapas do algoritmo: geração do segmento para a ontologia fonte (*source*) e alvo (*target*). Para determinar a porcentagem de acerto, os elementos incluídos pelos segmentos gerados pelo SOMA são comparados aos elementos que compõem os segmentos de referência.

Para avaliar a utilização de segmentos durante o alinhamento de ontologias ontologias, são considerados três cenários distintos. No primeiro, utilizam-se as ontologias completas para a geração de alinhamentos, enquanto que nos demais utilizam-se segmentos gerados pelo SOMA, com base nos requisitos de dados descritos a seguir. Para a execução dos alinhamentos, consideram-se a versão padrão do *matcher* LogMap e as versões do AML e COMA com os parâmetros de configuração que obtiveram o melhor desempenho. Para

a avaliação dos resultados, são consideradas as seguintes métricas de qualidade: tempo de execução, medida de precisão (*precision*, medida de cobertura (*recall*) e Medida-F. Para avaliar cada cenário, são considerados alinhamentos de referência (gold-standard) específicos[7].

4.4.4 Análise dos Resultados

Nesta seção são discutidos os resultados obtidos durante a execução dos experimentos. A Tabela 7 mostra um resumo das características dos segmentos gerados e a porcentagem de acerto quando comparado a um conjunto de segmentos de referência.

A Tabela 7 mostra que os segmentos gerados contêm boa parte dos elementos das ontologias originais (classes, propriedades de objeto, propriedades de classe), apesar de que em menor quantidade, o que era esperado, visto que foram incluídos neste caso apenas os elementos referentes aos requisitos definidos para o cenário C2. Em relação à comparação com os segmentos de referência, nota-se que foi obtido um acerto médio de 86,5%, o que se mostra um resultado satisfatório, considerando que os segmentos são gerados através de um processo automático, baseado em um conjunto de palavras-chave para a geração do segmento da ontologia fonte e no próprio segmento resultante para a geração do referente à ontologia alvo.

A Figura 12a ilustra, por meio de um gráfico, o tempo de execução total para a geração dos alinhamentos em cada cenário (C1, C2, C3). Comparando os resultados obtidos na comparação entre as ontologias completas (C1) com os resultados obtidos para a geração de alinhamentos entre os segmentos gerados pelo SOMA (C2, C3), conforme esperado, nota-se que no segundo grupo há uma significativa redução do tempo de execução para todos os *matchers*.

A Figura 12b apresenta um gráfico contendo a comparação das medidas de precisão obtidas em cada cenário. Nota-se que foi possível observar um aumento na precisão na maioria dos cenários em que foram considerados os segmentos gerados pelo SOMA, especialmente quando os que envolvem os *matchers* COMA e AML. Isto demonstra que não houve perda de qualidade com a redução do escopo de busca, no sentido de que os alinhamentos continuaram a informar as correspondências corretas relacionadas aos conceitos informados pelo usuário.

A Figura 12c ilustra a comparação do resultado dos cenários utilizando como parâmetro as medidas de cobertura (*recall*) obtidas. Neste caso, observa-se que considerando o *matcher* AML, a qualidade foi mantida no cenário C3, enquanto que o C2 obteve valor inferior ao C1. No COMA, ambos os cenários foram melhores do que o que não considera os segmentos e no LogMap e os três cenários obtiveram resultados praticamente idênticos.

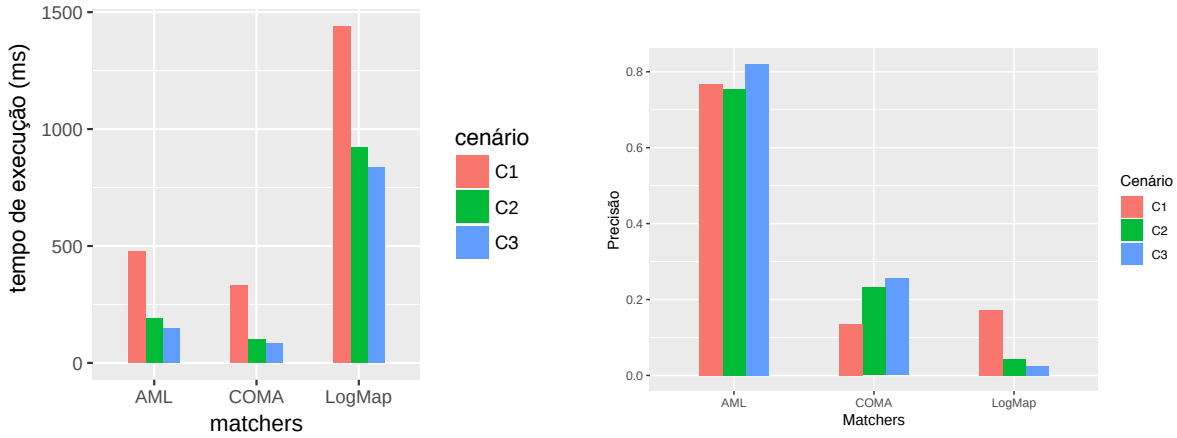
[7] Alinhamentos de referência referentes à cada cenário disponíveis em: <https://github.com/dass-cin/redoma-resources/tree/master/gold-standard>

Segmento	Classes da ontologia fonte	Propriedades de Objeto da Ontologia Fonte	Propriedades de Dado da Ontologia Fonte	Classes de Ontologia Alvo	Propriedades de Objeto da Ontologia Alvo	Propriedades de Dado da Ontologia Alvo	Comparação com Segmentos de Referência
source-segment-cmt-conference	15	25	4	12	31	10	92%
target-segment-ekaw-sigkdd	23	10	0	8	7	3	100%
source-segment-edas-iasted	47	10	1	9	23	0	88%
target-segment-ekaw-iasted	23	10	0	12	23	0	95%
target-segment-confOf-sigkdd	7	3	4	2	1	0	90%
source-segment-ekaw-iasted	23	10	0	12	23	0	87%
source-segment-edas-sigkdd	47	10	1	12	8	7	98%
target-segment-edas-sigkdd	47	10	1	12	8	7	90%
source-segment-conference-confOf	16	12	4	13	4	8	85%
target-segment-iasted-sigkdd	12	0	0	7	3	0	80%
source-segment-confOf-edas	7	3	4	31	9	0	75%
target-segment-edas-iasted	47	10	1	9	23	0	79%
source-segment-cmt-sigkdd	15	25	4	12	8	7	85%
target-segment-cmt-confOf	15	25	4	14	3	10	88%
source-segment-edas-ekaw	47	10	1	17	9	0	95%
source-segment-confOf-ekaw	7	3	4	10	6	0	90%
target-segment-cmt-ekaw	15	25	4	17	7	0	87%
target-segment-conference-confOf	16	12	4	13	4	8	85%
source-segment-cmt-iasted	15	25	4	9	23	0	89%
source-segment-conference-iasted	16	12	4	4	23	0	97%
target-segment-conference-ekaw	16	12	4	18	9	0	79%
target-segment-cmt-edas	15	25	4	21	20	12	82%
target-segment-cmt-conference	15	25	4	12	31	10	83%
source-segment-confOf-sigkdd	7	3	4	2	1	0	85%
source-segment-cmt-ekaw	15	25	4	17	7	0	89%
source-segment-conference-sigkdd	16	12	4	7	4	3	75%
target-segment-confOf-edas	7	3	4	31	9	0	93%
target-segment-edas-ekaw	47	10	1	17	9	0	85%
target-segment-conference-edas	16	12	4	37	17	4	83%
target-segment-conference-sigkdd	16	12	4	7	4	3	79%
source-segment-conference-ekaw	16	12	4	18	9	0	83%
target-segment-cmt-iasted	15	25	4	9	23	0	87%
target-segment-conference-iasted	16	12	4	4	23	0	92%
target-segment-cmt-sigkdd	15	25	4	12	8	7	89%
source-segment-conference-edas	16	12	4	37	17	4	81%
source-segment-ekaw-sigkdd	23	10	0	8	7	3	74%
source-segment-iasted-sigkdd	12	0	0	7	3	0	78%
source-segment-cmt-confOf	15	25	4	14	3	10	95%
source-segment-cmt-edas	15	25	4	21	20	12	83%
target-segment-confOf-ekaw	7	3	4	10	6	0	89%

Tabela 7 – Características dos segmentos de ontologias gerados e comparação com segmentos de referência

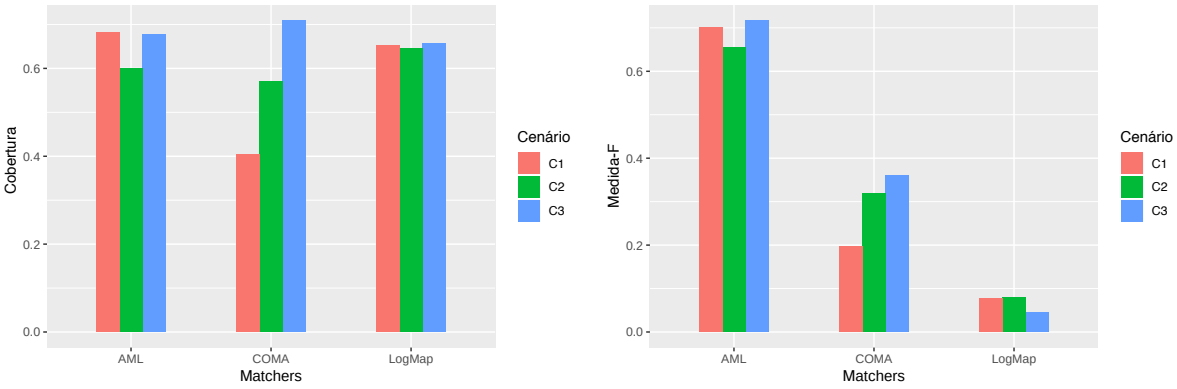
Esta diferença é refletida quando comparam-se os cenários usando a Medida-F (Figura 12d), que é uma medida de harmonização entre precisão e cobertura.

Apesar do objetivo deste experimento ter sido especificamente avaliar a eficácia da



(a) Tempo de execução obtido nos experimentos.

(b) Medida de precisão obtida nos experimentos.



(c) Medida de cobertura (*recall*) obtida nos experimentos.

(d) Medida-F (*F-Measure*) obtida nos experimentos.

Figura 12 – Resultados obtidos na execução dos experimentos do tipo 1 para os cenários C1, C2, C3

geração automática dos segmentos, a variação de desempenho relatada nos alinhamentos resultantes demonstra que, dependendo da métrica e do cenário, um *matcher* pode obter resultados de melhor ou pior qualidade. Este fato motiva a busca por abordagens que busquem levantar os aspectos que influenciam na obtenção de resultados de qualidade, de maneira a oferecer a recomendação dos melhores *matchers* para atender uma determinada tarefa de alinhamento. Esse é justamente o escopo do *framework* proposto no Capítulo 5.

4.5 CONSIDERAÇÕES FINAIS

A principal contribuição deste capítulo é um algoritmo para a geração de segmentos de ontologias, denominado SOMA. Três características deste algoritmo são: a) considera um conjunto de requisitos de dado (R_d), possibilitando que sejam obtidos segmentos que contenham conceitos relevantes para a aplicação; b) considera um conjunto de requisitos operacionais (R_o), permitindo que o usuário possa escolher entre obter segmentos com maior ou menor densidade, como também flexibilizar o valor mínimo para a medida

de similaridade entre conceitos; e c) leva em consideração um conjunto de princípios de conservação no momento da extração de conceitos, no intuito de minimizar a perda semântica.

Destaca-se que, uma vez obtidos os segmentos, eles podem ser utilizados como entrada para um sistema de geração de alinhamentos (*matcher*). Neste caso, há a possibilidade de utilização dos segmentos para a geração de alinhamentos parciais entre um par de ontologias. Com isso, pode-se focar apenas em um conjunto de conceitos relacionados aos requisitos. Outro uso seria trabalhar com a geração de alinhamentos completos, mas de maneira incremental, dividindo o processo em várias etapas e utilizando o resultado de cada execução como forma de aperfeiçoar as execuções posteriores.

No que concerne os experimentos, eles foram realizados de forma a comprovar as hipóteses levantadas neste tese em que é abordado o problemas de segmentação de ontologias. Os resultados foram discutidos e ilustrados por intermédio da apresentação de gráficos. Pelos resultados mostrados, observa-se que a estratégia de segmentação de ontologias apresentada no algoritmo SOMA produziu segmentos em conformidade com a conservação semântica dos elementos extraídos, reduzindo o tempo de execução do alinhamento de ontologias visto que trata-se apenas da comparação do conjunto de conceitos mais relevantes.

Outro detalhe que pode ser discutido é o tempo necessário para a geração dos segmentos, que somado ao tempo de execução, poderia onerar o desempenho de maneira geral. No entanto, como o SOMA utiliza uma estrutura de armazenamento baseada em grafos, mais especificamente o Neo4J, ele consegue fornecer segmentos no tempo de execução de uma consulta Cypher gerada pelo próprio algoritmo, o que nos experimentos conduzidos não chegou a levar mais de 2 segundos em todos os casos. Ressalta-se ainda que o Redoma suporta segmentos gerados por outros algoritmos ou até mesmo a utilização de ontologias completas como entrada, o que indica que o processo de geração de segmentos pelo SOMA é recomendável, mas não obrigatório.

Por fim, acredita-se que a partir da construção de uma base de conhecimento que contenha os resultados obtidos por um conjunto de *matchers* para alinhar segmentos derivados de ontologias diversas, seja possível ter uma referência para a definição de qual seria o *matcher* mais adequado para um cenário específico. O Capitulo 5, a seguir, aborda em detalhes o *framework* proposto para recomendação *matchers*. Nele, utilizam-se alinhamentos gerados entre segmentos fornecidos pelo SOMA como base para avaliação de desempenho dos *matchers*.

5 *FRAMEWORK* PARA RECOMENDAÇÃO DE *MATCHERS* DE ONTOLOGIAS

Conforme discutido nos capítulos anteriores, o problema de alinhamento de ontologias vem sendo amplamente estudado nos últimos anos. Como resultado, foram publicados vários trabalhos propondo sistemas para solucionar essa questão, denominados *matchers* (OCHIENG; KYANDA, 2018c; SATHIYA; GEETHA; SUGUMARAN, 2017). Os resultados das campanhas mais recentes da Ontology Alignment Evaluation Initiative (OAEI) têm demonstrado que os *matchers* alcançam desempenho variado, dependendo das ontologias consideradas como entrada (ACHICHI et al., 2017). Isto indica que não há um conjunto único de *matchers* que possa efetiva e eficientemente descobrir alinhamentos para todos os cenários possíveis. Além disso, com o aumento do tamanho e da heterogeneidade das ontologias, há a necessidade por sistemas automáticos, robustos e escaláveis para a geração de alinhamentos. Isto torna ainda mais desafiador para um usuário especialista, a seleção do *matcher* mais adequado.

Neste contexto, a principal contribuição desta tese é a concepção de um *framework* para prover recomendações de *matchers*, tomando como base as características das ontologias que serão alinhadas. Como diferencial, considera-se a extração de segmentos a partir de requisitos estabelecidos pelo usuário, utilizando o algoritmo proposto SOMA (vide Capítulo 4), que visa reduzir o escopo de busca durante a geração de alinhamentos, no intuito de possibilitar uma maior eficiência na recomendação de *matchers*.

O objetivo do *framework* proposto é melhorar a geração de alinhamentos de ontologias a partir do uso do melhor *matcher* (ou combinação de vários), considerando como entrada segmentos de ontologias. Neste sentido, de maneira mais geral, este capítulo apresenta as principais definições relacionadas ao uso do *framework*, seus componentes e como ele pode ser instanciado. De forma mais detalhada, o capítulo aborda os seguintes conteúdos: a Seção 5.1 apresenta a discussão em torno da motivação e definição do problema de recomendação de *matchers* no contexto deste trabalho. A Seção 5.2 introduz uma visão geral do *framework* proposto, servindo de direcionamento para as seções seguintes. A Seção 5.3 detalha a construção do perfil de ontologias, como forma de estabelecer um conjunto de métricas para classificar ontologias gerenciadas pelo *framework*. A Seção 5.4 descreve o processo de construção do perfil de *matchers*, destacando a sua dinamicidade, que reflete na concepção de um perfil específico para um *matcher* em tempo de execução, dependendo das ontologias fornecidas como entrada. A Seção 5.5 apresenta o processo de avaliação de qualidade dos *matchers* e o algoritmo para recomendar uma lista ordenada dos *matchers* que melhor se adequam ao cenário em questão. A Seção 5.6 mostra como o usuário poderá definir a estratégia utilizada para, a partir de um conjunto de *matchers* recomendados de maneira automática, selecionar o conjunto final que será considerado para a execução. A Seção 5.7 descreve como executar os *matchers* selecionados, de maneira

que os alinhamentos gerados e confirmados pelo usuário possam retroalimentar a base de conhecimento gerenciada pelo *framework*. A Seção 5.8 descreve os experimentos realizados para avaliar o *framework* Redoma, incluindo as os conjuntos de dados utilizados, objetivos, metodologia e a discussão dos resultados obtidos. Por fim, a Seção 5.9 resume o conteúdo e as contribuições apresentadas no capítulo.

5.1 MOTIVAÇÕES E DEFINIÇÃO DO PROBLEMA

A literatura mostra que nos últimos anos, centenas de *matchers* foram propostos, com o objetivo de oferecer diferentes perspectivas à resolução do problema de geração de alinhamentos de ontologias (OCHIENG; KYANDA, 2018c). Além das várias opções de classificação para os *matchers*, conforme sintetizado por Euzenat e Shvaiko (2013) e apresentadas no Capítulo 2, há a possibilidade de categorização baseada no tipo de informação utilizada para a geração de alinhamentos. Uma classificação neste sentido foi introduzida por Cruz et al. (2012), que consiste na divisão em quatro tipos: (i) sintáticos - que utilizam técnicas de processamento de *strings* para encontrar similaridades usando informações descritivas dos elementos, tais como nome, rótulo e comentários; (ii) estruturais - que usam técnicas de comparação de estrutura hierárquica para avaliar a similaridade baseada na hierarquia de relacionamentos entre os elementos; (iii) léxicos - que exploram características idiomáticas dos elementos, tais como sinônimos, palavras derivadas, entre outros - geralmente com o auxílio de um banco de dados externo (e.g., Wordnet) e (iv) de instância, que usam amostras de instâncias do dado pra detectar padrões e identificar similaridades entre elementos.

No geral, os sistemas de alinhamento de ontologias (e.g., COMA (MASSMANN et al., 2011), YAM (DUCHATEAU; BELLAHSENE, 2015)) usam mais de uma função de similaridade para explorar diferentes perspectivas ao tentar descobrir correspondências entre os elementos. No entanto, algumas características citadas anteriormente acabam tendo uma certa predominância. De fato, relatórios recentes da OAEI (ACHICHI et al., 2017) indicam que pelo menos 50% dos alinhamentos podem ser encontrados prontamente por meio de técnicas simples de comparação textual, o que explica uma maior predominância de *matchers* sintáticos e léxicos em relação aos demais. Isto também comprova que, por padrão, a maior parte dos pares de ontologias considerados na OAEI são ricas em características linguísticas. No entanto, este trabalho não considera isto como verdade absoluta para outros domínios, a ponto de descartar a avaliação da adequabilidade dessas duas categorias (como feito por Sathiya, Geetha e Sugumaran (2017)). Tendo em vista que, diferentemente da OAEI (que utiliza em sua maioria datasets sintéticos), em aplicações de mundo real, dependendo do domínio de conhecimento em que é aplicado, podem-se encontrar cenários em que se deseja alinhar ontologias com características bem peculiares (e.g., aplicações de domínio específico da Linked Open Data (HEATH; BIZER, 2011)). Por tal motivo, utiliza-se a segmentação para favorecer à combinação de técnicas diferentes que possam melhor

explorar as características de cada segmento extraído das ontologias.

Neste aspecto, considerando a recomendação de *matchers*, destacam-se alguns desafios existentes:

- **Ausência de uma relação clara entre as características dos *matchers* e a obtenção de resultados de melhor qualidade.** Um dos critérios que pode ser utilizado para avaliar a qualidade de um *matcher* é classificar a sua adequabilidade quanto às características predominantes nas ontologias fornecidas. Porém, mesmo para um usuário especialista, é bastante complexa a tarefa de identificar manualmente, na estrutura de elementos das ontologias, quais são as características que podem afetar a obtenção de resultados de melhor qualidade.

- **Definição de um conjunto de critérios e características de um *matcher* relevantes para indicar a sua adequação a um conjunto de ontologias específico.** Dependendo das características das ontologias fornecidas, um determinado tipo de *matcher* pode obter resultados de melhor ou pior qualidade. Por exemplo, os léxicos podem oferecer resultados próximos do ideal, quando lidam com ontologias contendo elementos descritos com termos conhecidos, que possam ser buscados em um dicionário de referência. Por outro lado, quando são usados para comparar ontologias contendo termos com nomenclatura desconhecida, o resultado obtido pode ser bastante inferior, em detrimento a outras estratégias, como a comparação no nível estrutural ou de amostras de instâncias. Assim, de certa forma, a boa avaliação de um *matcher* em cenários anteriores não garante que ele seja o mais adequado para o cenário corrente.

- **A definição da qualidade de um *matcher* depende de critérios fornecidos pelo usuário.** Os critérios utilizados para avaliar a qualidade de um *matcher* podem variar dependendo da aplicação. A exemplo disto, o usuário pode escolher priorizar a obtenção apenas das correspondências mais precisas envolvendo um determinado conceito, mesmo que com isso esteja abrindo mão de um resultado com maior cobertura. Por outro lado, pode-se desejar receber a maior quantidade possível de correspondências, tolerando a geração de uma certa quantidade de falsos-positivos. Além disso, algumas aplicações podem apresentar limitações no nível de infraestrutura (e.g., espaço em disco, memória disponível, banda de rede), enquanto outras podem ter restrições de tempo, não tolerando uma espera demasiada para a geração dos alinhamentos.

No intuito de abordar os desafios supracitados, propõe-se o Redoma (*Requirement-Driven Ontology Matching*), um *framework* para possibilitar a recomendação, seleção e execução de *matchers*. A escolha dos *matchers* é dada a partir da identificação de características dos segmentos de ontologias gerados pelo SOMA, da filtragem de alinhamentos

correlacionados gerados previamente e da observância de requisitos mínimos de qualidade definidos pelo usuário. Espera-se então, com o seu uso, resolver as seguintes questões:

i. Quais métricas calculadas a partir de ontologias podem ser utilizadas para possibilitar a correlação entre um par fornecido como entrada e as ontologias utilizadas em alinhamentos gerados anteriormente?

ii. Como medir, de maneira objetiva, a qualidade de um *matcher*, contemplando os interesses particulares do usuário na definição dos critérios?

iii. Como obter, a partir da definição de requisitos, um conjunto de *matchers* que possa fornecer melhores resultados para a geração de alinhamentos?

iv. Como manter o sistema de recomendação em constante aperfeiçoamento, de maneira a incorporar novas ontologias e, consequentemente, gerar novos alinhamentos?

Para lidar com a questão (i), utiliza-se o estado da arte das métricas para avaliação de ontologias, com o acréscimo de novas métricas (medidas de afinidade descritiva e estrutural) para estabelecer o perfil de uma ontologia. Este perfil é utilizado para detectar correlações entre as ontologias fornecidas como entrada e as que estiveram envolvidas em alinhamentos gerados anteriormente. Em relação à questão (ii), propõe-se o perfil de qualidade para um *matcher*, que leva em consideração, além do seu desempenho em cenários correlatos às ontologias fornecidas como entrada, também um conjunto de pesos para os critérios de qualidade definidos pelo usuário. Para a resolução da questão (iii), são introduzidas estratégias para possibilitar a filtragem mais refinada dos *matchers*, a partir, da definição de requisitos de qualidade. Por fim, em relação à questão (iv), o *framework* oferece uma infra-estrutura baseada em serviços para a execução dos *matchers*, de maneira padronizada (usando a Alignment API (DAVID et al., 2011)) e integrada a um conjunto de bases de dados gerenciadas pelo *framework*, possibilitando a persistência das informações obtidas durante e após a geração dos alinhamentos.

5.2 VISÃO GERAL DO *FRAMEWORK*

Visando lidar com os desafios levantados na seção anterior, propõe-se neste trabalho um *framework* para auxiliar o usuário na tomada de decisão em relação a qual *matcher* utilizar para resolver um alinhamento entre um par de ontologias. Denominado Redoma (*Requirement-driven Ontology Matching*), este *framework* tem como um de seus diferenciais a integração com um catálogo contendo exemplos de alinhamentos entre segmentos de ontologias gerados a partir de requisitos da aplicação. Através do seu uso, espera-se obter uma avaliação de qualidade dos *matchers* em tempo de execução, de acordo com as especificidades do cenário estabelecido para a geração de alinhamentos (e.g., características dos segmentos de ontologias a serem comparados).

