PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EXTRAÇÃO E ALINHAMENTO DE HIERARQUIAS EM PÁGINAS WEB

BERNARDO SEVERO DE SOUZA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientadora: Prof.ª Renata Vieira

Porto Alegre

2012

S729e Souza, Bernardo Severo de
Extração e alinhamento de hierarquias em páginas WEB / Bernardo Severo de Souza. – Porto Alegre, 2012.
90 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.

Orientador: Profª. Drª. Renata Vieira.

1. Informática. 2. Ontologia. 3. Semântica. 4. World Wide Web. I. Vieira, Renata. II. Título.

CDD 006.35

Ficha Catalográfica elaborada pelo

Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "***Extração e Alinhamento de Hierarquias em Páginas WEB***", apresentada por Bernardo Severo de Souza como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Inteligência Computacional, aprovada em 23/03/2012 pela Comissão Examinadora:

Profa. Dra. Renata Vieira - PPGCC/PUCRS
Orientadora

Profa. Dra. Vera Lúcia Strube de Lima - PPGCC/PUCRS

Profa. Dra. Carina Friedrich Dorneles - UFSC

Homologada em 22/06/2012, conforme Ata No. 013 pela Comissão Coordenadora.

Prof. Dr. Paulo Henrique Lemelle Fernandes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320–3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

AGRADECIMENTOS

A minha família e amigos, pelo incentivo, paciência e compreensão.

A minha namorada Gabriela, pelo apoio.

A minha orientadora Renata, pelos conselhos, ensinamentos e atenção.

Aos membros do PPGCC, professores e colegas, por proporcionar um aprendizado de excelência na área acadêmica.

Aos colegas do Grupo de PLN, pelas experiências trocadas, dicas e apoio no andamento do mestrado.

EXTRAÇÃO E ALINHAMENTO DE HIERARQUIAS EM PÁGINAS WEB

RESUMO

A heterogeneidade das formas como as informações são apresentadas na web é uma característica que dificulta a análise de dados entre diferentes fontes. Mesmo em estruturas hierárquicas, que possuem uma relação mínima de ordem, não há um padrão para como exibir os elementos e como referenciar os mesmos. Por isso, o foco principal deste trabalho é apresentar uma ferramenta visual e extensível que centraliza e auxilia operações sobre tais estruturas em páginas da internet.

Para tanto, foi elaborada a ferramenta PLATAL (Plataforma de Extração e Alinhamento de Hierarquias), voltada para facilitar as diversas operações de alinhamento de hierarquias. A ferramenta possui quatro módulos principais: um para extração de hierarquias de páginas da internet, tornando-as disponíveis para manipulação em formatos padrões da web semântica; um para alinhamento automatizado dessas hierarquias, baseado em diversas heurísticas e técnicas de alinhamento de ontologias; um para alinhamento manual de hierarquias, possibilitando a criação de alinhamentos de referência; e por fim, um para avaliação de alinhamentos, através da análise de precisão e abrangência.

Para avaliar as heurísticas de alinhamento, foram realizados experimentos no domínio de comércio eletrônico. Os resultados foram comparados com o produzido por outras ferramentas descritas na literatura. Portanto, este trabalho contribui como uma forma de viabilizar a criação de hierarquias alinhadas a partir das estruturas heterogêneas encontradas na web.

Palavras chave: hierarquia, ontologia, alinhamento, alinhamento de ontologias

HIERARCHIES EXTRACTION AND ALIGNMENT IN WEB PAGES

ABSTRACT

The heterogeneity of the ways information is presented on the web is a characteristic which complicates the analysis between different sources. Even in hierarchical structures, which have a minimum relation of order, there is no standard for how to display the elements and how to reference them. Therefore, this work's main focus is to present a visual and extensible tool that centralizes and supports operations on such structures in web pages.

To that end, the PLATAL (Platform of Hierarchy Extraction and Alignment) tool was developed, to facilitate the various operations of hierarchy alignment. The tool has four main modules: one for extracting hierarchies of web pages, making them available for manipulation in standard formats of the semantic web; one for automated alignment of these hierarchies, based on various heuristics and ontology alignment techniques; one for manual alignment of hierarchies, allowing the creation of reference alignments; and finally, one for evaluation of alignments, through the analysis of precision and recall.

To evaluate the heuristics of alignment, experiments were performed in the field of e-commerce. The results were compared with that produced by other tools described in the literature. Therefore, this work contributes as a way to enable the creation of aligned hierarchies from heterogeneous structures found on the web.