A Figura 13 apresenta uma visão geral do *framework* Redoma, que é composto por cinco componentes principais: i) Geração do Perfil das Ontologias; ii) Geração do Perfil dos *Matchers*, iii) Avaliação e Recomendação de *Matchers*, iv) Seleção de *Matchers* e v) Execução de *Matchers* e Avaliação de Alinhamentos, conforme descreve-se a seguir.

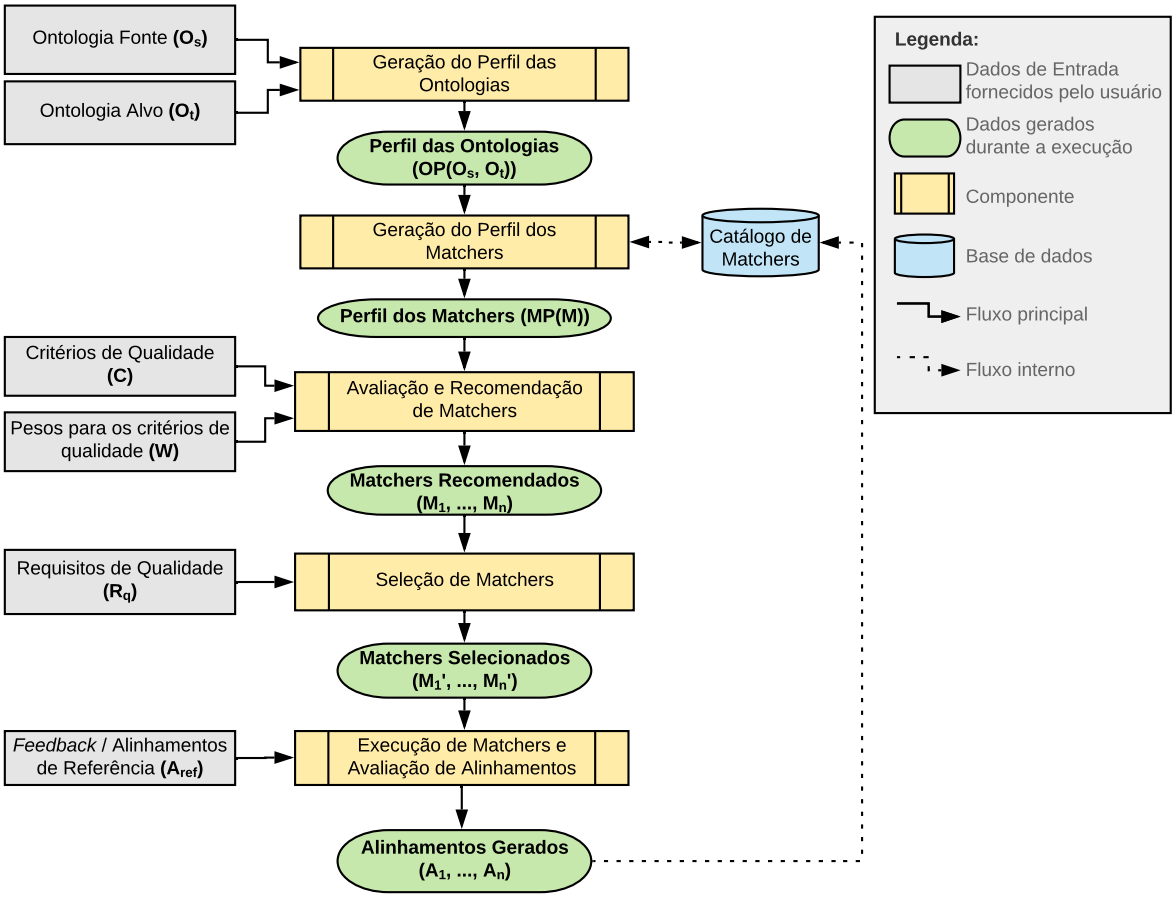


Figura 13 – Visão geral dos componentes do *framework* Redoma.

Geração do Perfil das Ontologias. Calcula uma série de métricas para a classificação de ontologias, denominada Perfil de Ontologia (OP), cujo intuito é quantificar as características das ontologias fornecidas como entrada (O_s e O_t), visando estabelecer correlações com as ontologias utilizadas em alinhamentos gerados previamente.

Geração do Perfil dos Matchers. Calcula um conjunto de métricas para caraterizar o desempenho de um *matcher*, denominado Perfil de *Matcher* (PM). Para isto, utiliza-se como base o seu desempenho em execuções anteriores que envolvam ontologias com as mesmas características das recebidas como entrada.

Avaliação e Recomendação de Matchers. Determina a medida de qualidade de um *matcher* ($Q(m)$). Para tal, utiliza-se o perfil do *matcher*, constituído na etapa anterior, juntamente de um conjunto de critérios de qualidade C, acompanhados dos respectivos pesos W.

Seleção de Matchers. Estabelece o conjunto final de *matchers* $(M'_1, ..., M'_n)$ a serem considerados a partir do cumprimento dos requisitos de qualidade R_q fornecidos.

Execução de *Matchers* e Avaliação de Alinhamentos. Dispara o processo de geração de alinhamentos utilizando os *matchers* recomendados. Para isto, utiliza-se a infraestrutura provida pelo *framework* para a execução de maneira padronizada. O resultado fornecido é um conjunto de alinhamentos $\{A_1, ..., A_n\}$, que poderá passar por uma avaliação final do usuário, no intuito de classificar as correspondências como corretas ou incorretas.

Para dar suporte aos componentes citados, é considerada uma base de dados auxiliar, denominada Catálogo de *Matchers*. O Catálogo de *Matchers* armazena informações sobre sistemas para a geração de alinhamentos. Além de metadados básicos como nome e versão, o catálogo disponibiliza informações como a URL de acesso, tornando possível executá-los como serviço. Este recurso também possibilita que o catálogo guarde o histórico de alinhamentos gerados, que, por sua vez, são utilizados como base para a avaliação dos *matchers*). Com base nisso, é concebido o perfil dos *matchers*, que suporta como critérios para avaliação de qualidade um conjunto de métricas objetivas (vide Seção 2.4.1), tais como precisão, cobertura e Medida-F, que são amplamente usadas na literatura.

Seguinte o conceito de reconciliação de esquemas definido por Belhajjame et al. (2011), os alinhamentos fornecidos como resultado ($\{A_1, ..., A_n\}$) são retornados ao usuário, que poderá avaliar cada correspondência contida nele, indicando se são válidas ou não. O conjunto de respostas do usuário resulta em um alinhamento de referência (A_{ref}) entre as ontologias avaliadas. Esse resultado é armazenado de volta no Catálogo de *Matchers* e poderá influenciar no cálculo do perfil do *matcher* em execuções futuras. Nas seções seguintes, apresentam-se detalhes sobre cada etapa relacionada aos componentes supracitados.

5.3 CONSTRUÇÃO DO PERFIL DAS ONTOLOGIAS

Nesta seção, detalha-se a construção dos perfis para as ontologias que são importadas no catálogo. Uma das contribuições apresentadas neste trabalho é a utilização de segmentos de ontologias para redução do escopo de busca na geração de alinhamentos (vide Capítulo 4). Especificamente para a recomendação de *matchers*, outro benefício em utilizar segmentos é a possibilidade de extrair as características de uma versão reduzida da ontologia, que contenha apenas os elementos mais relevantes para o usuário. Desta forma, torna-se possível avaliar os *matchers* com relação à sua eficiência ao gerar alinhamentos entre segmentos com as mesmas características dos segmentos fornecidos como entrada, o que visa possibilitar uma maior precisão na recomendação. A seguir, define-se a construção do perfil para as ontologias (O_s e O_t) recebidas como entrada.

Definição 5.1. Define-se por Perfil de Ontologia (*Ontology Profile*) um conjunto $OP = \{me_1, ..., me_n\}$, onde $\{me_1, ..., me_n\}$ são valores obtidos a partir de métricas que visam avaliar o nível de predominância de determinada característica em uma ontologia. Conforme descrito na Seção 2.5, essas métricas são categorizadas em quatro tipos: sintáticas, estruturais, léxicas e de instância. Além dos valores obtidos para cada tipo, o perfil da ontologia contém os valores obtidos para duas novas métricas propostas neste trabalho, definidas como medidas de afinidade descritivas e estrutural (vide Definição 5.3 e Definição 5.4), que realizam a agregação das métricas compatíveis com essas categorias.

O primeiro passo para a geração do perfil de uma ontologia (ou segmento) é a identificação das suas características básicas, tais como quantidade de elementos, classes, propriedades, entre outros (vide Seção 2.5). O Quadro 4 exemplifica a identificação dos atributos das ontologias Cmt e OpenConf, ambas advindas do conjunto de dados da trilha de conferências da OAEI.

Quadro 4 – Exemplo de identificação de atributos das ontologias Cmt e OpenConf

Ontologia Cmt		**Ontologia OpenConf**	
Característica	**Quantidade**	**Característica**	**Quantidade**
Atributos	10	Atributos	21
Classes	29	Classes	62
Propriedades de Dado	10	Propriedades de Dado	21
Propriedades de Objeto	49	Propriedades de Objeto	24
Rótulos nulos	88	Rótulos nulos	107
Comentários nulos	84	Comentários nulos	100
Total de Elementos	88	Total de Elementos	107
Total de relações de subclasse	43	Total de relações de subclasse	101
Outras relações	49	Outras relações	24

Com base nesses atributos, calculam-se as métricas que compõem o perfil das ontologias (vide Seção 2.5). Inicialmente, o perfil é calculado individualmente, i.e., para cada ontologia fornecida. Porém, para que seja possível relacionar um par de ontologias à alinhamentos gerados anteriormente, torna-se necessário unificar os valores obtidos para cada ontologia envolvida, de maneira a estabelecer um perfil de ontologia unificado que represente o alinhamento.

Para isso, define-se a função PC que realiza a combinação dos valores para as métricas contidas no perfil de cada uma das duas ontologias ($OP(O_1)$ e $OP(O_2)$):

Definição 5.2. A função *PC Profile Combination* recebe um par de perfis de ontologias ($OP(O_1)$ e $OP(O_2)$) e utiliza a função *FS-A Feature Similarity Average* para combinar cada métrica $me_i \in OP(O_1)$ e $me_j \in OP(O_2)$ em um conjunto agregado de métricas $\{me'_1, ..., me'_n\}$, unificando os valores obtidos para o par de ontologias fornecidas.

$$PC : \forall(me_i \in OP(O_1), me_j \in OP(O_2)) : me'_i = FS\text{-}A(me_i, me_j) \rightarrow \{me'_1, ..., me'_n\}$$

A função $FS\text{-}A$ utilizada para combinar os valores me_i e me_j, obtidos para cada métrica que compõe o perfil, foi proposta por Cruz et al. (2012) (descrita em detalhes na Seção 2.5.5) e consiste na agregação de outras duas funções: FS e A, conforme apresenta-se a seguir.

$$FS_{(M)} = \frac{M_L}{M_H[log(M_H - M_L + 1) + 1]},$$

$$FS\text{-}A_{(M)} = A + FS$$

Onde:

- M é o conjunto total de valores fornecidos, no caso $\{me_i, me_j\}$;
- M_L é o menor valor entre me_i e me_j;
- M_H é o maior valor entre me_i e me_j;
- A é a média dos valores me_i e me_j.

O Quadro 5 mostra um exemplo deste cálculo, considerando o par de ontologias Cmt e OpenConf. Ressalta-se que caso fossem considerados segmentos, que é inclusive uma das premissas do *framework*, seria seguido o mesmo procedimento, apenas seriam obtidos valores diferentes para as métricas, visto que um menor número de elementos seria considerado.

Quadro 5 – Exemplo do cálculo e da combinação dos perfis das ontologias Cmt e OpenConf

Perfil da Ontologia Cmt		**Perfil da Ontologia OpenConf**		**Perfil Combinado**
Métrica	**Valor**	**Métrica**	**Valor**	**FS-A**
Profundidade Média	2,34	Profundidade Média	2,62	3,19
Riqueza de Relacionamentos	0,53	Riqueza de Relacionamentos	0,19	0,63
Riqueza de Atributos	0,34	Riqueza de Atributos	0,33	1,37
Cobertura do Wordnet	1	Cobertura do Wordnet	0,79	1,55
Porcentagem de Comentários e Rótulos Nulos	0,97	Porcentagem de Comentários e Rótulos Nulos	0,96	1,89
Unicidade de Rótulos	0	Unicidade de Rótulos	0	0
Riqueza de Herança	0,48	Riqueza de Herança	0,94	1,08
Riqueza de Atributos de Classe	0	Riqueza de Atributos de Classe	1	0,5
Riqueza de Estrutura Externa de Conceitos	0	Riqueza de Estrutura Externa de Conceitos	0	0
Riqueza de Classe	0	Riqueza de Classe	0,04	0,02

A próxima etapa consiste em agrupar as métricas do perfil de ontologia utilizando como base a classificação existente na literatura, que define três tipos de métricas: sintáticas, estruturais e léxicas (vide Seção 2.5). Como o escopo deste trabalho se concentra

apenas na camada de modelo, foram desconsideradas as métricas que abordam o uso de instâncias. Ressalta-se que, como algumas categorias são compostas por mais de uma métrica, especialmente, as sintáticas e estruturais, seria pouco provável encontrar perfis pré-calculados com exatamente os mesmos valores de um novo par de ontologias fornecido, o que exigiria recorrer ao cálculo dos valores mais próximos para determinar a correlação entre perfis, tal qual é feito por Cruz et al. (2012). Porém, neste caso, a aproximação é feita por métrica individual e não por categoria. Assim, buscando uma melhor eficiência na correlação de perfis de ontologias, propõe-se duas novas métricas para compor o perfil de ontologias baseado nas duas categorias de métricas (descritivas e estruturais):

Definição 5.3. A *Afinidade Descritiva* (AF_{desc}) consiste na combinação das métricas que envolvem a utilização de elementos descritivos (terminológicos e léxicos) para a identificação de correspondências. Para tal, utiliza-se a combinação das métricas terminológicas (RCN e UR) somada ao valor obtido para a métrica léxica (CW), conforme descreve-se a seguir.

$$AF_{desc} = \frac{(1 - me_{RCN}) + me_{UR}}{2} + me_{CW}$$

Onde:

- me_{RCN} corresponde ao valor obtido para a métrica terminológica de Comentários e Rótulos Nulos. Como o que poderia beneficiar os *matchers* que usam técnicas de comparação descritivas seria o inverso (rótulos não-nulos), obtém-se o complemento desta métrica (i.e., o resultado da sua subtração por 1).

- me_{UR} corresponde ao valor obtido para a métrica terminológica de Unicidade de Rótulos, que indica a porcentagem de elementos (valor entre [0,1]) que possuem rótulo diferente do nome, no sentido de aproveitar alguma informação adicional presente nos rótulos.

- me_{CW} corresponde ao valor obtido para a métrica léxica de Cobertura do Wordnet, que indica a porcentagem (valor entre $[0, 1]$) de elementos descritos (pelo nome ou pelo rótulo) com termos válidos existentes na base de dados do Wordnet.

Definição 5.4. A *Afinidade Estrutural* (AF_{est}) consiste na combinação das métricas que consideram a análise da estrutura das ontologias. Para tal, calcula-se a média entre as métricas me_{RR}, me_{RH}, me_{PM}, sendo a última normalizada para conter valores no intervalo $[0, 1]$ a partir da divisão pelo valor α. Por fim, o resultado é somado com a média entre as métricas me_{RAC} e me_{RECE}, sendo estas calculadas de maneira que forneça a média ignorando os valores 0 (função NAN_MEAN), conforme detalha-se a seguir:

$$AF_{est} = \frac{me_{RR} + me_{RH} + (me_{PM}/\alpha)}{3} + NAN\_MEAN(me_{RAC}, me_{RECE})$$

Onde:

- me_{RR} corresponde ao valor obtido para a métrica de Riqueza de Relacionamentos, que calcula a porcentagem de relacionamentos que são diferentes de herança (subClassOf).
- me_{RH} corresponde ao valor obtido para a métrica de Riqueza de Herança, que calcula a porcentagem de relacionamentos do tipo subClassOf são referenciados pelo total de classes da ontologia.
- me_{PM} corresponde ao valor obtido para a métrica de Profundidade Média, que consiste na média do número máximo de superclasses referenciadas por cada classe presente na ontologia.
- α corresponde a uma constante que representa a profundidade máxima registrada por classe para todas as ontologias armazenadas, utilizada com o objetivo de normalizar o valor da Profundidade Média no formato de porcentagem.
- NAN_MEAN é uma função que calcula a média simples entre os valores fornecidos, ignorando os de valor nulo.
- me_{RAC} corresponde ao valor obtido para a métrica de Riqueza de Atributos de Classe, que é composto pela porcentagem do número de classes que contém propriedades de objeto.
- me_{RECE} corresponde ao valor obtido para a métrica de Riqueza de Estrutura Externa de Conceitos, que consiste na porcentagem do número de classes que possuem propriedades de dado.

A Tabela 8 mostra um exemplo de como as métricas do perfil de ontologias são agrupadas por categoria e como ocorre o cálculo das medidas de afinidade descritiva e estrutural propostas, considerando as ontologias Cmt e OpenConf. Para ilustrar o cálculo de AF_{est}, considera-se o valor 3 para a constante α.

Tabela 8 – Exemplo do cálculo das métricas dos perfis agrupadas por categoria

	Métricas do Perfil de Ontologias										
Ontologia	**Sintáticas**		**Léxicas**	**Estruturais**						**Afinidades**	
	UR	**RCN**	**CW**	**RR**	**RA**	**RH**	**PM**	**RAC**	**RECE**	**AF_{desc}**	**AF_{est}**
Cmt	0	0,95	1	0,53	0,34	0,48	2,34	0	0	**1,05**	**0,59**
OpenConf	0	0,93	0,79	0,19	0,33	0,94	2,62	1	0	**0,86**	**1,66**

Nota – Métricas do perfil de ontologias (descritas no Capítulo 2) - **UR:** Unicidade de Rótulos, **RCN:** Rótulos e Comentários Nulos, **RR:** Riqueza de Relacionamentos, **RA:** Riqueza de Atributos, **RAC:** Riqueza de Atributos de Classes, **PM:** Profundidade Média, **RECE:** Riqueza Estrutural de Conceitos Externos.

A partir do cálculo das medidas de afinidade, os valores definidos no perfil das ontologias são agrupados em afinidade descritiva e estrutural, o que facilita a classificação e reduz a necessidade de um conjunto de referência vasto, que contemple todas as métricas definidas

no perfil. Outro detalhe é que os valores de afinidade obtidos para cada ontologia (fonte, alvo) também são combinados. Para tal, da mesma forma como ocorre com as demais métricas que compõem o perfil, utiliza-se a função *FS-A* recebendo os valores obtidos para as medidas de afinidade nas duas ontologias. A escolha por esta função é justificada pelo fato de ela ter sido adotada anteriormente para a combinação dessas métricas por Cruz et al. (2012), obtendo um resultado satisfatório.

No entanto, para que seja possível estabelecer a proximidade entre os perfis calculados para diferentes pares de ontologias, apenas o valor absoluto fornecido pelas medidas de afinidade ainda não seria suficiente. Propõe-se, então, a classificação das afinidades em uma escala de três níveis: baixa, média e alta. Para essa definição, utiliza-se o cálculo dos percentis, considerando todo o histórico armazenado no catálogo, separados em três grupos. Assim, caso a afinidade combinada esteja no primeiro grupo, i.e., percentil entre $[0.0, 0.33]$, ela será classificada como "BAIXA". Caso esteja no segundo grupo, i.e., percentil entre $(]0.33, 0.66])$, ela será classificada como "MÉDIA". Por fim, caso esteja no terceiro grupo, i.e., percentil entre $(]0.66, 1])$, ela será classificada como "ALTA".

A Tabela 9 exemplifica a combinação das afinidades (AF_{desc} e AF_{est}) obtidas para cada ontologia (descritas na Tabela 8) e mostra a classificação com base nos percentis. Desta forma, ao preencher o catálogo de *matchers* com um conjunto de alinhamentos previamente executados, torna-se possível filtrar os *matchers* que obtiveram melhor desempenho relacionado à categoria mais presente no par de ontologias comparado. Como resultado, para avaliar um *matcher*, serão considerados apenas os alinhamentos que tenham sido gerados para pares de ontologias que possuem as mesmas medidas de afinidade do que foi obtido para o par de ontologias fornecido de entrada.

Tabela 9 – Exemplo da classificação das afinidades do par de ontologias

Medida de Afinidade	**Cmt**	**OpenConf**	**FS-A**	**Percentil**	**Classificação**
Afinidade Descritiva	1,02	0,86	1,54	0,75	**ALTA**
Afinidade Estrutural	0,59	1,66	1,33	0,55	**MÉDIA**

A obtenção dos segmentos e da classificação pela combinação do perfil das ontologias fornecidas como entrada é uma etapa essencial para viabilizar a recomendação dos *matchers*. Como forma de enriquecer o catálogo de ontologias ao longo do tempo, os segmentos e os perfis gerados são persistidos, sendo possível o acesso posterior pelos demais componentes do *framework*.

Na próxima seção, descreve-se como o perfil de ontologias é utilizado para o desenvolvimento de um perfil dinâmico para os *matchers*, estabelecendo uma avaliação que leve em consideração o desempenho dos *matchers* para gerar alinhamentos entre ontologias com as mesmas características dos segmentos fornecidos como entrada.

5.4 CONSTRUÇÃO DO PERFIL DOS *MATCHERS*

A avaliação de qualidade dos *matchers* é uma etapa fundamental para viabilizar o processo de recomendação. A literatura contém trabalhos (e.g., *data profiling*, proposto por Naumann (2014)), cujo intuito é avaliar a qualidade de um determinado sistema, sendo costumeiramente tratada por Perfil de Qualidade ou *Profiling*. Neste sentido, uma primeira solução seria estabelecer um perfil de qualidade geral para os *matchers*, a partir da utilização de uma base de conhecimento, realizando o cálculo de um conjunto de métricas que possa determinar qual seria o *matcher* de melhor qualidade.

No entanto, como citado anteriormente, o desempenho dos *matchers* pode variar de acordo com o cenário disposto, o que indicaria que a construção de um perfil estático, que forneça uma lista dos melhores *matchers* independentemente do domínio da aplicação, não refletiria, majoritariamente, as melhores recomendações. A proposta deste trabalho é, portanto, construir um perfil para os *matchers* que seja dinâmico, i.e., que considere os aspectos inerentes a um determinado par de ontologias que é submetido para a geração de alinhamentos, e a partir deles, calcular a qualidade do *matcher*.

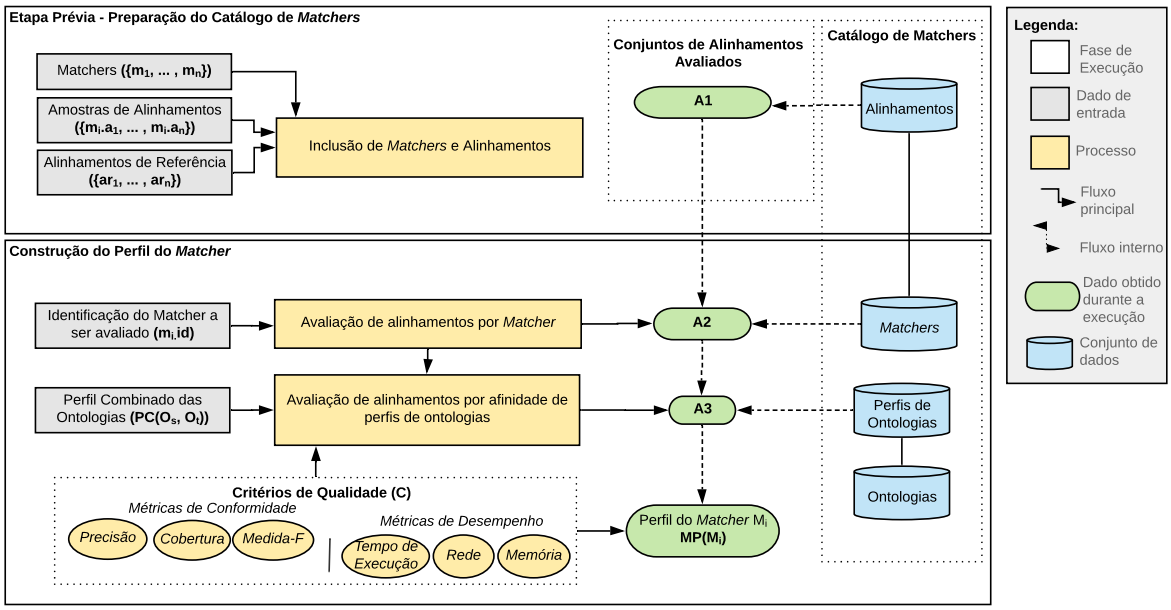


Figura 14 – Ilustração do processo de construção do perfil dinâmico de um *matcher*.

Seguindo esta premissa, considera-se, no processo de construção do perfil da qualidade de um *matcher*, apenas a amostra de resultados obtidos por ele, visando filtrar o desempenho de acordo com alinhamentos gerados para ontologias com o mesmo perfil. A Figura 14 ilustra como o perfil é concebido. Para viabilizar a construção do perfil, é requerida uma fase preliminar de preparação do catálogo de *matchers*, que obrigatoriamente deverá ser executada pelo menos uma vez, no intuito e popular o catálogo com alinhamentos e *matchers* que serão utilizados na construção do perfil, que por sua vez, é composto por duas etapas: i) Avaliação de alinhamentos por *matcher* e ii) Avaliação de alinhamentos

por afinidade de perfis de ontologias. As seções a seguir descrevem em detalhes cada uma dessas etapas.

5.4.1 Inclusão de *Matchers* e Alinhamentos

Para que seja possível gerar o perfil dos *matchers*, é requerida a execução de uma etapa preliminar, que envolve a inclusão de um conjunto inicial de *matchers* ($\{m_1, ..., m_n\}$), como também amostras de alinhamentos gerados por eles ($\{m_i.a_1, ..., m_n.a_n\}$), sendo $m_i \in \{m_1, ..., m_n\}$. Após isso, utilizam-se alinhamentos de referência ($\{ar_1, ..., ar_n\}$) para avaliar a qualidade dos alinhamentos fornecidos. Neste processo, a qualidade é mensurada a partir de um conjunto de critérios de qualidade C, que são definidos com base em métricas de conformidade (e.g., precisão, cobertura, Medida-F) e métricas de desempenho (e.g., tempo de execução, uso de rede e uso de memória), conforme descrito na Seção 2.4.1).

O Quadro 6 apresenta um exemplo da estrutura do catálogo contendo cinco *matchers* conhecidos: COMA, AML, LogMap, ALIN e YAM. O catálogo armazena informações como o nome, versão e um endereço representando um *endpoint* de Serviço Web. O *endpoint* é a forma pela qual o *matcher* receberá como entrada dois endereços de arquivos representando ontologias (e.g., OWL) e uma lista de parâmetros de configuração. O resultado fornecido é um alinhamento seguindo como padrão o modelo da Alignment API (DAVID et al., 2011). Desta forma, torna-se possível uniformizar os formatos de entrada e saída para a obtenção de resultados por vários *matchers*.

Quadro 6 – Exemplo de *matchers* cadastrados no catálogo

Nome	**Versão**	**Endpoint de Serviço Web**
COMA	3.0	http://localhost:8083/api/coma
AML	2.2	http://localhost:8084/api/aml
LogMap	2.0	http://localhost:8085/api/fcamap
ALIN	1.0	http://localhost:8082/api/alin
YAM	2.0	http://localhost:8081/api/yam

Após adicionar alguns *matchers* no catálogo, utiliza-se a sua infraestrutura de serviços para a geração de alinhamentos prévios, armazenando-os no catálogo e mantendo a referência para o *matcher* que os gerou. A Tabela 10 mostra um exemplo de como os alinhamentos gerados por diferentes *matchers* são armazenados. Neste caso, ilustram-se fragmentos de alinhamentos entre as ontologias Cmt e OpenConf, gerados pelos *matchers* COMA e LogMap.

Uma vez que o catálogo de *matchers* contém um conjunto inicial de alinhamentos, o próximo passo consiste em avaliar a qualidade das correspondências fornecidas utilizando alinhamentos de referência. A Tabela 11 ilustra a estrutura utilizada para armazenar a avaliação de um alinhamento gerado pelo *matcher* COMA entre as ontologias Cmt e

Tabela 10 – Exemplo de alinhamentos armazenados no catálogo de *matchers*

Ontologia Fonte (Cmt)	**Ontologia Alvo (OpenConf)**	**Relação**	**Similaridade**	**Matcher**
Person	Person	Equivalence	1	COMA
hasAuthor	has_authors	Equivalence	0,72	COMA
submitPaper	Publisher	Equivalence	0,45	LogMap
hasConflictOfInterest	has_an_abstract	Equivalence	0,33	LogMap

OpenConf. Nos experimentos realizados (Seção 5.8), utiliza-se essa estrutura para armazenar conjuntos de alinhamentos envolvendo as ontologias presentes nas campanhas da OAEI. Como são disponibilizados os alinhamentos de referência para cada par de ontologias, foi possível avaliar a qualidade do resultado inicial, incluindo diferentes configurações para os *matchers*, a partir da extração de vários segmentos por ontologias.

Tabela 11 – Exemplo de avaliação de alinhamentos do catálogo de *matchers*

Matcher	**Ontologia Fonte**	**Ontologia Alvo**	**Precisão**	**Cobertura**	**Medida-F**	**Corresp. Encontradas**	**Corresp. Esperadas**	**Corresp. Corretas**
COMA	Cmt	OpenConf	0,54	0,5	0,52	11	12	6

O conjunto completo de alinhamentos fornecidos e avaliados nesta etapa é definido como $A1$. Ao observar o desempenho dos *matchers* considerando todo o conjunto $A1$, pode-se estabelecer o desempenho global dos *matchers* registrados no catálogo. Isso pode ser utilizado, por exemplo, para definir um *matcher* padrão, a ser utilizado caso nenhum *matcher* seja recomendado pelo *framework*.

Ressalta-se, no entanto, que nas próximas etapas, o Redoma busca conceber um perfil dinâmico, por meio da extração de subconjuntos de alinhamentos para medir de maneira mais precisa o desempenho dos *matchers*, levando em consideração apenas os alinhamentos gerados entre ontologias com as mesmas características das fornecidas como entrada. Além disso, o intuito é que os dados contidos no catálogo possam ser atualizados ao longo do tempo, inclusive, levando em consideração, as possíveis intervenções de usuários para a avaliação dos alinhamentos obtidos.

5.4.2 Avaliação de Alinhamentos por Matcher

Uma vez que a etapa de preparação do catálogo foi executada ao menos uma vez, torna-se possível a construção do perfil dos *matchers*. Para tal, é avaliado o desempenho geral de cada *matcher* (m_i) individualmente, a partir do fornecimento da sua identificação. Em seguida, por intermédio de uma consulta ao catálogo, é extraído um subconjunto de alinhamentos que foram gerados por aquele determinado *matcher*.

Geralmente, é possível executar *matchers* com diferentes combinações de configuração. Porém, somente o desafio de encontrar a melhor configuração para um determinado cenário, já é um problema abordado há um tempo significativo na literatura, tratado por *tuning* ou calibração de *matchers* (e.g., (LEE et al., 2007), (YANG et al., 2014)). Para resolver este caso, o catálogo de *matchers* proposto trata cada configuração de *matcher* como um registro distinto, no sentido de possibilitar a avaliação de qualidade para diferentes combinações de parâmetros de configurações.

Por exemplo, nos experimentos gerados neste trabalho, foram consideradas 65 configurações diferentes para o COMA. Cada uma foi registrada no catálogo com um identificador próprio, possibilitando que o processo de avaliação possa levar em conta a melhor recomendação, incluindo as diferentes configurações de um mesmo sistema.

Outra possibilidade seria incluir combinações de diferentes *matchers* no catálogo, na qual seria possível testar o desempenho individual (ou de suas respectivas configurações), como também verificar como se daria o desempenho de combinações de sistemas distintos. Peukert, Massmann e Koenig (2010) destacam que várias funções podem ser utilizadas para a combinação de correspondências geradas por sistemas diferentes. Neste caso, cada combinação utilizando um determinado tipo de função poderia ser registrada como um identificador diferente no catálogo.

Desta forma, para contemplar a obtenção dos alinhamentos nesta etapa, define-se a seguir a função AM, responsável por gerar o subconjunto de alinhamentos $A2$ referentes a um *matcher* m_i.

Definição 5.5. A função AM (*Alignments by Matcher*) recebe o conjunto de alinhamentos $A1$, a identificação de um *matcher* ($m_i.id$), e fornece um subconjunto de alinhamentos ($A.m_i \subset A1$) denominado $A2$.

$$AM : A1 \times m.id \rightarrow A1 \times m_i \rightarrow A.m_i \subset A1 \rightarrow A2$$

O subconjunto de alinhamentos $A2$ é utilizado como entrada para a próxima etapa. A seguir, descreve-se como as métricas do perfil das ontologias, calculadas para as ontologias fornecidas como entrada (O_s e O_t), são utilizadas como critério para extrair um novo subconjunto a partir de $A2$.

Avaliação de Alinhamentos por Afinidade de Perfis de Ontologias

Quando uma ontologia é importada no catálogo, o seu perfil é construído e armazenado como parte dos seus metadados. Assim, a partir do conjunto de alinhamentos $A2$, que representa todo o histórico gerado por um *matcher* m_i, aplica-se a função AP para extrair um novo subconjunto ($A3$), levando em consideração os que tenham a classificação de afinidade compatível com as ontologias fornecidas como entrada.

Definição 5.6. A função *AOP* (*Alignments by Ontologies Profile*) recebe o subconjunto de alinhamentos $A2$, o perfil combinado das ontologias $PC(O_s, O_t)$ e fornece um subconjunto ($A2.P(O_s, O_t) \subset A2$), denominado $A3$.

$$AOP : A2 \times P(O_s, O_t) \rightarrow (A2.P(O_s, O_t)) \subset A2 \rightarrow A3$$

Em suma, o conjunto de alinhamentos $A3$ representa os que foram gerados por um determinado *matcher* e que envolvam ontologias com a mesma classificação de afinidade das ontologias fornecidas como entrada. Assim, o conjunto $A3$ é utilizado como referência para calcular os valores de cada critério de qualidade obtidos por um *matcher*.

A Tabela 12 apresenta um exemplo do perfil de qualidade para um conjunto de *matchers* $\{m_1, m_2, m_3, m_4\}$. No caso, são mostrados valores para métricas de conformidade (precisão, cobertura, Medida-F) e de desempenho (tempo de execução, uso de rede e uso de memória). Note que esse perfil é relativo a um cenário em que a afinidade estrutural seja classificada como "MÉDIA" e a afinidade descritiva como "ALTA", conforme exemplificado na Tabela 9. Como as medidas de afinidade podem variar de acordo com a classificação das ontologias fornecidas como entrada, os valores das métricas que compõem o perfil também são variáveis de acordo com o cenário em questão.

Tabela 12 – Exemplo do Cálculo do Perfil de Matchers para Afinidade Estrutural "MÉDIA" e Afinidade Descritiva "ALTA"

Matcher	**Precisão**	**Cobertura**	**Medida-F**	**Tempo**	**Rede**	**Memória**
m1	0,95	0,92	0,93	0,30	0,5	0,2
m2	0,40	0,70	0,51	0,25	0,3	0,6
m3	0,60	0,40	0,48	0,15	0	0,8
m4	0,80	0,20	0,32	0,90	0	0,5

A construção do perfil é uma etapa essencial para que seja possível avaliar a qualidade dos *matchers*. No entanto, para completar esse processo, há a necessidade de definição de quais critérios de qualidade serão utilizados e qual o nível de importância de cada um. No contexto do Redoma, isto é feito a partir da utilização de estratégias que permitam a definição de critérios e a atribuição de pesos, conforme abordado na seção seguinte.

5.5 AVALIAÇÃO E RECOMENDAÇÃO DE *MATCHERS*

A avaliação dos *matchers* é realizada a partir da obtenção dos valores para as métricas fornecidas pelo perfil de qualidade do *matcher*. Esses valores são agregados em uma medida global denominada medida de qualidade de um *matcher* ($Q(m)$), que toma como base o cálculo da média harmônica ponderada (EUZENAT; SHVAIKO, 2013). Neste caso, o usuário poderá estabelecer pesos para cada critério de qualidade escolhido, tornando possível priorizar os *matchers* que obtiveram melhor desempenho nas métricas de maior peso.

Definição 5.7. A **medida de qualidade de um *matcher*** ($Q(m)$) é calculada a partir da definição de um conjunto de critérios de qualidade C, um conjunto de pesos W para os critérios ($w_i \in W$, sendo $i \in C$) e um conjunto de valores das métricas para cada critério (V_i, sendo $i \in C$) obtidos por um *matcher* m, resultando em um valor único, conforme descrito:

$$Q(m) = \frac{\sum_{i \in C} w_i}{\sum_{i \in C} \frac{w_i}{V_i}} \tag{5.1}$$

A seguir é ilustrado como seria realizado o cálculo de $Q(m)$, tomando como exemplo dois cenários hipotéticos ($H1$ e $H2$) e um conjunto de *matchers* $\{m_1, m_2, m_3, m_4\}$. Neste caso, foram considerados como critérios de qualidade precisão, cobertura e tempo de execução. A Tabela 13 ilustra a definição de pesos para cada critério nos dois cenários, enquanto a Tabela 14 ilustra os valores obtidos para as respectivas métricas por cada *matcher*, incluindo os valores finais da métrica de qualidade $Q(m)$, levando em consideração os pesos definidos para cada cenário.

Tabela 13 – Exemplo da definição de pesos para cada critério de qualidade

Cenário	**Precisão**	**Cobertura**	**Tempo de Execução**
H1	0,3	0,5	1
H2	0,5	1	0,5

Tabela 14 – Exemplo da obtenção da medida de qualidade dos *Matchers* para os cenários H1 e H2

Matcher	**Precisão**	**Cobertura**	**Tempo**	**Q(m) em H1**	**Q(m) em H2**
m_1	0,95	0,92	0,30	0,75	0,51
m_2	0,40	0,70	0,25	0,48	0,32
m_3	0,60	0,40	0,15	0,37	0,26
m_4	0,80	0,20	0,90	0,30	0,65

A partir da obtenção dos valores de qualidade $Q(m)$ para cada *matcher* $\{m_i, ..., m_n\}$, torna-se possível classificá-los priorizando os de maior valor. O Algoritmo 3 detalha o fluxo do processo de recomendação de *matchers*, incluindo a geração dos perfis de ontologias, dos *matchers* e o cálculo de $Q(m)$.

Seguindo os valores de exemplo da Tabela 14, caso seja definido que devem ser selecionados os 3 melhores *matchers* classificados (com base nos valores de qualidade $Q(m)$), o cenário $H1$ consideraria os *matchers* $\{m_1, m_2, m_3\}$, enquanto que o $H2$ os *matchers* $\{m_4, m_1, m_2\}$, de acordo com os pesos definidos para os critérios de qualidade.

Para definir a quantidade de *matchers* selecionados dentre os recomendados no conjunto M', pode-se utilizar um limiar mínimo que será requerido para $Q(m)$, ou até mesmo

Algoritmo 3 Algoritmo para recomendação de *matchers* (MR)

1: **function** $MR(M,\ A,\ O_s, O_t,\ C,\ W)$
▷ M corresponde a uma lista de *matchers* candidatos fornecida pelo catálogo.
▷ $A1$ corresponde à lista de todos os alinhamentos armazenados no catálogo.
▷ O_s corresponde à ontologia fornecida como entrada;
▷ O_t corresponde à ontologia alvo fornecida como entrada;
▷ C corresponde a um conjunto de critérios de qualidade usados para a avaliação de um *matcher*
▷ W corresponde ao conjunto de pesos para cada métrica C
▷ AF corresponde à classificação de afinidade do perfil de uma ontologia
2: $PC_{(OP_1,OP_2)} \leftarrow PC(OP_{O_s}, OP_{O_t})$ ▷ Combina os perfis das ontologias
3: **for all** $m_i \in M$ **do** ▷ Percorre cada *matcher* $m_i \in M$
4: $A2 \leftarrow \emptyset$
5: **for all** $A_i \in A1$ **do** ▷ Percorre todos os alinhamentos
6: **if** $A_i.matcher = m_i$ **then** ▷ Adiciona a $A2$ apenas alinhamentos de m_i
7: $A2.push(A_i)$
8: **end if**
9: **end for**
10: $A3 \leftarrow \emptyset$
11: **for all** $A_j \in A_2$ **do** ▷ Percorre os alinhamentos do *matcher* m_i
12: **if** $AF(PC(A_j.source, A_j.target)) = AF(PC_{(OP_1,OP_2)})$ **then**
13: $A_2.push(A_j)$ ▷ Adiciona em A_2 apenas os alinhamentos cujo perfil das ontologias indique as mesmas afinidades com o perfil das ontologias fornecidas como entrada
14: **end if**
15: **end for**
16: $m_i.q \leftarrow Q(m_i) \leftarrow A3 \times C \times W$ ▷ Calcula a medida de qualidade Q para o *matcher* m_i com base na avaliação do conjunto $A3$, considerando os pesos W para os critérios C
17: **end for**
18: $M' \leftarrow M.sort(q)$ ▷ Ordena a lista de *matchers* pela medida de qualidade Q
19: **return** M' ▷ Retorna a lista de *matchers* atualizada com a medida de qualidade
20: **end function**

especificar uma quantidade fixa de *matchers* que serão considerados, conforme abordado na próxima seção.

5.6 SELEÇÃO DE *MATCHERS*

Durante a etapa de avaliação de qualidade, é atribuída uma medida $Q(m)$ para os *matchers* $\{m_1, ..., m_n\}$ existentes no catálogo, de acordo com a compatibilidade com as ontologias (ou segmentos) fornecidas como entrada. Desta maneira, pode-se estabelecer um *ranking* dos *matchers* avaliados. No entanto, o usuário poderá optar por definir uma estratégia para estabelecer a lista final que deve ser considerada. Para tal, propõe-se a utilização de um conjunto de requisitos de qualidade.

Os *Requisitos de Qualidade* (R_q) refletem os interesses de um usuário em relação à qualidade mínima que um *matcher* deve prover para ser considerado. Eles permitem que seja definido um valor mínimo de medida de qualidade $Q(m)$ ou até mesmo um valor mínimo por métrica individualmente (e.g., precisão de no mínimo $0,5$). A partir destes requisitos, é possível filtrar a lista de *matchers* fornecida, no intuito de levar em consideração apenas os que possam fornecer resultados mais eficientes em relação à expectativa do usuário.

A Tabela 7 descreve os valores possíveis para os R_q especificados, que são: (i) métrica (e.g., medida de qualidade ($Q(m)$), precisão, cobertura); (ii) valor mínimo; (iii) valor máximo e (iv) tipo (absoluto ou relativo). Desta forma, os requisitos podem definidos com base um número mínimo/máximo ou num intervalo, a partir dos valores obtidos para a métrica escolhida. Esses valores podem ser estabelecidos tanto de maneira absoluta, como relativa. No caso da absoluta, considera-se o valor real das métricas escolhidas (e.g., apenas os *matchers* com precisão mínima de 0,5 serão selecionados). Já no caso da relativa, leva-se em consideração a distribuição de valores para as métricas obtidas pelos demais *matchers* a partir de alinhamentos armazenados no catálogo (e.g., apenas os *matchers* com precisão 50% maior que os demais serão selecionados).

Quadro 7 – Valores possíveis para os Requisitos de qualidade

Métrica	**Valor Mínimo**	**Valor Máximo**	**Tipo**
Medida de Qualidade (Q(m)), Precisão, Cobertura, Medida-F, Tempo de Execução	Número decimal no intervalo [0,1]	Número decimal no intervalo [0,1]	absoluto, relativo

O fluxo para seleção de *matchers* é ilustrado no Algoritmo 4. Ao percorrer a lista fornecida na etapa anterior (denominado M'), verificam-se os que cumprem os requisitos de qualidade (R_q) definidos pelo usuário. A lista (denominada M'') é fornecida como resultado nesta etapa.

Algoritmo 4 Algoritmo para seleção de *matchers* (MS)

```
1: function MS(M', R_q)
   ▷ M' corresponde a uma lista de matchers pré-selecionados.
   ▷ R_q corresponde aos requisitos de qualidade definidos.
2:     M'' ← ∅                          ▷ Inicializa lista final de matchers selecionados
3:     for all m_i ∈ M' do              ▷ Percorre lista de matchers pré-selecionados
4:         for all r_i ∈ R_q do         ▷ Percorre cada requisito de qualidade r
5:             M'' ← M' ∩ r             ▷ Adiciona à lista M' os matchers que contemplem o
                                          requisito de qualidade r_i
6:         end for
7:     end for
8:     return M''                       ▷ Retorna a lista final matchers selecionados
9: end function
```

5.7 EXECUÇÃO DE *MATCHERS* E AVALIAÇÃO DE ALINHAMENTOS

Após a seleção dos *matchers* mais adequados (M'') determinada pelas etapas anteriores, o *framework* dispõe de uma infraestrutura que permite a execução dos *matchers* selecionados a partir de chamadas a serviços. Este recurso é útil para que o usuário possa obter e avaliar alinhamentos gerados pelos *matchers* selecionados, possibilitando a melhora contínua das bases de conhecimento. Outra forma de avaliação do resultado além da intervenção manual, é a utilização de um alinhamento de referência completo. No caso dos experimentos executados nesta pesquisa, como foram utilizados os conjuntos de dados da OAEI, pôde-se utilizar alinhamentos de referência fornecidos pela OAEI para verificar a eficiência dos *matchers* recomendados. A Tabela 15 ilustra a estrutura de um alinhamento de referência, em que se destaca a coluna "Avaliação", que corresponde à definição da correspondência como verdadeira ou falsa.

Tabela 15 – Exemplo de avaliação de alinhamentos por parte do usuário

Ontologia Cmt	**Ontologia OpenConf**	**Relação**	**Valor**	**Matcher**	**Avaliação**
Person	Person	Equivalência	1	COMA	VERDADEIRA
hasAuthor	has_authors	Equivalência	0,72	COMA	VERDADEIRA
submitPaper	Publisher	Equivalência	0,45	LogMap	FALSA
hasConflictOfInterest	has_an_abstract	Equivalência	0,33	LogMap	FALSA

Um desafio relacionado à etapa de execução é que cada *matcher* possui diferentes formas de acesso e funcionamento. Alguns oferecem a opção de navegação a partir de uma interface gráfica, outros disponibilizam APIs que podem ser utilizadas por aplicações de uma linguagem de programação específica e outros oferecem o acesso on-line, via Serviço Web. Logo, para que seja possível executá-los de maneira análoga, torna-se necessário padronizar os dados fornecidos como entrada, os gerados como saída e a forma de interação com o usuário.

Para isso, o *framework* proposto utiliza como entrada arquivos nos formatos mais

adotados para a representação de ontologias (OWL e RDF). Já a saída é padronizada utilizando o modelo de dados proposto pela Alignment API (DAVID et al., 2011), que fornece um padrão para a representação de correspondências e alinhamentos. Em relação à forma de interação com o usuário, adota-se como padrão o modelo de Serviços Web, por possibilitar maior interoperabilidade entre diferentes sistemas, independentemente da linguagem de programação, já que é utilizado JSON ou XML como formato para composição das mensagens.

Desta forma, por exemplo, pode-se fazer uma requisição HTTP POST a um endereço como http://localhost:8888/api/alignment, passando no corpo da mensagem o caminho para os dois arquivos de ontologia (fonte e alvo) e o nome do *matcher* cadastrado no catálogo. Com isso, o referido *matcher* será executado e o resultado será exibido em um formado padronizado, independentemente de quem forneceu o alinhamento.

Uma vez que o alinhamento foi gerado, ele é armazenado no catálogo e recebe uma identificação única. Esse registro pode ainda ser atualizado posteriormente, a partir da avaliação das correspondências fornecidas por intermédio de intervenção manual do usuário ou a partir de uma comparação com alinhamento de referência (*gold-standard*). Para a avaliação, utiliza-se a função AE, definida a seguir.