Keywords: hierarchy, ontology, alignment, ontology matching

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Neste capítulo, será apresentado o contexto do desenvolvimento desta dissertação, sua motivação, objetivo, plano de trabalho, e por fim a estrutura do texto.

1.1. CONTEXTO E MOTIVAÇÃO

O processo de análise e monitoramento das condições do mercado em que determinada empresa está inserida é chamado de Inteligência Competitiva (IC). Empreendimentos de todos os tamanhos passaram a buscar IC para conseguir vencer os desafios encontrados no mercado, frente a novos concorrentes que constantemente surgem, o desenvolvimento de novos produtos e serviços, pesquisas por novas tecnologias, etc.

Ao longo dos anos, tem ocorrido um crescimento do número de operações envolvendo o comércio e, portanto, da receita gerada por elas, onde apenas no Brasil tivemos um crescimento no volume de vendas de 7,3% em relação ao ano anterior [27]. Houve, como consequência, um crescimento também do volume de informações geradas sobre essas operações, e um aumento da complexidade da avaliação de tamanho conteúdo, a fim de extrair a uma IC útil a tempo para sua aplicação.

Muitas vezes dados importantes e relacionados encontram-se dispersos, sem uma apresentação padrão ou organização semântica, dificultando o acesso, comparação e obtenção de conhecimento sobre os mesmos. Na busca de auxílio para a descoberta de informações relevantes em tal contexto, encontramos apoio tecnológico em tópicos muito discutidos atualmente na área de Processamento de Linguagem Natural (PLN), devido à forma como tais informações se encontram (não estruturadas e em linguagem natural).

Esses dados se encontram semi-estruturados, isto é, possuem uma ordem, uma hierarquia, que tipicamente é desigual nas diferentes fontes existentes, apesar de possuírem muitas vezes o mesmo significado. O que se deseja é então encontrar uma forma de identificar os elementos comuns representados em diferentes hierarquias, facilitando assim o acesso, comparação e análise da informação de forma mais simplificada.

Este trabalho propõe uma abordagem ao problema composta por duas etapas principais. A primeira delas é a construção de um *framework*, isto é, uma ferramenta que facilite as atividades necessárias para a construção do alinhamento que se deseja. O mesmo agregará funcionalidades associadas a alinhamentos que não se encontram disponíveis, ou então são muito limitadas em outros sistemas.

Para a segunda etapa, será utilizado o sistema construído na primeira etapa, aplicado a um cenário prático. Primeiramente será realizada a extração de hierarquias de fontes web. Como cada fonte apresenta as informações, mesmo que semi-estruturadas, com diferentes codificações, procurou-se uma forma de auxiliar essa extração, e padronizar sua apresentação.

Em seguida, é proposto um alinhamento automático das hierarquias obtidas da extração. Esse segundo momento é similar a tarefa de alinhamento de ontologias, porém com os elementos das hierarquias não possuem necessariamente uma relação “é um” para com seu elemento pai, e os elementos representados podem possuir o mesmo nome.

Por fim, o trabalho é encerrado com a elaboração de alinhamentos de referência e a avaliação dos alinhamentos sugeridos pela ferramenta quando comparados aos de referência. Também foram utilizados alinhamento de outros sistemas para comparação, onde de uma forma geral se encontrou um resultado de mesmo nível que os demais, e em alguns casos desempenho superior a todos os outros.

Este trabalho visa, portanto, aplicar técnicas elaboradas no meio acadêmico em problemas práticos atuais do mercado. Com o alinhamento entre hierarquias proposto, pretende-se prover um suporte para elaborar uma visualização unificada e simplificada de informações distribuídas em diferentes fontes. E com isso, auxiliar na análise de informações mercadológicas relevantes em fontes web, contribuindo para tarefas típicas de IC.

1.1.1. ATUALIDADE DE PÁGINAS DE E-COMMERCE

É possível observar para o contexto que será abordado na Subseção 2.6, a atualidade do *e-commerce* em nível de estrutura. Ao acessar páginas da internet com *e-catalogs*, percebemos uma grande heterogeneidade, seja de apresentação visual das categorias (posição, cores, letras maiúsculas ou minúsculas, etc.), como das

classificações existentes para as mesmas. Vemos algumas dessas diferenças na Figura 1.