Definição 5.8. A função AE (*Alignment Evaluation*) recebe um alinhamento A, um alinhamento de referência A_{ref} e fornece uma versão atualizada de A, denominada A', contendo um conjunto de métricas de avaliação, tais como Precisão, Cobertura e Medida-F. A Tabela 16 mostra um exemplo das propriedades que são adicionadas ao alinhamento para representar a sua avaliação.

$$AE : A \times A_{ref} \rightarrow A'$$

Tabela 16 – Exemplo de atributos adicionados ao alinhamento após avaliação

Matcher	**Precisão**	**Cobertura**	**Medida-F**	**Cor. Encontradas**	**Cor. Esperadas**	**Cor. Corretas**
COMA	0,36	0,66	0,47	6	11	4
COMA	0,5	0,71	0,58	10	7	5
LogMap	0,03	0,8	0,06	229	10	8
LogMap	0,01	1,0	0,03	230	4	4

Os atributos apresentados na Tabela 16 representam a qualidade que um *matcher* atingiu ao gerar determinado alinhamento. Como discutido nas seções anteriores, esse resultado poderá ser utilizado para a construção do perfil do *matcher*, desde que as ontologias fornecidas como entrada possuam características compatíveis com as que foram comparadas no alinhamento. A avaliação, seja via intervenção manual do usuário ou a partir da indicação de alinhamentos de referência, é uma etapa importante para o

aperfeiçoamento da base de conhecimento, possibilitando a melhoria contínua do processo de recomendação.

5.8 EXPERIMENTOS E RESULTADOS

Nesta seção são apresentados objetivo, cenários e a análise dos resultados dos experimentos realizados, no que diz respeito à recomendação de *matchers*. Os detalhes de implementação e as formas de acesso a cada serviço disponibilizado pelo framework são descritos no Apêndice B.

5.8.1 Objetivo

O objetivo deste experimento é avaliar a eficiência do mecanismo de recomendação de *matchers* proposto pelo *framework* Redoma, de maneira a mostrar que a sua utilização resulta no aumento da qualidade dos alinhamentos gerados, em comparação com o uso de *matchers* existentes isoladamente. Com isso, busca-se comprovar as hipóteses **H2.1** e **H2.2** descritas abaixo.

H2.1 A geração de alinhamentos utilizando o Redoma apresenta melhorias de qualidade nos resultados, seguindo as métricas de precisão, cobertura e Medida-F, ao ser comparado com *matchers* isoladamente.

H2.2 A combinação de *matchers* a partir do alinhamento de segmentos possibilita melhores resultados quando comparado com o alinhamento entre ontologias inteiras.

5.8.2 Configuração do Ambiente para Execução dos Experimentos

Para a preparação do ambiente dos experimentos, foi necessário inicializar o Catálogo de Ontologias e o Catálogo de *Matchers* com dados suficientes para a condução dos testes. Em relação aos conjuntos de dados de ontologias, foi aproveitado o mesmo utilizado nos experimentos para a avaliação do SOMA (vide Seção 4.4.2), que consiste em ontologias da trilha de conferências da OAEI e de segmentos extraídos com base em requisitos, caracterizando três cenários distintos (C1, C2, C3). Para a inicialização dos *matchers*, foram criadas centenas de registros, advindos de combinações dos *matchers* utilizados nos experimentos anteriores, contemplando todas as possíveis configurações disponibilizadas por cada um, conforme detalhe-se a seguir.

Matchers Escolhidos e suas Combinações

Para a execução dos experimentos, são considerados como base três *matchers* existentes com participações recentes na OAEI: LogMap, YAM, COMA e AML. No entanto, no intuito de gerar mais opções de *matchers* disponíveis, especialmente, para os experimentos

voltados à recomendação de *matchers*, foram realizadas várias combinações de configurações dos matchers COMA e AML, resultando no registro de 135 *matchers* no catálogo.

O *matcher* COMA possui dois parâmetros de configuração básicos: i) Medida de Similaridade e ii) Estratégia de resolução. A primeira diz respeito a qual técnica será aplicada para medir a similaridade, explorando primordialmente aspectos terminológicos, admitindo um total de 11 técnicas conhecidas (e.g., Edit Distance, Jaro-Winkler). Já a segunda está relacionada aos aspectos estruturais da ontologia (e.g., devem ser considerados os nós filhos, pais, vizinhos), admitindo um total de 6 valores. A combinação dos valores possíveis de configuração resultou na geração de 64 versões diferentes do COMA[1].

A configuração do AML é carregada a partir de um arquivo no formato *ini*. O Quadro 8 descreve os parâmetros que compõem o arquivo, juntamente dos valores possíveis considerados para cada um. Esses valores são combinados para gerar um total de 66 versões diferentes[2].

Descrição	**Parâmetro**	**Valores Possíveis para Combinação**
Utilização de Bases de Conhecimento Prévio	bk_sources	all, none
Matchers baseados em técnicas léxicas	word_matchers	auto
Matchers baseados em técnicas terminológicas	string_matchers	global, local, none, auto
Medidas de similaridade terminológicas	string_measures	ISub, Levenstein, Jaro-Winkler, Q-gram
Matchers estruturais	struct_matchers	ancestors, descendants, average, maximum, minimum, none, auto
Parâmetros da geração de alinhamento	match_properties	auto
Tipo de seleção de matcher	selection_types	auto
Reparo de alinhamentos	repair_alignments	false

Quadro 8 – Parâmetros de Configurações possíveis considerados para o AML

A Tabela 17 mostra um detalhamento da totalização dos *matchers* que foram utilizados nos experimentos.

Ressalta-se que apesar do YAM contar apenas como um *matcher*, durante a sua execução ele combina diferentes *matchers* para construir um dedicado à entrada fornecida. Portanto, neste caso é dispensada a necessidade de criar combinações diferentes de parâmetros de configuração para o YAM, visto que isso é ajustado de maneira automática em tempo de execução.

[1] Lista de parâmetros de configuração referentes à cada versão do COMA disponível em: <https://github.com/dass-cin/redoma-resources/tree/master/matchers-configurations/COMA>

[2] Arquivos de configuração referentes à cada versão do AML disponíveis em: <https://github.com/dass-cin/redoma-resources/tree/master/matchers-configurations/AML>

Matcher	Combinações
LogMap	1
YAM*	1
COMA	64
AML	70
Total	136

Tabela 17 – Resumo do total dos *matchers* considerados

Parâmetros de Configuração do Framework

Em relação às configurações para a execução do framework, no que concerne os pesos para os critérios e os requisitos de qualidade para a seleção dos *matchers*, foram definidas as seguintes configurações:

Critérios de Qualidade. Foram consideradas três variações de pesos para as métricas de qualidade. Para tal, em cada variação, definiu-se o peso 1 para uma das métricas (precisão, cobertura, Medida-F) e foi atribuído o peso 0 às demais.

Requisitos de Qualidade. Foi definido como requisito para seleção do melhor *matcher* a quantidade máxima de 1 sistema recomendado. Tal definição justifica-se por facilitar a comparação dos resultados obtidos, visto que em cada execução é conduzida exclusivamente por um *matcher* (o melhor pontuado naquele cenário), o que evita interferências ou geração de ruído na avaliação dos resultados.

5.8.3 Metodologia

Para avaliar a recomendação de *matchers*, primeiramente utilizam-se os *matchers* disponíveis no catálogo para gerar alinhamentos entre todas as ontologias pertencentes a cada um dos cenários (C1, C2, C3). Em seguida, compara-se o desempenho médio obtido para os *matchers* COMA, AML e LogMap, com o desempenho dos *matchers* recomendados pelo *framework* em cada cenário. Para medir a qualidade, são consideradas as métricas de precisão, cobertura e Medida-F.

5.8.4 Análise dos Resultados

Nesta seção apresenta-se a comparação entre o desempenho dos *matchers* AML, COMA e LogMap com os *matchers* recomendados pelo *framework* Redoma[3], em relação à obtenção de alinhamentos entre as ontologias considerados nos cenários C1, C2 e C3.

[3] Relação completa de *matchers* recomendados pelo Redoma para cada cenário (C1, C2, C3) e métrica priorizada (Precisão, Cobertura, Medida-F) disponível em: <https://github.com/dass-cin/redoma-resources/tree/master/recommendations>

A Figura 15 ilustra, por meio de um gráfico, a comparação dos resultados obtidos pelo AML, COMA, LogMap e pelos *matchers* recomendados pelo *framework* Redoma. Neste gráfico, leva-se em consideração a média obtida para medida de precisão alcançada durante a geração de alinhamentos entre as ontologias que compõem os cenários C1, C2 e C3. Neste caso, é possível observar que os alinhamentos fornecidos pelo Redoma obtiveram resultados superiores aos obtidos pelos demais *matchers* considerados. O Apêndice C apresenta a lista com todos os *matchers* recomendados pelo Redoma para cada um dos cenários e métricas consideradas.

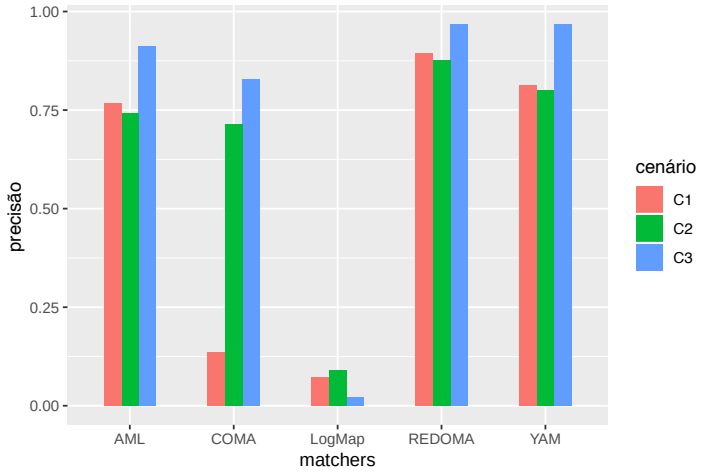


Figura 15 – Resultados obtidos para a geração de alinhamentos considerando a métrica de precisão.

A Figura 16 apresenta um gráfico com uma nova comparação dos resultados obtidos, levando em consideração a medida de cobertura. Note que a geração de alinhamentos utilizando o mecanismo de recomendação do *framework* Redoma também forneceu resultados superiores aos demais *matchers* considerados.

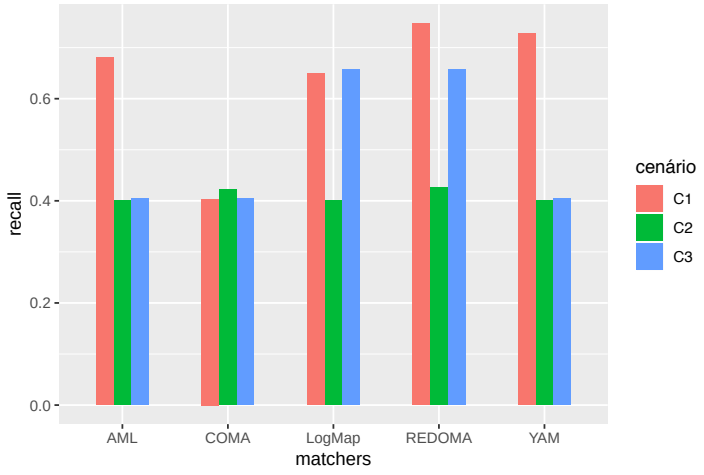


Figura 16 – Resultados obtidos para a geração de alinhamentos considerando a métrica de cobertura.

Por fim, a Figura 17 mostra a comparação dos resultados considerando a Medida-F como critério. Por ser uma medida de harmonização entre precisão e cobertura, como o Redoma obteve melhor desempenho em ambos, o mesmo ocorre ao considerar a média obtida para a Medida-F.

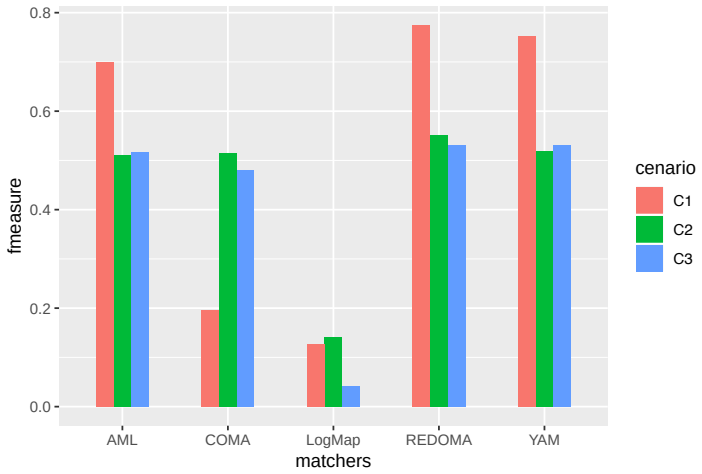


Figura 17 – Resultados obtidos para a geração de alinhamentos considerando a Medida-F.

Observa-se que os resultados obtidos pelo AML e YAM mostram-se bem próximos aos oferecidos pelo Redoma, exceto para o cenário C3 quando é considerada a medida de cobertura, em que o Redoma mostra um desempenho bem superior. Ressalta-se que tanto AML como YAM têm sido ferramentas participantes da OAEI que conseguem obter melhores resultados nesta trilha de conferências, e mesmo assim, foi possível obter ganhos de qualidade. Como o Redoma é um framework de recomendação, ressalta-se que para resolver alinhamentos parciais entre alguns segmentos, foram utilizadas combinações diferentes de configuração inclusive do próprio AML ou YAM, mas também de outros *matchers* (COMA e LogMap), de acordo com a melhor adequação às características dos segmentos envolvidos, o que explica a melhora no desempenho.

Assim, de posse dos resultados apresentados, confirma-se a hipótese de que a geração de alinhamentos utilizando o *framework* Redoma apresenta ganhos de qualidade, considerando as métricas de precisão, cobertura e Medida-F. Um fator primordial para isto é a adoção de segmentos de ontologias durante o processo. Desta forma, torna-se possível dividir uma tarefa de alinhamento envolvendo duas ontologias em vários pares segmentos e utilizar *matchers* distintos para resolver cada um, de acordo com a compatibilidade de características. Com isso, também confirma-se a hipótese de que a utilização de segmentos possibilita uma maior eficiência para a recomendação de *matchers* durante a geração de alinhamentos.

5.9 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o *framework* Redoma, cujo objetivo é oferecer um mecanismo para a geração de alinhamentos de ontologias a partir da recomendação e seleção de *matchers* com base em requisitos do usuário. Inicialmente, foi apresentado o problema da avaliação e recomendação, em que se destacaram alguns desafios, tais como a ausência de indicações claras sobre quais características de ontologias podem favorecer a obtenção de um maior número de alinhamentos corretos, como também a própria necessidade pela especificação das métricas para a avaliação de qualidade, que depende, intrinsecamente, de critérios estabelecidos pelo usuário.

Como forma de abordar esses problemas, utiliza-se o conceito de perfil de ontologias - composto por métricas utilizadas para avaliação - em conjunto com o conceito de perfil de *matchers*, que é gerado dinamicamente, em tempo de execução, com base em alinhamentos gerados anteriormente entre ontologias com as mesmas características (obtidas a partir dos respectivos perfis) das consideradas como entrada. Como forma de especificar os elementos do perfil com base nos requisitos do usuário, ao invés de utilizar as ontologias inteiras, são extraídos segmentos utilizando o algoritmo SOMA.

O *framework* Redoma visa agregar esses conceitos no contexto da geração de alinhamentos, visando a recomendação dos melhores *matchers*. Além da geração dos perfis de ontologias e de *matchers*, o *framework* utiliza, no processo de avaliação e recomendação, o cálculo da medida de qualidade baseada no perfil obtido ao receber duas ontologias, o que resulta em um ranking dos *matchers* mais adequados.

Por fim, a partir da definição de requisitos de qualidade, o usuário poderá selecionar o conjunto final de *matchers* a serem considerados e então, a partir da própria infraestrutura do *framework*, realizar a sua execução, resultando em um conjunto de alinhamentos correspondente a cada *matcher*. Neste momento, permite-se que o usuário possa dar o seu *feedback* em relação à veracidade de cada correspondência fornecida. Tal resultado é armazenado, juntamente com o alinhamento, na base de conhecimento, possibilitando uma melhoria contínua dos processo de recomendação.

No que concerne os experimentos, eles foram realizados de forma a comprovar as hipóteses levantadas neste tese para os problemas de segmentação de ontologias e recomendação de *matchers*. Os resultados foram discutidos e ilustrados por intermédio da apresentação de gráficos. Pelos resultados mostrados, observa-se que:

- A recomendação de *matchers* apresentada no *framework* Redoma mostrou melhores resultados dos alinhamentos gerados em relação outras técnicas existentes, considerando como critério de qualidade as medidas de precisão, cobertura e Medida-F;
- A segmentação de ontologias integrada ao processo de alinhamento possibilitou a combinação de *matchers*, de maneira a viabilizar a utilização dos *matchers* mais adequados para resolver cada alinhamento entre segmentos.

Destaca-se que para a execução dos experimentos foram considerados os pesos máximos para cada critério de qualidade. No entanto, na prática, o usuário poderá definir livremente os valores, o que pode fazer com que outras ferramentas sejam recomendadas durante o processo, buscando obter o melhor resultado que atenda aos critérios prioritários. Por fim, foram utilizadas diversas combinações de configuração para a um mesmo *matcher*, o que também torna o *framework* útil para o problema de calibração (tuning) de *matchers*. Neste caso, ressalta-se que podem ser incluídos *matchers* que utilizem uma combinação de diferentes *matchers* durante a geração das correspondências.

6 CONCLUSÕES E TRABALHOS FUTUROS

Nos últimos anos, o problema da geração de alinhamento de ontologias tem se tornado ainda mais desafiador, em função do aumento do número de fontes e de vocabulários disponíveis para descrever os seus dados. Isto refletiu em um aumento da quantidade de elementos a comparar, além da chance de obtenção de um alto número de correspondências, o que torna a avaliação manual um processo demorado e custoso ao usuário. Neste contexto, este trabalho propõe um algoritmo para a redução do escopo de busca para a geração de alinhamentos, especificamente, para segmentação de ontologias, denominado SOMA, em que são extraídos os elementos mais relevantes para o usuário. Para isto, introduz-se o uso de requisitos da aplicação categorizados em dois tipos: requisitos de dado (um conjunto de palavras-chave que indicam conceitos importantes para o usuário), e requisitos operacionais (um conjunto de parâmetros que influenciam na geração dos segmentos).

No que concerne a obtenção de alinhamentos, propõe-se um *framework*, denominado Redoma, para prover alinhamentos por meio de *matchers* recomendados automaticamente. Buscando uma maior eficiência na recomendação, propõe-se alinhar segmentos de ontologias extraídos pelo SOMA ao invés de ontologias inteiras. Como parte deste *framework*, introduz-se o Catálogo de *Matchers*, responsável por manter um conjunto de *matchers* com a incumbência de encontrar correspondências entre os segmentos fornecidos. Para o processo de recomendação, o *framework* quantifica as características das ontologias fornecidas como entrada e as correlaciona com as ontologias envolvidas em alinhamentos gerados anteriormente, de maneira a avaliar o desempenho do *matcher* apenas nos cenários com a mesma classificação da demanda corrente.

Neste sentido, o algoritmo e o *framework* propostos foram avaliados a partir da realização de experimentos. Para tal, foram implementados protótipos para instanciar e possibilitar a avaliação dos conceitos introduzidos. O objetivo geral dos experimentos conduzidos foi medir o benefício das soluções propostas em relação ao processo de geração de alinhamentos. Durante a execução dos experimentos, foram considerados três cenários contendo ontologias a alinhar, sendo o primeiro composto por um conjunto de ontologias completas provido pela trilha de conferência da OAEI, enquanto que os demais contém segmentos derivados das ontologias da mesma trilha, gerados com base em requisitos distintos. A partir desses cenários, foram conduzidos dois tipos de experimentos, um para medir o benefício de realizar alinhamentos entre segmentos gerados pelo SOMA, em detrimento a considerar as ontologias inteiras, e outro para medir a eficiência dos alinhamentos gerados pelos *matchers* recomendados pelo Redoma.

A partir dos resultados obtidos, é possível observar que o alinhamento de segmentos pôde reduzir significativamente o tempo de execução necessário para a geração de alinhamentos, sem comprometer a qualidade dos resultados obtidos. Além disso, o uso de segmentos pôde

prover uma maior eficiência na recomendação de *matchers* para a geração de alinhamentos, de maneira que supera o desempenho dos *matchers* existentes a partir da utilização dos que foram recomendados pelo Redoma.

Este capítulo é organizado da seguinte maneira: a Seção 6.1 discute as contribuições provenientes desta pesquisa; a Seção 6.2 indica algumas direções pelas quais esta pesquisa poderia ser evoluída; e a Seção 6.3 conclui a tese com algumas observações finais.

6.1 RESUMO DAS CONTRIBUIÇÕES

Embasado nos resultados descritos anteriormente, as principais contribuições deste trabalho estão relacionadas à geração de segmentos de ontologias e à recomendação de *matchers* para a geração de alinhamentos. Neste aspecto, são propostos o algoritmo SOMA e o *framework* Redoma, resultando nas contribuições descritas a seguir.

6.1.1 Contribuições Teóricas

- Especificação do perfil de uma ontologia, contemplando métricas que resultam na classificação de ontologias com base nas características estruturais, sintáticas, léxicas e de instância.
- Definição de propriedades de conservação semântica de ontologias, minimizando a perda semântica dos elementos extraídos na geração de segmentos;
- Especificação de requisitos de dado para determinar os elementos que irão compor os segmentos de ontologias e requisitos operacionais para configurar como os segmentos são gerados.
- Introdução de uma nova métrica de qualidade para os *matchers* (*Q(m)*), agregando métricas existentes (e.g., precisão, cobertura) com a inclusão de pesos para cada uma, de acordo com as preferências do usuário.

6.1.2 Sistemas, Recursos e Aplicações Desenvolvidas

- Algoritmo SOMA (<https://github.com/dass-cin/SOMA>), disponibilizado como uma aplicação baseada em serviços, facilitando a integração da execução do fluxo principal do algoritmo com aplicações externas.
- *Framework* Redoma (<https://github.com/dass-cin/redoma-resources>), disponibilizado como uma aplicação baseada em serviços, facilitando a integração com o mecanismo de recomendação de *matchers* automático por aplicações externas.
- Catálogo de Matchers (<https://github.com/dass-cin/matchers-calalog>), viabilizando a execução de diferentes *matchers* através do acesso padronizado à uma interface de serviços.

- Catálogo de Ontologias (<https://github.com/dass-cin/ontology-catalog>), possibilitando o armazenamento e enriquecimento de ontologias para otimizar a segmentação e facilitar a construção do perfil de uma ontologia.
- Construção de Roteiros para Automatização dos experimentos (<https://github.com/diegoep/expert-systems-experiments>).

6.1.3 Publicações

- PESSOA, D. E. R.; SALGADO, A. ; LÓSCIO, B. Improving Ontology Matching using Application Requirements for Segmenting Ontologies. Journal of Universal Computer Science, 2019.
- PESSOA, D. E. R.. A Framework for Recommending Ontology Matching Systems based on Application Requirements. In: International Semantic Web Conference, 2017, Vienna. Proceedings of the Doctoral Consortium at the 16th International Semantic Web Conference (ISWC 2017) co-located with 16th International Semantic Web Conference, 2017.
- PESSOA, D. E. R.; SALGADO, A.; LÓSCIO, B. Towards a Multi-Layer Architecture for Combination of Schema Matchers. In: 10th Alberto Mendelzon International Workshop on Foundations of Data Management, 2016, Panama City, Panama. Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, Panama City, Panama, Jun 8 - 10, 2016, 2016.
- PESSOA, D. E. R.; OLIVEIRA, M. I. S.; LÓSCIO, B. ; SALGADO, A. A proposal for RDF data integration benchmarking. In: 2nd International Workshop on Benchmarking RDF Systems, 2014, Hangzhou, China. 2nd International Workshop on Benchmarking RDF Systems, 2014.

6.2 TRABALHOS FUTUROS

A partir da perspectiva apresentada ao longo deste texto, destacam-se algumas sugestões de evolução do trabalho, apontadas como possíveis temas para pesquisas futuras, tais como:

- Utilizar os requisitos de dado considerados pelo algoritmo de segmentação como critério para localizar ontologias disponíveis por diversas fontes, de maneira a recuperar apenas as que cumpram os requisitos de dado fornecidos.
- Realizar experimentos com ontologias de domínios de conhecimento contendo elementos e estruturas não-convencionais.

- Avaliar a utilização de mais de um *matcher* como resultado da recomendação, de maneira a identificar a melhor estratégia para realizar a agregação dos alinhamentos gerados por *matchers* diferentes.

- Integrar o catálogo de ontologias com repositórios de ontologias existentes (e.g., Schema.org[1], Bioportal[2]).

- Integrar o catálogo de *matchers* a ferramentas de avaliação de *matchers* utilizados atualmente pela OAEI, como SEALS[3] e HOBBIT (JIMÉNEZ-RUIZ et al., 2018), de maneira a mitigar a limitação que existe pela necessidade de execução dos *matchers* usando a infraestrutura do Redoma.

- Usar histórico de alinhamentos armazenados no catálogo para extrair conjuntos de treinamento para técnicas de Aprendizado de Máquina (*Machine Learning*), de maneira a utilizar diferentes tipos de classificadores para prover a seleção final de *matchers* totalmente automática (sem depender de pesos atribuídos pelo usuário).

- Adicionar suporte à geração de alinhamentos de múltiplas ontologias em um único processo (ao invés de realizar múltiplos alinhamentos par-a-par (1-1)), como a concepção de redes de alinhamentos de esquemas/ontologias abordada por Rodrigues (2018) e Santos, Revoredo e Bai (2018).

- Expandir o número de relacionamentos suportados na geração de segmentos, incluindo os que possam servir para a utilização de antipadrões na geração de alinhamentos (e.g., owl:disjointWith), conforme abordado por Silva, Baiao e Revoredo (2016).

- Desenvolver um estudo de caso com usuários reais de um domínio de conhecimento específico, incluindo a definição de requisitos para a segmentação e alinhamento de ontologias.

6.3 OBSERVAÇÕES FINAIS

Neste trabalho foi apresentado um *framework* (Redoma) que utiliza requisitos da aplicação como forma de prover maior eficiência na recomendação e combinação de *matchers* para a geração de alinhamentos de ontologias. A introdução do *framework* Redoma, fundamentado pela formalização de suas definições básicas, algoritmos implementados e experimentos realizados, mostrou-se como uma solução eficiente para este problema, trazendo benefícios tanto para a redução do tempo da geração de alinhamentos, como também para a melhoria da qualidade dos resultados obtidos.

[1] Acesso ao repositório Schema.org pelo endereço: <http://schema.org>
[2] Acesso ao Bioportal pelo endereço: <http://bioportal.bioontology.org>
[3] Acesso ao SEALS pelo endereço: <http://www.seals-project.eu>

REFERÊNCIAS

ACAMPORA, G.; LOIA, V.; SALERNO, S.; VITIELLO, A. A hybrid evolutionary approach for solving the ontology alignment problem. *International Journal of Intelligent Systems*, v. 27, n. 3, p. 189–216, jan. 2012.

ACHICHI, M.; CHEATHAM, M.; DRAGISIC, Z.; EUZENAT, J.; FARIA, D.; FERRARA, A.; FLOURIS, G.; FUNDULAKI, I.; HARROW, I.; IVANOVA, V.; JIMÉNEZ-RUIZ, E.; KOLTHOFF, K.; KUSS, E.; LAMBRIX, P.; LEOPOLD, H.; LI, H.; MEILICKE, C.; MOHAMMADI, M.; MONTANELLI, S.; PESQUITA, C.; SAVETA, T.; SHVAIKO, P.; SPLENDIANI, A.; STUCKENSCHMIDT, H.; THIÉBLIN, É.; TODOROV, K.; SANTOS, C. Trojahn dos; ZAMAZAL, O. Results of the Ontology Alignment Evaluation Initiative 2017. In: *OM 2017 - 12th ISWC workshop on ontology matching*. Wien, Austria: No commercial editor., 2017. p. 61–113. Achichi2017a. Disponível em: <https://hal.archives-ouvertes.fr/hal-01670238>.

ANAM, S.; KIM, Y. S.; KANG, B. H.; LIU, Q. Adapting a knowledge-based schema matching system for ontology mapping. *ACE/ACSC/AISC/APCMM/AUIC/AWC*, p. 27–10, 2016.

ARAÚJO, T. B.; PIRES, C. E. S.; NOBREGA, T. P. da; NASCIMENTO, D. C. A Parallel Approach for Matching Large-scale Ontologies. *JIDM*, v. 6, n. 1, p. 18–31, 2015.

ARP, R.; SMITH, B.; SPEAR, A. D. *Building Ontologies with Basic Formal Ontology*. [S.l.]: The MIT Press, 2015. ISBN 0262527812, 9780262527811.

AUM"ULLER, D.; DO, H.-H.; MASSMANN, S.; RAHM, E. Schema and ontology matching with COMA++. In: *Proc. ACM SIGMOD Conference*. [S.l.: s.n.], 2005.

BELHAJJAME, K.; PATON, N. W.; FERNANDES, A. A. A.; HEDELER, C.; EMBURY, S. M. *User Feedback as a First Class Citizen in Information Integration Systems*. 2011.

BELLA, G.; GIUNCHIGLIA, F.; MCNEILL, F. Language and domain aware lightweight ontology matching. *J. Web Semant.*, v. 43, p. 1–17, 2017. Disponível em: <https://doi.org/10.1016/j.websem.2017.03.003>.

BELLAHSENE, Z.; NGO, D. H.; BELLAHSENE, Z. YAM++ : (not) Yet Another Matcher for Ontology Matching Task. *Bases de Données Avancées*, p. 5, 2012.

BENERECETTI, M.; BOUQUET, P.; GHIDINI, C. Contextual reasoning distilled. *JETAI*, v. 12, n. 3, p. 279–305, 2000. Disponível em: <citeseer.ist.psu.edu/benerecetti00contextual.html>.

Bernardo Cuenca, G.; IAN, H.; YEVGENY, K.; ULRIKE, S. Modular reuse of ontologies: theory and practice. *Journal of Artificial Intelligence Research*, v. 31, n. 1, p. 273–318, 2008. ISSN 10769757. Disponível em: <https://www.jair.org/media/2375/live-2375-3703-jair.pdf>.

BOUQUET, P.; EHRIG, M.; EUZENAT, J.; FRANCONI, E.; HITZLER, P.; KRÖTZSCH, M.; SERAFINI, L.; STAMOU, G.; SURE, Y.; TESSARIS, S. *Specification of a common framework for characterizing alignment*. [S.l.], 2004.

CHAN, P. K.; SCHLAG, M. D. F.; ZIEN, J. Y. Spectral k-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 13, n. 9, p. 1088–1096, Sep. 1994. ISSN 0278-0070.

CHODOROW, K. *MongoDB: The Definitive Guide*. [S.l.]: O'Reilly Media, Inc., 2013. ISBN 1449344682, 9781449344689.

CORCHO, Ó. *A Layered Declarative Approach to Ontology Translation with Knowledge Preservation*. IOS Press, 2005. v. 116. I-XVIII, 1-149 p. (Frontiers in Artificial Intelligence and Applications, v. 116). ISBN 978-1-58603-477-1. Disponível em: <http://www.iospress.nl/book/a-layered-declarative-approach-to-ontology-translation-with-knowledge-preservation/>.

CRUZ, I. F.; ANTONELLI, F. P.; STROE, C. Agreementmaker: Efficient matching for large real-world schemas and ontologies. *PVLDB*, v. 2, p. 1586–1589, 2009.

CRUZ, I. F.; FABIANI, A.; CAIMI, F.; STROE, C.; PALMONARI, M. Automatic Configuration Selection Using Ontology Matching Task Profiling. *ESWC*, v. 7295, n. Chapter 19, p. 179–194, 2012.

CURINO, C.; JONES, E.; ZHANG, Y.; MADDEN, S. Schism: A workload-driven approach to database replication and partitioning. *Proc. VLDB Endow.*, VLDB Endowment, v. 3, n. 1-2, p. 48–57, set. 2010. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/1920841.1920853>.

DAVID, J.; EUZENAT, J.; SCHARFFE, F.; SANTOS, C. Trojahn dos. The Alignment API 4.0. *Semantic Web ()*, v. 2, n. 1, p. 3–10, jan. 2011.

DO, H.-H.; RAHM, E. Matching large schemas: Approaches and evaluation. *Information Systems*, v. 32, n. 6, p. 857–885, set. 2007.

DO, H.-H.; RAHM, E. Matching large schemas: Approaches and evaluation. *Information Systems*, v. 32, n. 6, p. 857–885, set. 2007.

DOAN, A.; HALEVY, A.; IVES, Z. *Principles of data integration*. [S.l.]: Access Online via Elsevier, 2012.

DOAN, A.; MADHAVAN, J.; DOMINGOS, P.; HALEVY, A. Learning to map between ontologies on the semantic web. In: *WWW '02: Proceedings of the 11th international conference on World Wide Web*. New York, NY, USA: ACM, 2002. p. 662–673. ISBN 1-58113-449-5.

DONG, X.; GABRILOVICH, E.; HEITZ, G.; HORN, W.; LAO, N.; MURPHY, K.; STROHMANN, T.; SUN, S.; ZHANG, W. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: GOOGLE LLC, MOUNTAIN VIEW, UNITED STATES. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.], 2014. p. 601–610.

DONG, X.; SRIVASTAVA, D. *Big Data Integration*. Morgan & Claypool Publishers, 2015. (Synthesis Lectures on Data Management). ISBN 9781627052245. Disponível em: <https://books.google.com.br/books?id=p7d1BwAAQBAJ>.

DRUMM, C.; SCHMITT, M.; DO, H.-H.; RAHM, E. Quickmig: Automatic schema matching for data migration projects. In: *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2007. (CIKM '07), p. 107–116. ISBN 978-1-59593-803-9. Disponível em: <http://doi.acm.org/10.1145/1321440.1321458>.

DUCHATEAU, F. *Towards a Generic Approach for Schema Matcher Selection: Leveraging User Pre- and Post-match Effort for Improving Quality and Time Performance. (Une Approche Générique pour la Sélection d'Outils de Découverte de Correspondances entre Schémas)*. Tese (Doutorado) — Montpellier 2 University, France, 2009.

DUCHATEAU, F.; BELLAHSENE, Z. YAM: A Step Forward for Generating a Dedicated Schema Matcher. *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXV - Volume 9620*, v. 9620, dez. 2015.

EUZENAT, J. Towards a Principled Approach to Semantic Interoperability. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Ontologies and Information Sharing*. [S.l.: s.n.], 2001. p. 19–25.

EUZENAT, J.; FERRARA, A.; HOLLINK, L.; ISAAC, A.; JOSLYN, C.; MALAISÉ, V.; MEILICKE, C.; NIKOLOV, A.; PANE, J.; SABOU, M.; SCHARFFE, F.; SHVAIKO, P.; SPILIOPOULOS, V.; STUCKENSCHMIDT, H.; ŠVÁB-ZAMAZAL, O.; SVÁTEK, V.; TROJAHN, C.; VOUROS, G.; WANG, S. Results of the ontology alignment evaluation initiative 2009. In: AEGEAN UNIVERSITY. *OM'09: Proceedings of the 4th International Conference on Ontology Matching - Volume 551*. [S.l.]: CEUR-WS.org, 2009. p. 73–126.

EUZENAT, J.; SHVAIKO, P. *Ontology Matching*. 2nd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2013. ISBN 3642387209, 9783642387203.

FAHAD, M. Initial results for Ontology Matching workshop 2015 DKP-AOM : results for OAEI 2015. *CEUR Workshop Proceedings*, 2015.

FALBO, R.; aO, F. B.; LOPES, M.; GUIZZARDI, G. The role of foundational ontologies for domain ontology engineering: An industrial case study in the domain of oil and gas exploration and production. *Int. J. Inf. Syst. Model. Des.*, IGI Global, Hershey, PA, USA, v. 1, n. 2, p. 1–22, abr. 2010. ISSN 1947-8186. Disponível em: <http://dx.doi.org/10.4018/jismd.2010040101>.

FARIA, D.; PESQUITA, C.; SANTOS, E.; PALMONARI, M.; CRUZ, I. F.; COUTO, F. M. The agreementmakerlight ontology matching system. In: _____. *On the Move to Meaningful Internet Systems: OTM 2013 Conferences: Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

FREUND, Y.; SCHAPIRE, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.*, v. 55, n. 1, p. 119–139, 1997.

GAL, A.; ANABY-TAVOR, A.; TROMBETTA, A. Evaluating Matching Algorithms: the Monotonicity Principle. *Proc. Workshop on Information Integration on the Web*, p. 47–52, 2003.

GAL, A.; SAGI, T. Tuning the ensemble selection process of schema matchers. *Information Systems*, v. 35, n. 8, p. 845–859, dez. 2010.

GANDON, F. A Survey of the First 20 Years of Research on Semantic Web and Linked Data. *Revue des Sciences et Technologies de l'Information - Série ISI : Ingénierie des Systèmes d'Information*, Lavoisier, dez. 2018. Disponível em: <https://hal.inria.fr/hal-01935898>.

GANGEMI, A.; GUARINO, N.; MASOLO, C.; OLTRAMARI, A.; SCHNEIDER, L. Sweetening ontologies with dolce. In: _____. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web: 13th International Conference, EKAW 2002 Sigüenza, Spain, October 1–4, 2002 Proceedings*. Berlin, Heidelberg: Springer, 2002. p. 166–181. ISBN 978-3-540-45810-4. Disponível em: <http://link.springer.de/link/service/series/0558/bibs/2473/24730166.htm>.

GIUNCHIGLIA, F.; AUTAYEU, A.; PANE, J. S-Match: An open source framework for matching lightweight ontologies. *Semantic Web ()*, v. 3, n. 3, p. 307–317, 2012.

GLIMM, B.; HORROCKS, I.; MOTIK, B.; STOILOS, G.; WANG, Z. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, v. 53, n. 3, p. 245–269, 2014. ISSN 15730670.

GRAU, B. C.; PARSIA, B.; SIRIN, E.; KALYANPUR, A. Modularity and Web Ontologies. *Tenth International Conference on Principles of Knowledge Representation and Reasoning KR2006*, p. 198–209, 2006. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.98.9469>.

GUARINO, N.; OBERLE, D.; STAAB, S. What Is an Ontology? *Handbook on Ontologies*, 2009.

GUIZZARDI, G. *Ontological foundations for structural conceptual models*. Tese (Doutorado) — University of Twente, 10 2005.

GUIZZARDI, G.; WAGNER, G.; ALMEIDA, J. P. A.; GUIZZARDI, R. S. Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. *Applied Ontology*, v. 10, n. 3-4, p. 259–271, 2015. ISSN 18758533.

HAMEED, A.; PREECE, A.; SLEEMAN, D. Ontology reconciliation. In: STAAB, S.; STUDER, R. (Ed.). *Handbook on Ontologies in Information Systems*. Springer Verlag, 2003. p. 231–250. Disponível em: <http://eprints.aktors.org/332/>.

HEATH, T.; BIZER, C. *Linked Data: Evolving the Web into a Global Data Space*. [s.n.], 2011. v. 1. 1–136 p. ISSN 2160-4711. ISBN 9781608454303. Disponível em: <http://www.morganclaypool.com/doi/abs/10.2200/S00334ED1V01Y201102WBE001>.

HORROCKS, I.; PATEL-SCHNEIDER, P. F.; BOLEY, H.; TABET, S.; GROSOF, B.; DEAN, M. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. [S.l.], 2004. Disponível em: <http://www.w3.org/Submission/SWRL>.

HUZA, M.; HARZALLAH, M.; TRICHET, F. Ontomas: a tutoring system dedicated to ontology matching. In: GONÇALVES, R. J.; MÜLLER, J. P.; MERTINS, K.; ZELM, M. (Ed.). *Enterprise Interoperability II*. London: Springer London, 2007. p. 377–388. ISBN 978-1-84628-858-6.

IVANOVA, V.; LAMBRIX, P.; ÅBERG, J. Requirements for and Evaluation of User Support for Large-Scale Ontology Alignment. In: *The Semantic Web. Latest Advances and New Domains*. Cham: Springer International Publishing, 2015. p. 3–20.

JAURO, F.; JUNAIDU, S.; ABDULLAHI, S. Falcon-AO++ - An Improved Ontology Alignment System. *International Journal of Computer Applications*, v. 94, n. 2, p. 1–7, 2014.

JIMÉNEZ-RUIZ, E.; GRAU, B. C. LogMap: Logic-Based and Scalable Ontology Matching. In: *The Semantic Web – ISWC 2011*. Berlin, Heidelberg: Springer, Berlin, Heidelberg, 2011. p. 273–288.

JIMÉNEZ-RUIZ, E.; SAVETA, T.; ZAMAZAL, O.; HERTLING, S.; RÖDER, M.; FUNDULAKI, I.; NGOMO, A.-C. N.; SHERIF, M. A.; ANNANE, A.; BELLAHSENE, Z.; YAHIA, S. B.; DIALLO, G.; FARIA, D.; KACHROUDI, M.; KHIAT, A.; LAMBRIX, P.; LI, H.; MACKEPRANG, M.; MOHAMMADI, M.; RYBINSKI, M.; BALASUBRAMANI, B. S.; TROJAHN, C. Introducing the hobbit platform into the ontology alignment evaluation campaign. In: *Proceedings of the Ontology Matching Workshop 2018*. [s.n.], 2018. Disponível em: <http://disi.unitn.it/~pavel/om2018/papers/om2018_LTpaper5.pdf>.

KILGARRIFF, A.; FELLBAUM, C. WordNet: An Electronic Lexical Database. *Language*, v. 76, n. 3, p. 706, 2000.

KIM, Y. S.; COMPTON, P.; KANG, B. H. Ripple-Down Rules with Censored Production Rules. In: *Knowledge Management and Acquisition for Intelligent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 175–187.

KLEIN, M. Combining and relating ontologies: an analysis of problems and solutions. In: GOMEZ-PEREZ, A.; GRUNINGER, M.; STUCKENSCHMIDT, H.; USCHOLD, M. (Ed.). *Workshop on Ontologies and Information Sharing, IJCAI'01*. Seattle, USA: [s.n.], 2001.

LAMBRIX, P.; KALIYAPERUMAL, R. A session-based ontology alignment approach enabling user involvement. *Semantic Web*, v. 8, n. 2, p. 225–251, 2017. ISSN 22104968.

LAMBRIX, P.; TAN, H.; LIU, Q. SAMBO and SAMBOdtf results for the ontology alignment evaluation initiative 2008. *CEUR Workshop Proceedings*, v. 431, p. 190–198, 2008. ISSN 16130073.

LEE, Y.; SAYYADIAN, M.; DOAN, A.; ROSENTHAL, A. S. eTuner: tuning schema matching software using synthetic scenarios. *The VLDB Journal — The International Journal on Very Large Data Bases*, v. 16, n. 1, p. 97–122, jan. 2007.

LEHMANN, J.; ISELE, R.; JAKOB, M.; JENTZSCH, A.; KONTOKOSTAS, D.; MENDES, P. N.; HELLMANN, S.; MORSEY, M.; KLEEF, P. van; AUER, S.; BIZER, C. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, v. 6, n. 2, p. 167–195, 2015.

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. Disponível em: <http://doi.acm.org/10.1145/2831270>.

LI, J.; TANG, J.; LI, Y.; LUO, Q. Rimom: A dynamic multistrategy ontology alignment framework. *IEEE Trans. on Knowl. and Data Eng.*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 21, n. 8, p. 1218–1232, ago. 2009. ISSN 1041-4347. Disponível em: <http://dx.doi.org/10.1109/TKDE.2008.202>.

LOZANO, J.; CARBONERA, J.; ABEL, M.; PIMENTA, M. Ontology View Extraction: An Approach Based on Ontological Meta-properties. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, v. 2014-Decem, p. 122–129, 2014. ISSN 10823409.

MALIK, J.; BELONGIE, S.; LEUNG, T.; SHI, J. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, v. 43, n. 1, p. 7–27, Jun 2001. ISSN 1573-1405. Disponível em: <https://doi.org/10.1023/A:1011174803800>.

MARIE, A.; GAL, A. Boosting Schema Matchers. In: *On the Move to Meaningful Internet Systems: OTM 2008*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 283–300.

MASSMANN, S.; RAUNICH, S.; AUMÜLLER, D.; ARNOLD, P.; RAHM, E. Evolution of the COMA match system. In: UNIVERSITY OF LEIPZIG. *OM'11: Proceedings of the 6th International Conference on Ontology Matching - Volume 814*. [S.l.]: CEUR-WS.org, 2011. p. 49–60.

MELNIK, S.; GARCIA-MOLINA, H.; RAHM, E. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. *ICDE*, p. 117–128, 2002.

MITCHELL, T. M. *Machine Learning*. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

MOCHOL, M.; JENTZSCH, A. Towards a rule-based matcher selection. In: GANGEMI, A.; EUZENAT, J. (Ed.). *Knowledge Engineering: Practice and Patterns*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 109–119. ISBN 978-3-540-87696-0.

MOCHOL, M.; JENTZSCH, A.; EUZENAT, J. Applying an analytic method for matching approach selection. In: FREE UNIVERSITY OF BERLIN. *OM'06: Proceedings of the 1st International Conference on Ontology Matching - Volume 225*. [S.l.]: CEUR-WS.org, 2006. p. 37–48.

MORK, P.; ALLEN, D.; SMITH, K.; SELIGMAN, L.; MORSE, M.; ROSENTHAL, A.; WOLF, C. The Role of Schema Matching in Large Enterprises. *arXiv*, cs.DB, 2009.

NAUMANN, F. Data profiling revisited. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 42, n. 4, p. 40–49, fev. 2014. ISSN 0163-5808. Disponível em: <http://doi.acm.org/10.1145/2590989.2590995>.

NILES, I.; PEASE, A. Towards a standard upper ontology. In: *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*. New York, NY, USA: ACM, 2001. (FOIS '01), p. 2–9. ISBN 1-58113-377-4. Disponível em: <http://doi.acm.org/10.1145/505168.505170>.

OCHIENG, P.; KYANDA, S. A K-way spectral partitioning of an ontology for ontology matching. *Distributed and Parallel Databases*, Springer US, v. 36, n. 4, p. 643–673, 2018. ISSN 15737578. Disponível em: <https://doi.org/10.1007/s10619-018-7222-8>.

OCHIENG, P.; KYANDA, S. A statistically-based ontology matching tool. *Distributed and Parallel Databases*, Springer US, v. 36, n. 1, p. 195–217, 2018. ISSN 15737578.

OCHIENG, P.; KYANDA, S. Large-Scale Ontology Matching. *ACM Computing Surveys*, v. 51, n. 4, p. 1–35, 2018. ISSN 03600300. Disponível em: <http://dl.acm.org/citation.cfm?doid=3236632.3211871>.

OTERO-CERDEIRA, L.; RODRÍGUEZ-MARTÍNEZ, F. J.; GÓMEZ-RODRÍGUEZ, A. Ontology matching: A literature review. *Expert Systems with Applications*, Elsevier, 2014.

PESQUITA, C.; FARIA, D.; STROE, C.; SANTOS, E.; CRUZ, I. F.; COUTO, F. M. What's in a 'nym'? synonyms in biomedical ontology matching. In: ALANI, H.; KAGAL, L.; FOKOUE, A.; GROTH, P.; BIEMANN, C.; PARREIRA, J. X.; AROYO, L.; NOY, N.; WELTY, C.; JANOWICZ, K. (Ed.). *The Semantic Web – ISWC 2013*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 526–541. ISBN 978-3-642-41335-3.

PEUKERT, E.; EBERIUS, J.; RAHM, E. A Self-Configuring Schema Matching System. *ICDE*, p. 306–317, 2012.

PEUKERT, E.; MASSMANN, S.; KOENIG, K. Comparing Similarity Combination Methods for Schema Matching. *Gi jahrestagung (1)*, 2010.

PFAFF, M.; NEUBIG, S.; KRCMAR, H. Ontology for Semantic Data Integration in the Domain of IT Benchmarking. *Journal on Data Semantics*, Springer Berlin Heidelberg, v. 7, n. 1, p. 29–46, 2018. ISSN 18612040.

POKORNY, J. Nosql databases: A step to database scalability in web environment. In: *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*. New York, NY, USA: ACM, 2011. (iiWAS '11), p. 278–283. ISBN 978-1-4503-0784-0. Disponível em: <http://doi.acm.org/10.1145/2095536.2095583>.

QUEIROZ-SOUSA, P. O.; SALGADO, A. C.; PIRES, C. E. S. A Method for Building Personalized Ontology Summaries. *JIDM*, 2013.

RAHM, E. Towards large-scale schema and ontology matching. In: BELLAHSENE, Z.; BONIFATI, A.; RAHM, E. (Ed.). *Schema Matching and Mapping*. Heidelberg: Springer, 2011. cap. 1, p. 3–27.

RAHM, E.; PEUKERT, E.; EBERIUS, J. Rule-based construction of matching processes. In: *the 20th ACM international conference*. New York, New York, USA: ACM Press, 2011. p. 2421.

RICHARDSON, L.; AMUNDSEN, M.; RUBY, S. *RESTful Web APIs*. [S.l.]: O'Reilly Media, Inc., 2013. ISBN 1449358063, 9781449358068.

RODRIGUES, D. *A Study on Machine Learning Techniques for the Schema Matching Networks Problem*. Tese (Doutorado) — Programa de Pós-graduação em Informática, 2018. Instituto de Computação. Disponível em: <https://tede.ufam.edu.br/handle/tede/6801>.

RODRIGUES, D.; SILVA, A. S. da; RODRIGUES, R.; SANTOS, E. dos. Using active learning techniques for improving database schema matching methods. *IJCNN*, p. 1–8, 2015.

SANTOS, E.; FARIA, D.; PESQUITA, C.; COUTO, F. M. Ontology alignment repair through modularization and confidence-based heuristics. *PLOS ONE*, Public Library of Science, v. 10, n. 12, p. 1–19, 12 2016. Disponível em: <https://doi.org/10.1371/journal.pone.0144807>.

SANTOS, F.; REVOREDO, K.; BAI, F. A Proposal for Optimizing Internetwork Matching of Ontologies. p. 1–5, 2018.

SARAMAGO, J. *A Jangada de Pedra*. [S.l.]: Companhia das Letras, 2006.

SATHIYA, B.; GEETHA, T.; SUGUMARAN, V. A metric based automatic selection of ontology matchers using bootstrapped patterns. *Journal of Web Engineering*, v. 16, n. 7-8, p. 619–652, 2017. ISSN 15409589.

SCHAPIRE, R. E. The Strength of Weak Learnability. *Machine Learning ()*, 1990.

SEDDIQUI, M. H.; AONO, M. An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size. *Web Semantics: Science, Services and Agents on the World Wide Web*, v. 7, n. 4, p. 344–356, 2009.

SEIDENBERG, J.; SEIDENBERG, J.; RECTOR, A.; RECTOR, A. Web ontology segmentation: analysis, classification and use. *WWW '06: Proceedings of the 15th international conference on World Wide Web*, p. 13–22, 2006. ISSN 0893-6080. Disponível em: <http://doi.acm.org/10.1145/1135777.1135785>.

SELLAMI, S.; BOUCELMA, O. Web services discovery and composition: A schema matching approach. In: *ICWS*. IEEE Computer Society, 2011. p. 706–707. ISBN 978-1-4577-0842-8. Disponível em: <http://dblp.uni-trier.de/db/conf/icws/icws2011.html#SellamiB11>.

SHVAIKO, P.; EUZENAT, J. A Survey of Schema-Based Matching Approaches. v. 3730, n. Chapter 5, p. 146–171, 2005.

SHVAIKO, P.; EUZENAT, J. Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 25, n. 1, p. 158–176, 2013.

SHVAIKO, P.; SHVAIKO, P.; EUZENAT, J. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, v. 25, n. 1, p. 158–176, 2013.

SILVA, J.; BAIAO, F. A.; REVOREDO, K. Interactive ontology alignment using alignment antipatterns: A first experiment. In: . Porto Alegre, Brazil, Brazil: Brazilian Computer Society, 2016. (SBSI 2016), p. 28:208–28:215.

SOMMERVILLE, I. *Software Engineering*. 9th. ed. USA: Addison-Wesley Publishing Company, 2010.

SPIELMAN, D. A.; TENG, S.-H. Spectral partitioning works: Planar graphs and finite element meshes. *Linear Algebra and its Applications*, v. 421, n. 2, p. 284 – 305, 2007. ISSN 0024-3795. Special Issue in honor of Miroslav Fiedler. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0024379506003454>.

VIRGILIO, R. Semantic Search over the Web. *Data-Centric Systems and Applications*, p. Online–Ressource TS – WorldCat, 2012.

WACHE, H.; VOEGELE, T.; VISSER, T.; STUCKENSCHMIDT, H.; SCHUSTER, H.; NEUMANN, G.; HUEBNER, S. Ontology-based integration of information - a survey of existing approaches. In: STUCKENSCHMIDT, H. (Ed.). *IJCAI-01 Workshop: Ontologies and Information*. [s.n.], 2001. p. 108–117. Disponível em: <http://www.cs.vu.nl/~heiner/public/ois-2001.pdf>.

XUE, X.; CHEN, J. Optimizing ontology alignment through hybrid population-based incremental learning algorithm. *Memetic Computing*, Springer Berlin Heidelberg, p. 1–9, 2018. ISSN 18659292. Disponível em: <https://doi.org/10.1007/s12293-018-0255-8>.

XUE, X.; PAN, J.-S. A segment-based approach for large-scale ontology matching. *Knowledge and Information Systems*, v. 52, n. 2, p. 467–484, 2017.

XUE, X.; WANG, Y. Optimizing ontology alignments through a memetic algorithm using both matchfmeasure and unanimous improvement ratio. *Artificial Intelligence*, v. 223, p. 65 – 81, 2015. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0004370215000399>.

YANG, P.; WANG, P.; JI, L.; CHEN, X.; HUANG, K.; YU, B. Ontology matching tuning based on particle swarm optimization: Preliminary results. In: ZHAO, D.; DU, J.; WANG, H.; WANG, P.; JI, D.; PAN, J. Z. (Ed.). *The Semantic Web and Web Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 146–155. ISBN 978-3-662-45495-4.

ZHENG, Q.; SHAO, C.; LI, J.; WANG, Z.; HU, L. RiMOM2013 results for OAEI 2013. *CEUR Workshop Proceedings*, v. 1111, p. 161–168, 2013. ISSN 16130073.

APÊNDICE A – IMPLEMENTAÇÃO DO ALGORITMO SOMA

Conforme foi apresentado em detalhes no Capítulo 4, o SOMA é um algoritmo de segmentação de ontologias baseado em requisitos fornecidos pelo usuário. O seu funcionamento ocorre em três etapas: i) Importação de ontologias no catálogo e coleta de sinônimos; ii) Validação de requisitos e segmentação da ontologia fonte e ii) Geração de segmento da ontologia alvo. Em todas as etapas é demandado o acesso a uma base de dados denominada catálogo de ontologias. Neste apêndice, são descritos detalhes ligados aos aspectos de implementação do algoritmo e do catálogo de ontologias.

A.1 IMPLEMENTAÇÃO DO CATÁLOGO DE ONTOLOGIAS

O protótipo do catálogo de ontologias foi concebido na linguagem Java[1]. Como é um repositório de ontologias que utiliza a estrutura de grafos para o armazenamento, o catálogo utiliza a OWL API[2] para ler/escrever ontologias e um banco de dados de grafos (Neo4J[3]) para guardar e buscar no grafo extraído a partir das ontologias. A opção pelo Neo4j foi motivada pela facilidade na manipulação de grafos devido à linguagem de consultas Cypher[4] como também o suporte nativo a diversos algoritmos de manipulação de grafos existentes na literatura[5].

Abaixo são descritas as entidades que compõem o modelo de dados do catálogo de ontologias, além de ser apresentada a sua arquitetura de componentes e, por fim, descreve-se a API contendo os serviços oferecidos.

Modelo de Dados

A Figura 18 descreve o modelo de dados considerado para o catálogo de ontologias. São considerados os seguintes tipos de elementos no catálogo:

Palavra-chave (*Keyword*). representam termos que podem estar relacionados a classes ou propriedades (e.g., sinônimos);

Classe (*Class*). representam os conceitos (e.g., Conferência, Artigo, Pessoa), e possuem relacionamentos de subclasse (isA) com outras classes, domain e range com propriedades e hasKeywords com nós que representam palavras-chaves relacionadas à classe.

[1] Código-fonte do catálogo de ontologias disponível em <https://github.com/dass-cin/ontology-catalog>
[2] OWL API disponível em <http://owlcs.github.io/owlapi/>
[3] Neo4J disponível em: <https://neo4j.com>
[4] Documentação da linguagem Cypher disponível em: <https://neo4j.com/developer/cypher/>
[5] Lista de algoritmos suportados pelo Neo4j disponível em: <https://neo4j.com/docs/graph-algorithms/current/>

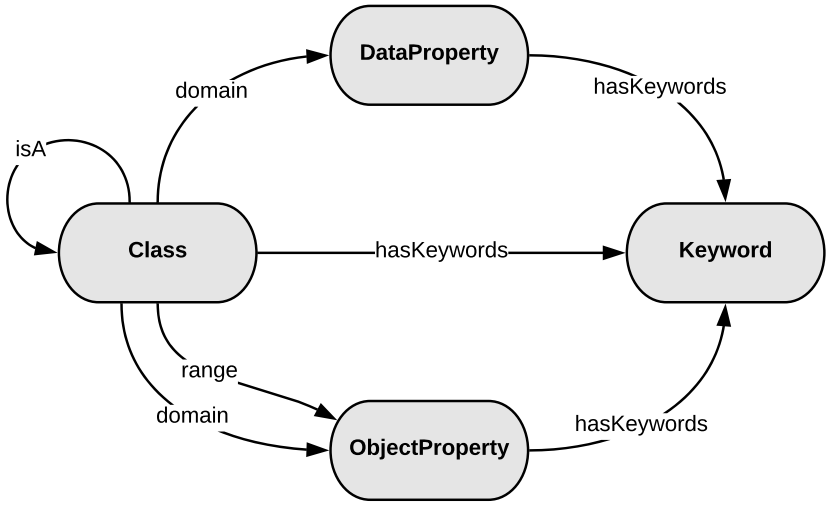


Figura 18 – Ilustração do modelo de dados do Catálogo de Ontologias.

Propriedade de Dado (*DataProperty*). representam as propriedades que se relacionam a tipos primitivos (e.g., título, nome, idade), e possuem relacionamentos domain com classes e hasKeywords com nós que representam palavras-chaves relacionadas à propriedade de dado;

Propriedade de Objeto (*ObjectProperty*). representam as propriedades que relacionam duas classes (e.g., hasAuthor, reviewedBy), e possuem relacionamentos domain e range para classes e hasKeywords com nós que representam palavras-chave relacionadas à propriedade de objeto.

Visão Geral da Arquitetura de Componentes

A Figura 19 descreve a arquitetura de componentes do Catálogo de Ontologias, que é organizada em duas camadas: i) serviços - que concentra a lógica de execução e ii) repositórios - que abstrai o acesso à base de dados Neo4j.

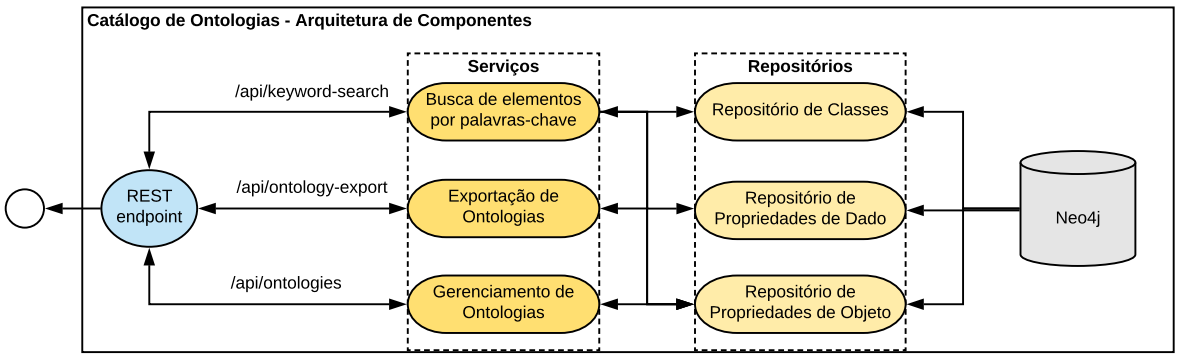


Figura 19 – Arquitetura de Serviços do Catálogo de Ontologias.

A camada de serviços é composta pelos seguintes componentes:

Busca de Elementos por Palavras-Chave. Oferece serviços para a recuperação de nós do grafo (e.g., classes, propriedades) relacionados a um conjunto de palavras-chave;

Exportação de Ontologias. Fornece a opção de exportar um grafo de ontologia (ou segmento) como arquivo de ontologia no formato OWL;

Gerenciamento de Ontologias. Disponibiliza serviços para a importação, listagem e busca a partir de consultas Cypher.

A camada de repositórios oferece métodos para listagem/inserção/atualização/remoção de objetos contidos na base de dados. É definido um componente de repositório próprio para cada tipo de objeto gerenciado (classes, propriedades de dados e propriedades de objetos).

A.2 DOCUMENTAÇÃO DA API DE SERVIÇOS DO CATÁLOGO DE ONTOLOGIAS

A Figura 20 apresenta a interface gráfica disponibilizada para visualização da documentação on-line dos serviços oferecidos pelo catálogo de ontologias. Essa interface é ofertada no mesmo endereço base de acesso aos serviços. Ao clicar em cada item, o usuário poderá visualizar detalhes dos tipos de dados de entrada e saída. Para a construção dessa interface, foi utilizada a biblioteca Swagger[6]. Em seguida descrevem-se em detalhes cada um dos serviços ofertados pelos componentes do catálogo.

Serviços do Componente de Busca de Elementos Por Palavras-Chave

GET /api/keywords-search *(Elemento por Palavras-chave).* Recebe como entrada uma palavra-chave descrevendo um termo referente a uma classe ou propriedade de ontologia e retorna o elemento mais próximo desse conceito em todas as ontologias. Esse serviço é útil para identificar qual elemento em uma determinada ontologia de domínio de conhecimento específico é capaz de representar um determinado conceito.

GET /api/keyword-search-list/{ontologyName} *(Elementos por Palavras-chave e Ontologia).* Recebe como entrada o identificador único de uma ontologia e uma palavra-chave descrevendo um termo referente a uma classe ou propriedade e retorna o elemento da ontologia mais próximo do fornecido. Um caso de uso para este serviço seria a identificação do elemento de uma ontologia específica que melhor contemple um determinado termo.

GET /api/keyword-search/{ontologyName} *(Elementos por Palavras-chaves e Ontologia).*) Recebe como entrada o identificador único de uma ontologia e uma

[6] <https://swagger.io>

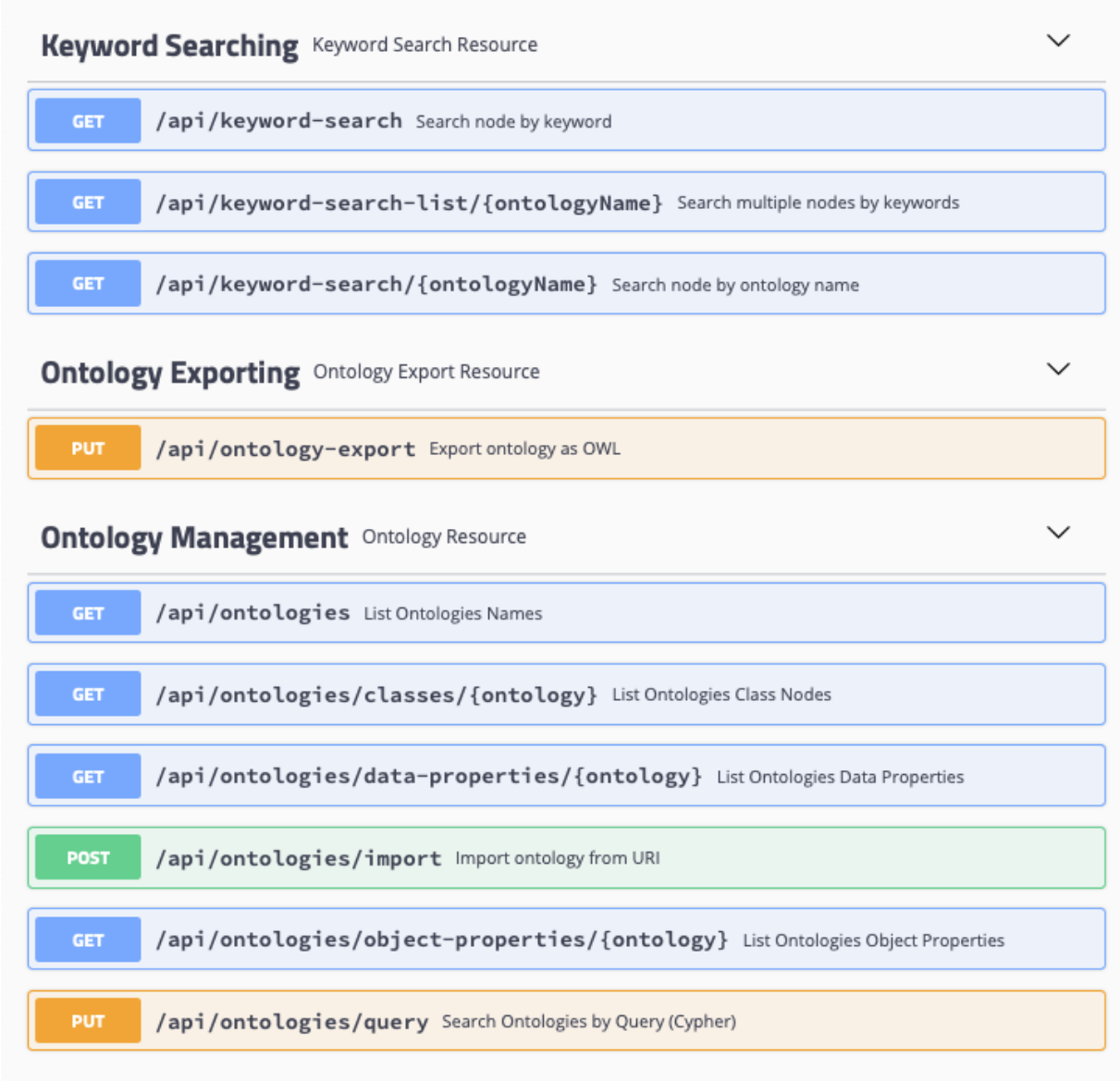


Figura 20 – Interface Gráfica da API do Catálogo de Ontologias.

palavra-chave descrevendo um termo referente a uma classe ou propriedade e retorna um conjunto de elementos que possuam algum tipo de relação semântica com o referido termo. Uma utilidade para este serviço seria na extração dos elementos de uma ontologia, ao considerar apenas os que estão relacionados ao fornecido.

Serviços do Componente de Exportação de Ontologias

PUT /api/ontology-export ***(Exportação de Ontologia).*** Recebe a identificação única de uma ontologia, um caminho no sistema de arquivos e uma consulta Cypher para o caso da extração de um segmento e exporta o resultado como arquivo OWL no caminho indicado. No escopo do SOMA, este serviço é útil para a exportação dos segmentos gerados, que podem ser utilizados como entrada posteriormente em um *matcher*.

Serviços do Componente de Gerenciamento de Ontologias

GET /api/ontologies ***(Listagem de nomes de ontologias).*** Lista os nomes completos (identificadores únicos) de todas as ontologias armazenadas no catálogo. Esse serviço funciona como um índice das ontologias disponíveis.

GET /api/ontologies/classes/{ontology} ***(Classes de uma Ontologia).*** Recebe a identificação única de uma ontologia armazenada e lista as suas classes.

GET /api/ontologies/data-properties/{ontology} ***(Propriedades de Dado de uma Ontologia).*** Recebe a identificação única de uma ontologia armazenada e lista as propriedades de dado.

POST /api/ontologies/import ***(Importação de Ontologia por URI).*** Recebe uma URI indicando o endereço de uma ontologia a ser importada. Essa URI pode descrever um caminho no sistema de arquivos local ou uma referência para um endereço da Web.

GET /api/ontologies/object-properties/{ontology} ***(Propriedades de Objeto de uma Ontologia).*** Recebe a identificação única de uma ontologia armazenada e lista as suas propriedades de objeto.

PUT /api/ontologies/query ***(Busca de Elementos por Execução de Consulta).*** Recebe uma consulta Cypher e retorna o conjunto de elementos resultante da sua execução. No contexto do SOMA, essa consulta é gerada automaticamente a partir da busca por palavras-chave e é utilizada para a extração de segmentos das ontologias.

Os serviços ofertados pelo catálogo são utilizados durante o fluxo de execução do algoritmo SOMA. Esse uso é ilustrado nas seções a seguir, em que descrevem-se os aspectos referentes à implementação de cada etapa do algoritmo.

IMPLEMENTAÇÃO DO FLUXO PRINCIPAL DO ALGORITMO SOMA

Para a implementação do algoritmo SOMA foi construído um protótipo desenvolvido na linguagem Java[7]. O modelo adotado foi o de programação distribuída baseada na troca de mensagens. A comunicação entre os diferentes componentes que compõem o protótipo é realizada através de requisições HTTP REST (RICHARDSON; AMUNDSEN; RUBY, 2013). Essa escolha se motivou por ser um modelo interoperável, tornando possível a implementação de componentes em linguagens de programação distintas. Na seção seguinte, detalham-se os aspectos de implementação do catálogo de ontologias. Adiante são descritos os detalhes envolvendo a implementação de cada etapa da execução do algoritmo SOMA.

[7] Plataforma de Desenvolvimento em Java disponível em <https://www.oracle.com/technetwork/pt/java/index.html>

Processo de importação de ontologias e coleta de sinônimos

O processo de importação inicia-se a partir do fornecimento de uma URI contendo um caminho para ontologia a ser importado no catálogo. A partir disso, o componente de Gerenciamento de Ontologias é acionado, e realiza-se o processo a partir das seguintes etapas: i) leitura do arquivo e checagem de consistência; ii) importação de classes e iii) importação de propriedades.

Para a leitura do arquivo, recebe-se o caminho informado pelo usuário e utiliza-se a OWL API para carregar em memória as informações referentes à ontologia. Em seguida, para a checagem de consistência, utiliza-se o *reasoner* HermiT (GLIMM et al., 2014), que é capaz de realizar inferências tanto no nível de classes como no de propriedades. Neste processo, são identificados, além de erros de sintaxe do arquivo fornecido, restrições internas com base nos elementos que compõem a ontologia. Caso seja reconhecida alguma inconsistência, um código de erro será fornecido como retorno. Ressalta-se que os segmentos extraídos pelo SOMA também passam por este processo de checagem, visto que são importados no catálogo seguindo o mesmo processo.

Durante a importação de classes, utiliza-se o métodos da OWL API para percorrer cada uma importada individualmente. No decorrer do processo são coletados alguns metadados básicos referentes à sua identificação (e.g., nome, URI), como também é estabelecida uma lista de palavras-chave contendo uma versão normalizada do nome da classe e de um conjunto de sinônimos, extraídos do banco de dados léxico do Wordnet. Além disso, utiliza-se o *reasoner* HermiT para coletar os relacionamentos de superClasse, que são utilizados para construir as ligações entre as nós de classe no grafo.

Na importação de propriedades (de objeto e de dado), utiliza-se novamente métodos da OWL API para percorrer cada uma. Da mesma forma que ocorre com as classes, são coletados metadados básicos das propriedades e estabelecida a lista de palavras-chave contendo nome e sinônimos normalizados. Em seguida, utiliza-se o *reasoner* para descobrir os relacionamentos domain e range para outros elementos.

Processo de Segmentação de Ontologias (fonte e alvo)

Visando aproveitar os recursos disponibilizados pelo banco de dados de grafo Neo4j, a extração de segmentos ocorre por meio da geração automática de uma consulta Cypher. Essa consulta é submetida ao catálogo, que por usa vez é executada e fornece o segmento como sendo composto pelos elementos resultantes dela. Para a geração da consulta, utilizam-se os serviços de busca por palavras-chave disponibilizados no catálogo de ontologias para identificar os elementos relacionados aos requisitos. É também nesta etapa que levam-se em consideração as propriedades de conservação (vide Capítulo 4), que são utilizadas para decidir pela inclusão de elementos relacionados aos selecionados. Outro fator que afeta a quantidade de relacionamentos considerada é a definição dos requisitos operacionais.

Para a geração do segmento da ontologia fonte, recebe-se o conjunto de requisitos de dado (palavras-chave) e o valor atribuído para a extensão de segmento nos requisitos operacionais (que pode ser “SIMPLES” ou “EXPANDIDO”). Para iniciar, percorre-se cada palavra-chave definida e utiliza-se o serviço de busca do catálogo de ontologias para descobrir os conceitos que sejam relacionados. Em seguida, verifica-se o tipo do elemento relacionado (se é classe, propriedade) e constrói-se a consulta de maneira a seguir as regras de conservação para cada tipo. Nesta etapa, também define-se o nível de profundidade dos relacionamentos que será considerado na consulta, de acordo com a configuração escolhida para a extensão do segmento.

Já para a geração do segmento da ontologia alvo, é utilizado como entrada o segmento gerado para a ontologia fonte, obtido a partir da chamada ao serviço de exportação do catálogo, que fornece o segmento como um arquivo OWL. O segmento é então percorrido, utilizando a OWL API e o *reasoner* HermiT, de maneira a construir uma consulta que considera a busca por elementos da ontologia alvo que possuam palavras-chave comuns com o segmento da ontologia fonte. O segmento resultando da execução dessa consulta pode ser então armazenado no catálogo ou também exportado para um arquivo OWL, a ser utilizado em conjunto com o da ontologia fonte para a geração de alinhamentos.

APÊNDICE B – IMPLEMENTAÇÃO DO *FRAMEWORK* REDOMA

Conforme foi apresentado em detalhes no Capítulo 5, o Redoma é um *framework* para recomendação de *matchers* disponíveis em uma base de dados denominada catálogo de *matchers*. Neste apêndice, são descritos detalhes ligados aos aspectos de implementação deste catálogo e do *framework*, exemplificando o processo de execução previsto em cada um dos seus componentes.

B.1 IMPLEMENTAÇÃO DO CATÁLOGO DE MATCHERS

Da mesma forma que o catálogo de ontologias, o protótipo do catálogo de *matchers* também foi desenvolvido na linguagem de programação Java[1]. Para o armazenamento dos dados, utiliza-se o banco de dados NoSQL (POKORNY, 2011) baseado em documentos MongoDB (CHODOROW, 2013). Tal escolha se deu devido ao fato de que o catálogo suporta vários tipos de *matchers*, que utilizam parâmetros de configuração e metadados diferentes. Logo, a estrutura que armazena esses dados não poderia ser fixa. Como o MongoDB oferece o armazenamento de documentos sem esquemas fixos (*schemaless*), é possível trabalhar com atributos que podem conter tipos variados, dependendo do registro. A seguir apresentam-se o modelo de dados, a arquitetura de componentes e as APIs de serviços do catálogo de *matchers*.

Modelo de Dados

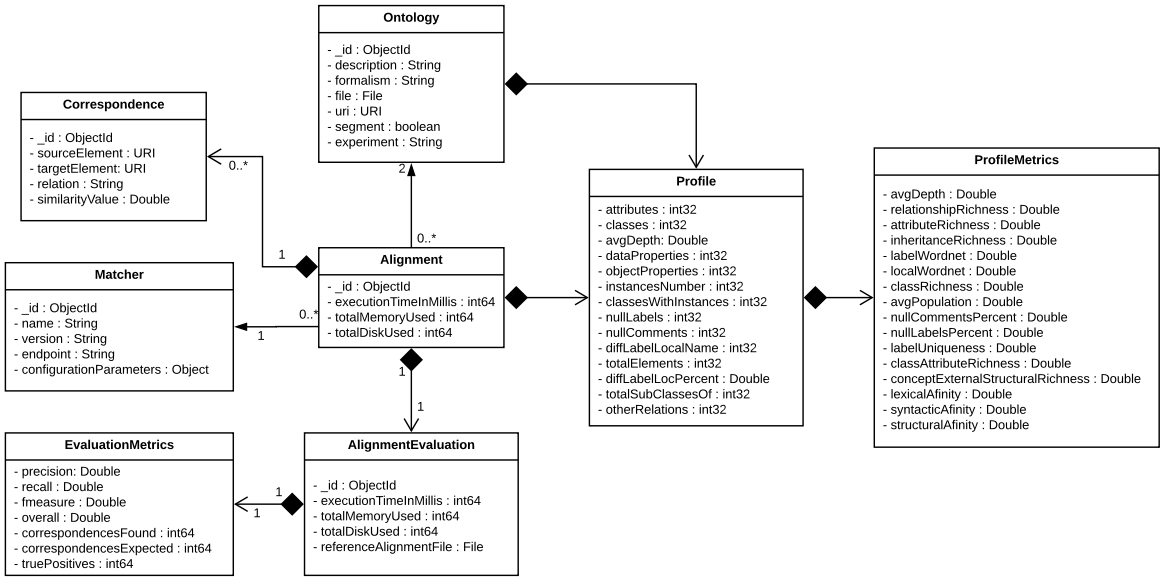


Figura 21 – Modelo de dados do catálogo de *matchers*.

[1] Código-fonte do catálogo de *matchers* disponível em <https://github.com/dass-cin/matchers-calalog>

A Figura 21 descreve o modelo de dados do catálogo de *matchers*. São consideradas as seguintes entidades:

Alinhamento (*Alignment*). Um alinhamento é composto por basicamente um conjunto de correspondências, que descreve um relacionamento entre dois elementos de ontologias.

Correspondência (*Correspondence*). Uma correspondência descreve um tipo de relacionamento (e.g., similaridade) entre dois elementos de ontologias distintas, acompanhado de um valor numérico que quantifica o nível de relacionamento (e.g., valor de similaridade).

Avaliação de Alinhamento (*Alignment Evaluation*). Após o alinhamento ser gerado, o catálogo oferece recursos para computar métricas de desempenho e de qualidade (a partir do fornecimento de um alinhamento de referência). Estas são persistidas na entidade *EvaluationMetrics*.

***Matcher*.** A entidade *Matcher* persiste, além de metadados básicos de um matcher, a referência para um endpoint REST que será considerado para a sua execução e uma lista de parâmetros de configuração. Esse endpoint é fornecido por um dos wrappers disponíveis no *framework*, conforme detalha-se na seção seguinte.

Ontologia (*Ontology*). As ontologias utilizadas nos alinhamentos também são armazenadas, no intuito de aplicar as métricas de qualidade (entidade *ProfileMetrics*) que irão estabelecer o seu perfil (entidade *Profile*). Esse perfil será útil para estabelecer a correlação entre o tipo de matcher mais adequado a alinhar determinado par de ontologias.

Perfil de Ontologia (*Profile*). O perfil de uma ontologia fornece acesso ao conjunto de atributos extraídos das ontologias e às métricas calculadas com base neles.

Métricas do Perfil de Ontologias (*ProfileMetrics*). As métricas do perfil de ontologia são calculadas com base nos atributos básicos de uma ontologia e visam medir a afinidade da ontologia com aspectos abordados pelos tipos de *matchers* existentes, tais como estruturais, sintáticos, léxicos e e instâncias.

Visão Geral da Arquitetura de Componentes

A Figura 22 descreve a arquitetura de componentes do Catálogo de *Matchers* que é organizada em três camadas: i) Serviços - que concentra a lógica de execução; ii) Repositórios - que fornece dados e abstrai o acesso à base de dados MongoDB e iii) *Wrappers* - que oferecem acesso via REST à implementação de *matchers* existentes.

A camada de serviços é composta pelos componentes:

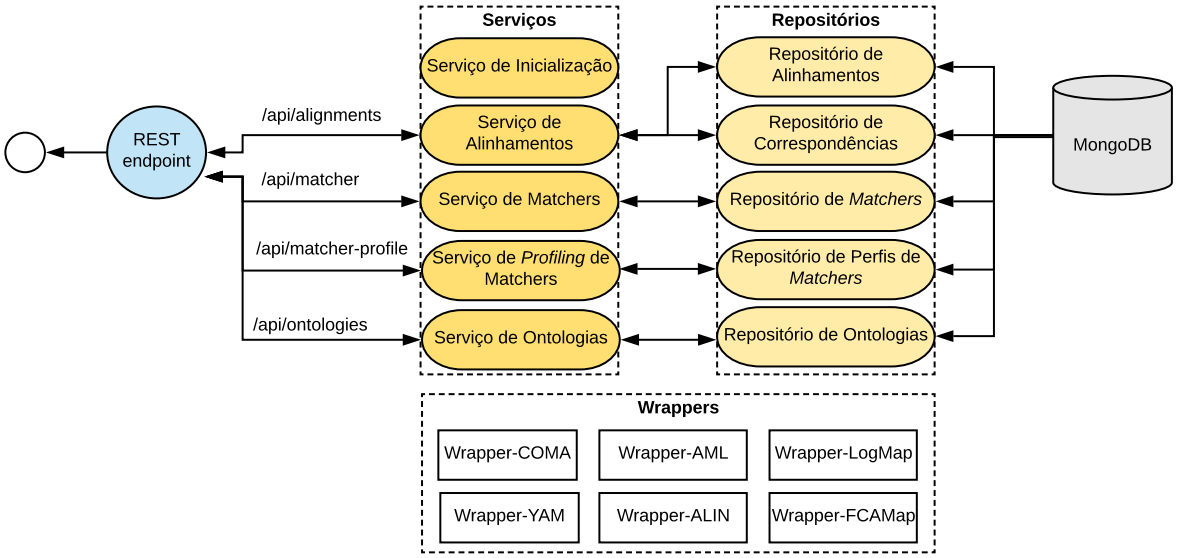


Figura 22 – Arquitetura de Componentes do Catálogo de *Matchers*.

Serviço de Inicialização. carrega um conjunto de *matchers* padrões mediante uma configuração inicial pré-definida. É executado apenas durante a inicialização da aplicação.

Serviço de Alinhamentos. oferece serviços para a geração de alinhamentos, i.e., execução de *matchers* para comparar elementos de duas ontologias, como também para a avaliação de qualidade de alinhamentos gerados.

Serviço de *Matchers*. mantém operações de gerenciamento dos *matchers* do catálogo, i.e., listagem, adição, atualização, remoção.

Serviço de Profiling de *Matchers*. estabelece o perfil dinâmico de um *matcher*. Para tal, recebe um par de ontologias e obtém o desempenho do *matcher* em alinhamentos anteriores compatíveis com as mesmas características do par fornecido.

Serviço de Ontologias. mantém operações de visualização das ontologias envolvidas em alinhamentos gerados anteriormente. Também disponibiliza-se serviços para o cálculo de métricas de qualidade de ontologias.

A camada de repositórios contém componentes referentes às principais entidades gerenciadas pelo catálogo (alinhamentos, correspondências, *matchers*, perfis de *matchers* e ontologias), oferecendo métodos para listagem/inserção/atualização/remoção de dados contidos no MongoDB. Essa camada é importante por abstrair o acesso ao banco, minimizando o impacto, caso venha a ser utilizada outra tecnologia para o armazenamento dos dados.

Por fim, a camada de adaptadores (*wrappers*), fornece um *endpoint* REST padrão para cada *matcher* registrado. Neste, recebem-se os caminhos de dois arquivos descrevendo as

ontologias que se deseja alinhar, um conjunto de parâmetros de configuração e é fornecido um conjunto de alinhamentos, estruturados no modelo padrão da Alignment API.

Cada um dos componentes pode ser acessado pelo usuário via requisições HTTP REST. Na seção a seguir detalham-se os serviços oferecidos por cada um, quais dados são requeridos como entrada e qual a saída fornecida.

B.2 API DE SERVIÇOS DO CATÁLOGO DE *MATCHERS*

A API de serviços do Catálogo de *Matchers* é dividida em três grupos principais: i) gerenciamento de *matchers* (Figura 23); ii) gerenciamento de alinhamentos (Figura 24); iii) gerenciamento de ontologias (Figura 26). A seguir, descrevem-se cada um desses grupos, incluindo os detalhes de cada serviço oferecido.

Serviços do Componente de Gerenciamento de *Matchers*

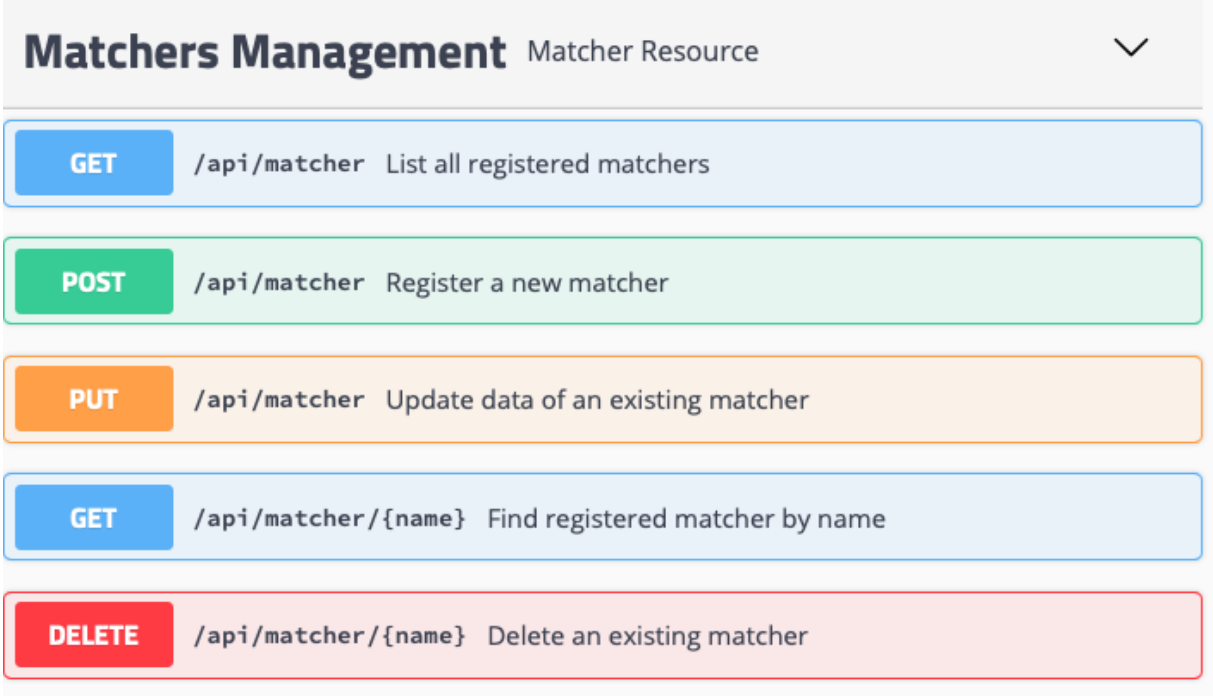


Figura 23 – Interface gráfica da API do Catálogo para o gerenciamento de *matchers*.

GET /api/matcher (*Listagem de todos os nós registrados*). Fornece a lista contendo todos os nós registrados até o momento no catálogo, incluindo os seus metadados principais;

POST /api/matcher (*Registro de um novo matcher*). Recebe como entrada os metadados do matcher (nome, descrição, versão, *endpoint*, parâmetros de configuração) e inclui ele no catálogo, passando a ser utilizado para a geração de alinhamentos e posteriormente no processo de recomendação;

PUT /api/matcher (*Atualização de dados de um matcher*). Recebe os mesmos dados considerados no registro, de maneira a sobrescrever o que já foi considerado para um *matcher* existente.

GET /api/matcher/{name} (*Recuperação de um matcher por nome*). Recebe o nome de um *matcher* e fornece o acesso aos metadados detalhados presentes no catálogo.

DELETE /api/matcher/{name} (*Remoção de um matcher existente*). Remove todos os dados envolvendo o *matcher* correspondente ao nome fornecido. Isto inclui também a remoção dos alinhamentos gerados por ele.

Serviços dos Componentes de Avaliação e de Gerenciamento de Alinhamentos

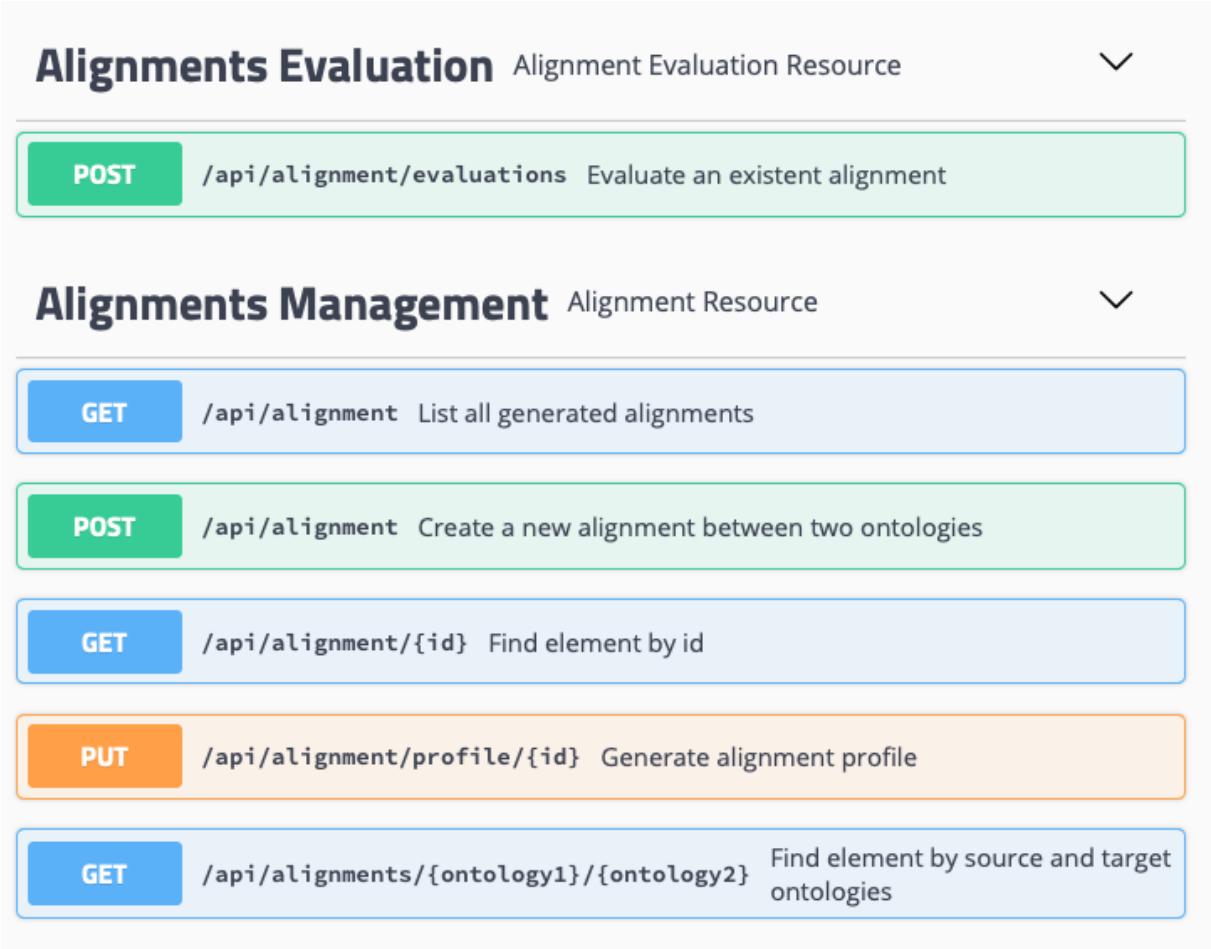


Figura 24 – Interface gráfica da API do Catálogo de *Matchers* para o gerenciamento de alinhamentos.

POST /api/alignments/evaluations (*Avaliação de Alinhamentos*). Uma vez que o alinhamento já foi gerado, é possível fornecer como entrada um alinhamento

de referência que será usado para calcular métricas de qualidade em relação às correspondências fornecidas pelo alinhamento.

GET /api/alignments (*Listagem de alinhamentos gerados*). Lista todos os alinhamentos gerados até o momento.

POST /api/alignment (*Geração de um novo alinhamento*). Dispara o processo de alinhamento de ontologias. Para tal, exige-se que seja fornecida a identificação do *matcher* registrado que será utilizado, como também os caminhos para as duas ontologias a alinhar.

GET /api/alignment/{id} (*Recuperação de alinhamentos por id*). Retorna o alinhamentos correspondente à identificação (id) fornecido como parâmetro.

PUT /api/alignment/profile/{id} (*Geração do Perfil do Alinhamento*). Cria um perfil para o alinhamento correspondente à identificação fornecida (parâmetro id), composto pela a combinação do perfil das ontologias envolvidas nele.

GET /api/alignment/{ontology1}/{ontology2} (*Alinhamentos por ontologias*). Retorna os alinhamentos gerados anteriormente entre um determinado par de ontologias fornecidos (parâmetros ontology1 e ontology2), independentemente do *matcher* utilizado.

Serviços do Componente de *Profiling* de Matchers

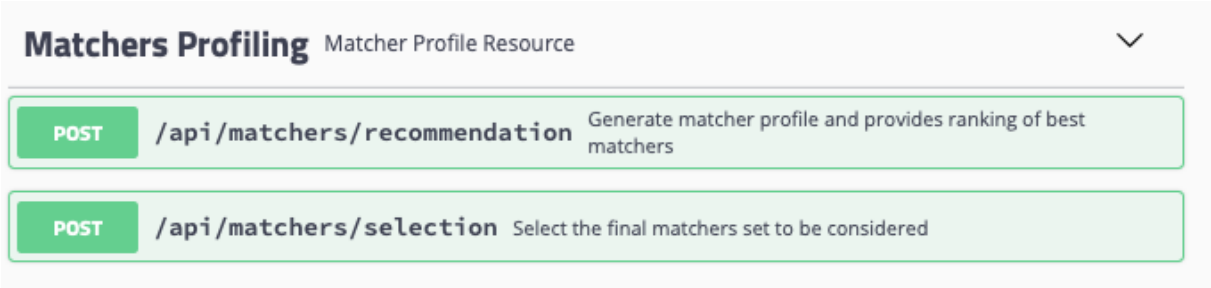


Figura 25 – Interface gráfica da API para geração do perfil e recomendação de *matchers*.

POST /api/matchers/recommendation (*Recomendação de matchers*). Recebe a referência para um par de ontologias e um conjunto de pesos para os critérios de qualidade para gerar o perfil de cada *matcher*, fornecendo como resultado a lista ordenada por medida de qualidade.

POST /api/matchers/selection (*Seleção de matchers*). Recebe a lista de *matchers* recomendada, juntamente dos requisitos de qualidade e fornece a lista final de *matchers* a ser considerada.

Serviços do Componente de Gerenciamento de Ontologias

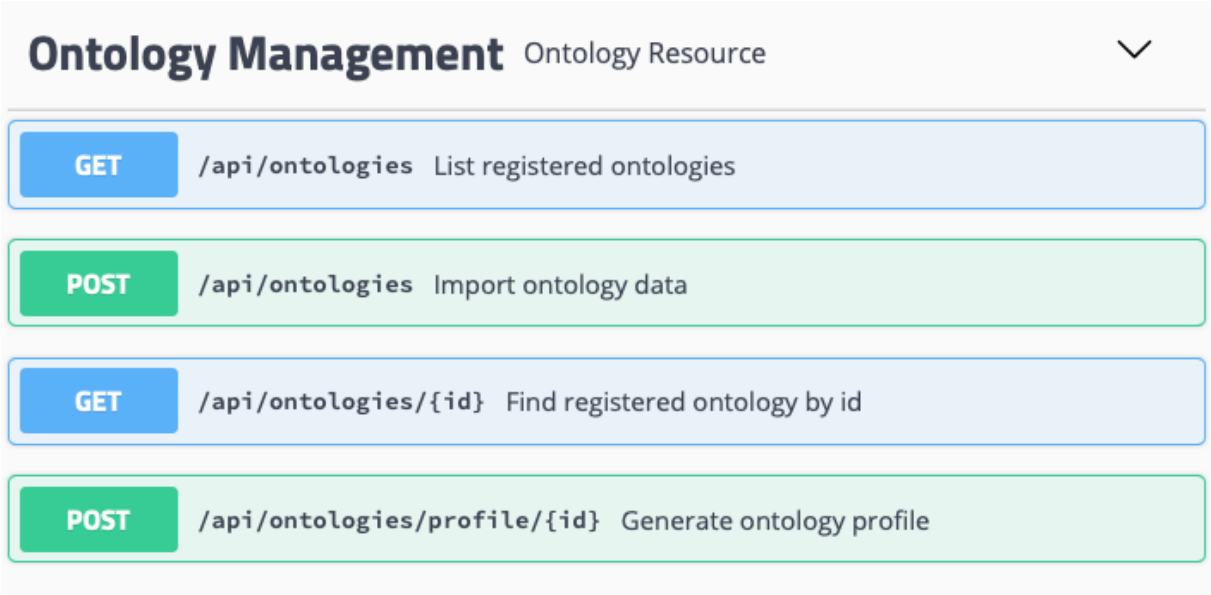


Figura 26 – Interface gráfica da API do Catálogo de *Matchers* para o gerenciamento de ontologias.

GET /api/ontologies (*Listagem de Ontologias*). Lista todas as ontologias envolvidas até o momento na geração de alinhamentos.

POST /api/ontologies (*Importação de dados de ontologias*). Importa dados de ontologias para que seja possível calcular o seu perfil. Esse serviço é utilizado primordialmente durante a geração de alinhamentos.

GET /api/ontologies/{id} (*Recuperação de ontologia por id*). Recupera os metadados existentes de uma ontologia (incluindo o seu perfil) a partir do id fornecido como parâmetro.

POST /api/ontologies/profile/{id} *Geração do perfil de ontologia.* Recebe a identificação de uma ontologia (parâmetro id) e realiza o cálculo das métricas de qualidade consideradas. O resultado é persistido junto aos metadados básicos da ontologia.

Nas seções seguintes, descrevem-se como esses serviços são utilizados por cada componente previsto no *framework* Redoma.

IMPLEMENTAÇÃO DOS COMPONENTES PREVISTOS NO FLUXO PRINCIPAL DO FRAMEWORK

A seguir são descritos os aspectos de implementação envolvendo os componentes que compõem o fluxo principal do framework Redoma: i) geração dos perfis de ontologias; ii)

geração do perfil dos *matchers*; iii) avaliação e recomendação de *matchers*; iv) seleção de *matchers* e v) execução e validação de alinhamentos.

Geração dos perfis de Ontologias

No contexto do Redoma, a primeira etapa do processo de recomendação envolve o recebimento de um par de segmentos de ontologias para a geração do seu perfil. Este é utilizado adiante como referência para os matchers que serão selecionados. Para a sua geração, primeiro é necessário estabelecer o conjunto básico de atributos da ontologia, tais como, número de classes, propriedades de dado, propriedades de objeto, profundidade média, entre outros. Para isto, utiliza-se a o *framework* Apache JENA [2], que contém métodos que auxiliam na obtenção desses atributos. Após a obtenção dos atributos básicos, foram utilizadas como referência as definições teóricas descritas nos trabalhos de Cruz et al. (2012) e Sathiya, Geetha e Sugumaran (2017) para a implementação das métricas de qualidade correspondentes ao perfil.

Geração do Perfil dos Matchers

A preparação do perfil de um *matcher* consiste em: i) receber um par de ontologias contendo suas métricas de qualidade já calculadas e ii) buscar no catálogo de *matchers* alinhamentos gerados anteriormente que envolvam ontologias que contenham as mesmas características das fornecidas como entrada. A consulta utilizada para determinar a compatibilidade de ontologias leva em consideração as medidas de afinidade descritiva e estrutural, propostas nesta tese.

Avaliação e Recomendação de Matchers

Uma vez que o perfil dos *matchers* foi estabelecido com base na tarefa de alinhamento em questão, a lista atualizada é carregada da base NoSQL para a memória. Em seguida, percorre-se cada *matcher*, e a partir da utilização de um conjunto com os critérios de qualidade e os respectivos pesos estabelecidos pelo usuário, determina-se a medida de qualidade para cada um. Como essa medida é referente, especificamente, ao alinhamento em questão, ela não é persistida no catálogo junto aos metadados do *matcher*, sendo mantida em memória para a sua utilização pelo componente de seleção.

Seleção de Matchers

A seleção de *matchers* trata-se basicamente de filtrar a lista que foi estabelecida em memória, para reduzi-la a um conjunto mínimo, no caso da recomendação de mais de um *matcher*. Para isto, utiliza-se como referência um conjunto de requisitos de qualidade

[2] Apache JENA disponível em: <https://jena.apache.org>

fornecidos pelo usuário, que irão indicar critérios (e.g., medida de qualidade mínima) para selecionar o conjunto final de *matchers* dentre os pré-selecionados.

Execução e Validação de Alinhamentos

O componente de execução recebe a identificação de um *matcher* juntamente com o par de ontologias a ser alinhado e fornece um conjunto de correspondências. Este componente pode ser utilizado para executar os *matchers* selecionados na etapa anterior ou de maneira independente. Levando em consideração o primeiro caso, o *framework* informa os identificadores dos *matchers* selecionados e invoca o serviço de geração de alinhamentos para cada um deles. Para tal, leva-se em consideração o atributo endpoint de cada *matcher*, que é acionado via requisição REST para a geração de alinhamentos. Para isto, é necessário que além do Catálogo de *Matchers*, o *wrapper* referente ao *matcher* também esteja com o serviço rodando.

Por exemplo, suponha que os *matchers* ALIN e COMA foram selecionados na etapa anterior. Ao subir o serviço de *wrapper* do ALIN, será iniciada uma instância rodando no endereço http://localhost:8082/api/alin, enquanto que o serviço do wrapper do COMA iniciará em outro endereço, e.g., http://localhost:8083/api/coma. Como esses endereços recebem o mesmo padrão de entrada (dois arquivos de ontologia e um conjunto de parâmetros de configuração) e fornecem a saída no mesmo formato, o *framework* realiza a execução de cada *matcher* e une os resultados coletados pelos dois como se fosse uma execução única.

Após a geração, o usuário pode fornecer alinhamentos de referência (parciais ou completos), para que sejam computadas as métricas de avaliação de qualidade (e.g., precisão, cobertura, medida-F).

APÊNDICE C – *MATCHERS* RECOMENDADOS PELO *FRAMEWORK* REDOMA

Neste apêndice é apresentada a lista de *matchers* recomendados pelo *Framework* Redoma, considerando os três cenários (C1, C2 e C3) definidos nos experimentos.

C.1 CENÁRIO C1

Nesta seção são apresentados os *matchers* recomendados pelo Redoma no cenário C1, que envolve o alinhamento entre as ontologias inteiras, sem passar pelo processo de segmentação. A Tabela 18 mostra os resultados quando a métrica de precisão é priorizada. A Tabela 19 apresenta os resultados em que a métrica de cobertura é priorizada e a Tabela 20 mostra os resultadoo em que a métrica prioritária é a Medida-F. Entende-se por "prioritária" a atribuição de um peso maior para uma métrica do que para as outras durante a definição dos critérios de qualidade.

Ontologia Fonte	**Ontologia Alvo**	**Precisão**	**Matcher**
confOf	ekaw	0,94	AML-WC-2
confOf	iasted	0,67	YAM
iasted	sigkdd	0,93	AML-WC-22
edas	sigkdd	0,67	COMA-RES2_SELFPATH-SIM_STR_COSINE_PPJOINPLUS
confOf	edas	0,92	AML-WC-7
cmt	ekaw	1,00	AML-SC-17
ekaw	sigkdd	1,00	COMA-RES2_SELFPATH-SIM_STR_JACCARD
cmt	sigkdd	1,00	AML-SC-70
ekaw	iasted	0,88	YAM
edas	ekaw	0,94	YAM
edas	iasted	0,80	YAM
cmt	edas	1,00	COMA-RES2_SELFNODE-SIM_STR_COSINE_PPJOINPLUS
cmt	confOf	0,88	AML-SC-12
cmt	iasted	0,80	AML-WC-24
confOf	sigkdd	1,00	AML-SC-37

Tabela 18 – *Matchers* recomendados pelo Redoma para o cenário C1 tendo como prioridade a métrica de precisão

Ontologia Fonte	Ontologia Alvo	Cobertura	Matcher
ekaw	iasted	0,70	AML-SC-56
edas	sigkdd	0,80	AML-SC-27
confOf	iasted	0,56	AML-SC-12
edas	ekaw	0,65	YAM
cmt	edas	1,00	YAM
edas	iasted	0,47	AML-WC-14
cmt	ekaw	0,64	AML-SC-1
iasted	sigkdd	0,93	LogMap
confOf	sigkdd	0,86	COMA-RES2_SELFNODE-SIM_STR_TRIGRAM_COMA
cmt	confOf	0,44	AML-SC-59
cmt	sigkdd	0,92	AML-SC-11
ekaw	sigkdd	0,82	COMA-RES2_SELFNODE-SIM_STR_TRIGRAM_COMA
confOf	ekaw	0,80	AML-SC-46
cmt	iasted	1,00	AML-SC-23
confOf	edas	0,63	AML-WC-7

Tabela 19 – *Matchers* recomendados pelo Redoma para o cenário C1 tendo como prioridade a métrica de cobertura

Ontologia Fonte	Ontologia Alvo	Medida-F	Matcher
ekaw	iasted	0,78	YAM
ekaw	sigkdd	0,86	YAM
edas	ekaw	0,77	YAM
iasted	sigkdd	0,90	AML-WC-2
cmt	edas	0,90	YAM
confOf	ekaw	0,84	AML-SC-46
cmt	confOf	0,58	AML-SC-67
edas	iasted	0,56	AML-WC-16
edas	sigkdd	0,73	COMA-RES2_SELFPATH-SIM_STR_JACCARD
confOf	sigkdd	0,86	YAM
cmt	ekaw	0,71	AML-SC-18
confOf	iasted	0,56	AML-WC-28
cmt	iasted	0,89	AML-WC-16
cmt	sigkdd	0,96	AML-SC-48
confOf	edas	0,75	AML-WC-7

Tabela 20 – *Matchers* recomendados pelo Redoma para o cenário C1 tendo como prioridade a Medida-F

C.2 CENÁRIO C2

Nesta seção são apresentados os *matchers* recomendados pelo Redoma no cenário C2, que envolve o alinhamento entre segmentos de ontologias gerados a partir dos requisitos descritos na Seção 4.4.2. A Tabela 18 mostra os resultados quando a métrica de precisão é priorizada. A Tabela 19 apresenta os resultados em que a métrica de cobertura é priorizada e a Tabela 20 mostra os resultadoo em que a métrica prioritária é a Medida-F. Entende-se por "prioritária" a atribuição de um peso maior para uma métrica do que para as outras durante a definição dos critérios de qualidade.

Ontologia Fonte	Ontologia Alvo	Precisão	Matcher
source-segment-iasted-sigkdd	target-segment-iasted-sigkdd	0,00	COMA-RES2_SIBLINGS-SIM_STR_TRIGRAM_LOWMEM
source-segment-cmt-sigkdd	target-segment-cmt-sigkdd	1,00	AML-WC-22
source-segment-cmt-confOf	target-segment-cmt-confOf	1,00	COMA-RES2_SELFPATH-SIM_STR_TRIGRAM_LOWMEM
source-segment-cmt-ekaw	target-segment-cmt-ekaw	1,00	AML-SC-68
source-segment-cmt-iasted	target-segment-cmt-iasted	1,00	COMA-RES2_PARENTS-SIM_STR_EQUAL
source-segment-ekaw-sigkdd	target-segment-ekaw-sigkdd	1,00	AML-WC-4
source-segment-edas-iasted	target-segment-edas-iasted	0,67	COMA-RES2_SELFNODE-SIM_STR_TRIGRAM_OPT
source-segment-conference-sigkdd	target-segment-conference-sigkdd	1,00	COMA-RES2_SELFNODE-SIM_STR_JAROWINKLER
source-segment-ekaw-iasted	target-segment-ekaw-iasted	1,00	AML-SC-21
source-segment-cmt-edas	target-segment-cmt-edas	1,00	AML-SC-70
source-segment-confOf-ekaw	target-segment-confOf-ekaw	0,00	AML-WC-16
source-segment-conference-confOf	target-segment-conference-confOf	1,00	AML-WC-24
source-segment-conference-edas	target-segment-conference-edas	1,00	AML-WC-16
source-segment-cmt-conference	target-segment-cmt-conference	1,00	COMA-RES2_SELFPATH-SIM_STR_COSINE PPJOIN-PLUSFULLYCACHED
source-segment-conference-iasted	target-segment-conference-iasted	1,00	AML-WC-7
source-segment-conference-ekaw	target-segment-conference-ekaw	1,00	COMA-RES2_PARENTS-SIM_STR_COSINE PPJOIN-PLUS
source-segment-confOf-edas	target-segment-confOf-edas	1,00	COMA-RES2_PARENTS-SIM_STR_EQUAL
source-segment-edas-sigkdd	target-segment-edas-sigkdd	1,00	COMA-RES2_PARENTS-SIM_STR_EDITDIST
source-segment-confOf-sigkdd	target-segment-confOf-sigkdd	1,00	COMA-RES2_SELFPATH-SIM_STR_JACCARD
source-segment-edas-ekaw	target-segment-edas-ekaw	1,00	AML-SC-35

Tabela 21 – *Matchers* recomendados pelo Redoma para o cenário C2 tendo como prioridade a métrica de precisão

Ontologia Fonte	**Ontologia Alvo**	**Cobertura**	**Matcher**
source-segment-cmt-ekaw	target-segment-cmt-ekaw	0,43	AML-SC-69
source-segment-cmt-confOf	target-segment-cmt-confOf	0,27	COMA-RES2_SELFPATH-SIM_STR_TRIGRAM_LOWMEM
source-segment-ekaw-iasted	target-segment-ekaw-iasted	0,50	AML-SC-26
source-segment-edas-sigkdd	target-segment-edas-sigkdd	0,50	AML-SC-42
source-segment-confOf-ekaw	target-segment-confOf-ekaw	0,00	COMA-RES2_SELFPATH-SIM_STR_TRIGRAM_OPT
source-segment-ekaw-sigkdd	target-segment-ekaw-sigkdd	0,57	AML-SC-37
source-segment-conference-sigkdd	target-segment-conference-sigkdd	0,67	COMA-RES2_SELFNODE-SIM_STR LEVENSH-TEIN_LUCENE
source-segment-cmt-sigkdd	target-segment-cmt-sigkdd	0,57	COMA-RES2_SELFPATH-SIM_STR_TRIGRAM_COMA
source-segment-edas-iasted	target-segment-edas-iasted	0,40	AML-SC-38
source-segment-confOf-sigkdd	target-segment-confOf-sigkdd	0,25	COMA-RES2_CHILDREN-SIM_STR_TRIGRAM_LOWMEM
source-segment-conference-confOf	target-segment-conference-confOf	0,75	AML-WC-6
source-segment-conference-ekaw	target-segment-conference-ekaw	0,38	COMA-RES2_SELFPATH-SIM_STR_TRIGRAM_LOWMEM
source-segment-edas-ekaw	target-segment-edas-ekaw	0,30	AML-SC-51
source-segment-cmt-iasted	target-segment-cmt-iasted	0,75	AML-SC-20
source-segment-cmt-edas	target-segment-cmt-edas	0,63	AML-SC-66
source-segment-conference-edas	target-segment-conference-edas	0,63	COMA-RES2_SELFNODE-SIM_STR LEVENSH-TEIN_LUCENE
source-segment-confOf-edas	target-segment-confOf-edas	0,33	AML-SC-10
source-segment-iasted-sigkdd	target-segment-iasted-sigkdd	0,30	COMA-RES2_SELFNODE-SIM_STR_TRIGRAM_IFUICE
source-segment-conference-iasted	target-segment-conference-iasted	0,40	AML-WC-19
source-segment-cmt-conference	target-segment-cmt-conference	0,20	AML-WC-24

Tabela 22 – *Matchers* recomendados pelo Redoma para o cenário C2 tendo como prioridade a métrica de cobertura

Ontologia Fonte	**Ontologia Alvo**	**Medida-F**	**Matcher**
source-segment-conference-confOf	target-segment-conference-confOf	0,86	AML-SC-54
source-segment-conference-ekaw	target-segment-conference-ekaw	0,50	COMA
source-segment-conference-sigkdd	target-segment-conference-sigkdd	0,80	COMA
source-segment-cmt-confOf	target-segment-cmt-confOf	0,43	COMA-RES2_SELFPATH-SIM_STR_TRIGRAM_LOWMEM
source-segment-cmt-sigkdd	target-segment-cmt-sigkdd	0,73	AML-SC-21
source-segment-cmt-iasted	target-segment-cmt-iasted	0,75	AML-SC-1
source-segment-edas-ekaw	target-segment-edas-ekaw	0,46	COMA-RES2_SELFNODE-SIM_STR_EQUAL
source-segment-ekaw-iasted	target-segment-ekaw-iasted	0,67	AML-WC-7
source-segment-confOf-sigkdd	target-segment-confOf-sigkdd	0,40	COMA-RES2_SELFPATH-SIM_STR_TRIGRAM_OPT
source-segment-conference-iasted	target-segment-conference-iasted	0,57	AML-SC-54
source-segment-ekaw-sigkdd	target-segment-ekaw-sigkdd	0,73	AML-SC-69
source-segment-cmt-edas	target-segment-cmt-edas	0,77	COMA-RES2_SELFNODE-SIM_STR_COSINE_PPJOINPLUS
source-segment-iasted-sigkdd	target-segment-iasted-sigkdd	0,00	COMA-RES2_SELFNODE-SIM_STR_TRIGRAM_COMA
source-segment-edas-iasted	target-segment-edas-iasted	0,50	AML-SC-1
source-segment-cmt-ekaw	target-segment-cmt-ekaw	0,60	AML-SC-59
source-segment-cmt-conference	target-segment-cmt-conference	0,33	COMA-RES2_SELFPATH-SIM_STR_JACCARD
source-segment-conference-edas	target-segment-conference-edas	0,77	COMA-RES2_SELFNODE-SIM_STR_TRIGRAM_COMA
source-segment-edas-sigkdd	target-segment-edas-sigkdd	0,67	AML-SC-54
source-segment-confOf-edas	target-segment-confOf-edas	0,50	AML-SC-1
source-segment-confOf-ekaw	target-segment-confOf-ekaw	0,00	AML-SC-60

Tabela 23 – *Matchers* recomendados pelo Redoma para o cenário C2 tendo como prioridade a Medida-F

C.3 CENÁRIO C3

Nesta seção são apresentados os *matchers* recomendados pelo Redoma no cenário C3, que envolve o alinhamento entre segmentos de ontologias gerados a partir dos requisitos descritos na Seção 4.4.2. A Tabela 18 mostra os resultados quando a métrica de precisão é priorizada. A Tabela 19 apresenta os resultados em que a métrica de cobertura é priorizada e a Tabela 20 mostra os resultadoo em que a métrica prioritária é a Medida-F. Entende-se por "prioritária" a atribuição de um peso maior para uma métrica do que para as outras durante a definição dos critérios de qualidade.

Ontologia Fonte	**Ontologia Alvo**	**Precisão**	**Matcher**
source-segment-cmt-sigkdd	target-segment-cmt-sigkdd	1,00	COMA-RES2_SELFNODE-SIM_STR_JAROWINKLER
source-segment-iasted-sigkdd	target-segment-iasted-sigkdd	1,00	AML-SC-39
source-segment-conference-iasted	target-segment-conference-iasted	1,00	AML-SC-59
source-segment-cmt-confOf	target-segment-cmt-confOf	1,00	COMA-RES2_SIBLINGS-SIM_STR_TRIGRAM_OPT
source-segment-confOf-ekaw	target-segment-confOf-ekaw	0,67	COMA-RES2_SELFNODE-SIM_STR_COSINE PPJOIN-PLUSFULLYCACHED
source-segment-ekaw-iasted	target-segment-ekaw-iasted	1,00	AML-SC-3
source-segment-edas-sigkdd	target-segment-edas-sigkdd	1,00	AML-SC-12
source-segment-cmt-iasted	target-segment-cmt-iasted	1,00	COMA-RES2_PARENTS-SIM_STR_TRIGRAM_IFUICE
source-segment-ekaw-sigkdd	target-segment-ekaw-sigkdd	1,00	AML-SC-32
source-segment-confOf-sigkdd	target-segment-confOf-sigkdd	1,00	AML-WC-16
source-segment-edas-iasted	target-segment-edas-iasted	0,67	AML-WC-31
source-segment-conference-confOf	target-segment-conference-confOf	1,00	AML-SC-18
source-segment-conference-ekaw	target-segment-conference-ekaw	1,00	AML-SC-2
source-segment-cmt-conference	target-segment-cmt-conference	1,00	COMA-RES2_PARENTS-SIM_STR_TRIGRAM_COMA
source-segment-conference-edas	target-segment-conference-edas	1,00	AML-SC-32
source-segment-edas-ekaw	target-segment-edas-ekaw	1,00	COMA-RES2_SELFPATH-SIM_STR_ LEVENSH-TEIN_LUCENE
source-segment-confOf-iasted	target-segment-confOf-iasted	1,00	AML-SC-43
source-segment-conference-sigkdd	target-segment-conference-sigkdd	1,00	COMA-RES2_PARENTS-SIM_STR_COSINE_PPJOINPLUS
source-segment-cmt-edas	target-segment-cmt-edas	1,00	AML-SC-48
source-segment-confOf-edas	target-segment-confOf-edas	1,00	AML-SC-13
source-segment-cmt-ekaw	target-segment-cmt-ekaw	1,00	AML-SC-24

Tabela 24 – *Matchers* recomendados pelo Redoma para o cenário C3 tendo como prioridade a métrica de precisão

Ontologia Fonte	**Ontologia Alvo**	**Cobertura**	**Matcher**
source-segment-cmt-edas	target-segment-cmt-edas	0,63	LogMap
source-segment-confOf-edas	target-segment-confOf-edas	0,50	LogMap
source-segment-confOf-sigkdd	target-segment-confOf-sigkdd	1,00	LogMap
source-segment-conference-ekaw	target-segment-conference-ekaw	0,20	AML-WC-21
source-segment-cmt-ekaw	target-segment-cmt-ekaw	0,56	LogMap
source-segment-confOf-iasted	target-segment-confOf-iasted	0,67	AML-SC-51
source-segment-ekaw-iasted	target-segment-ekaw-iasted	0,75	LogMap
source-segment-conference-confOf	target-segment-conference-confOf	0,25	LogMap
source-segment-iasted-sigkdd	target-segment-iasted-sigkdd	1,00	LogMap
source-segment-edas-ekaw	target-segment-edas-ekaw	0,83	LogMap
source-segment-cmt-confOf	target-segment-cmt-confOf	0,63	LogMap
source-segment-confOf-ekaw	target-segment-confOf-ekaw	0,83	LogMap
source-segment-cmt-sigkdd	target-segment-cmt-sigkdd	1,00	LogMap
source-segment-cmt-iasted	target-segment-cmt-iasted	1,00	LogMap
source-segment-cmt-conference	target-segment-cmt-conference	0,20	COMA-RES2_SELFNODE-SIM_STR_COSINE_ PPJOIN-PLUS
source-segment-conference-edas	target-segment-conference-edas	0,33	LogMap
source-segment-conference-sigkdd	target-segment-conference-sigkdd	0,20	COMA-RES2_SELFNODE-SIM_STR_EQUAL
source-segment-edas-iasted	target-segment-edas-iasted	1,00	COMA-RES2_SELFNODE-SIM_STR_TRIGRAM_COMA
source-segment-edas-sigkdd	target-segment-edas-sigkdd	0,75	LogMap
source-segment-conference-iasted	target-segment-conference-iasted	0,67	AML-SC-39
source-segment-ekaw-sigkdd	target-segment-ekaw-sigkdd	0,80	LogMap

Tabela 25 – *Matchers* recomendados pelo Redoma para o cenário C3 tendo como prioridade a métrica de cobertura

Ontologia Fonte	**Ontologia Alvo**	**Medida-F**	**Matcher**
source-segment-cmt-edas	target-segment-cmt-edas	0,40	AML-SC-57
source-segment-conference-edas	target-segment-conference-edas	0,36	AML-SC-14
source-segment-conference-confOf	target-segment-conference-confOf	0,22	AML-SC-31
source-segment-confOf-iasted	target-segment-confOf-iasted	0,80	AML-WC-15
source-segment-confOf-ekaw	target-segment-confOf-ekaw	0,44	YAM
source-segment-conference-ekaw	target-segment-conference-ekaw	0,33	AML-SC-15
source-segment-cmt-sigkdd	target-segment-cmt-sigkdd	0,57	AML-SC-11
source-segment-conference-sigkdd	target-segment-conference-sigkdd	0,33	AML-SC-60
source-segment-confOf-edas	target-segment-confOf-edas	0,55	AML-WC-3
source-segment-cmt-conference	target-segment-cmt-conference	0,33	COMA-RES2_PARENTS-SIM_STR_TRIGRAM_LOWMEM
source-segment-iasted-sigkdd	target-segment-iasted-sigkdd	0,62	AML-SC-16
source-segment-edas-sigkdd	target-segment-edas-sigkdd	0,67	AML-SC-47
source-segment-cmt-ekaw	target-segment-cmt-ekaw	0,20	AML-SC-9
source-segment-confOf-sigkdd	target-segment-confOf-sigkdd	0,80	AML-SC-66
source-segment-edas-iasted	target-segment-edas-iasted	0,80	AML-SC-28
source-segment-ekaw-sigkdd	target-segment-ekaw-sigkdd	0,57	COMA-RES2_SELFNODE-SIM_STR_COSINE_PPJOINPLUS
source-segment-cmt-confOf	target-segment-cmt-confOf	0,40	AML-WC-28
source-segment-conference-iasted	target-segment-conference-iasted	0,80	COMA-RES2_SELFNODE-SIM_STR_JAROWINKLER
source-segment-cmt-iasted	target-segment-cmt-iasted	0,80	AML-SC-56
source-segment-ekaw-iasted	target-segment-ekaw-iasted	0,67	AML-WC-22
source-segment-edas-ekaw	target-segment-edas-ekaw	0,50	COMA-RES2_SELFPATH-SIM_STR_LEVENSHTEIN_LUCENE

Tabela 26 – *Matchers* recomendados pelo Redoma para o cenário C3 tendo como prioridade a Medida-F