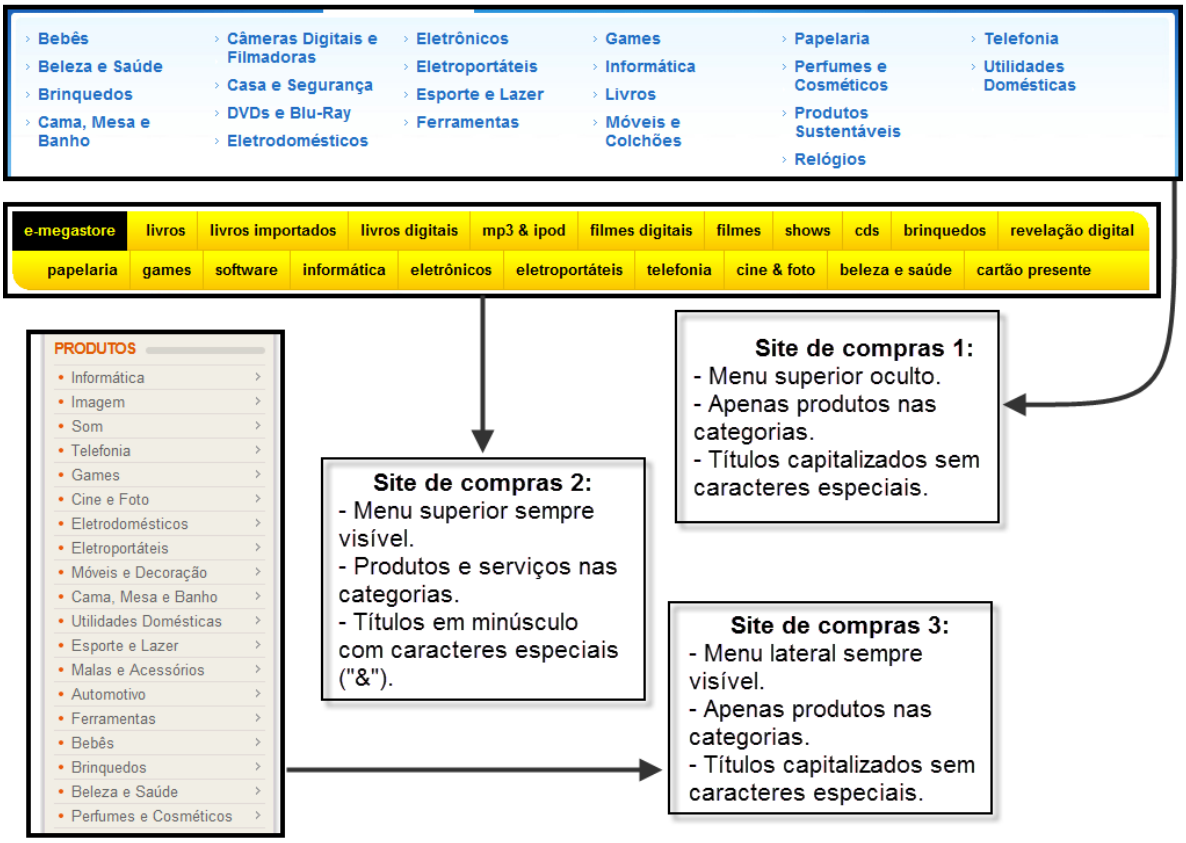

Figura 1 - Exemplos de diferentes *e-catalogs*

Na Figura 2 é possível ver que um determinado site de *e-commerce* do Brasil possui inconsistências, como a categoria “Informática e Acessórios” estar no mesmo nível que “Informática”. Um usuário que acessa essa página deve questionar se os produtos da primeira categoria se encontram na segunda, e vice-versa.

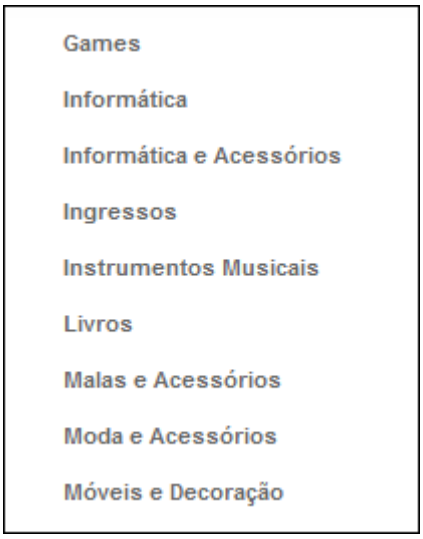

Figura 2 - Exemplos de *e-catalog* com inconsistências hierárquicas

Muitos misturam serviços, produtos, alguns apresentam seus menus em uma posição superior, outros na lateral. Não existe um consenso quanto a como e quais informações exibir.

Pensando em uma provável padronização das hierarquias dos *sites* para publicação de produtos em serviços de comparação de preços e serviços de vendas de produtos usados, foi também investigada a estrutura desses *websites*. Encontrou-se que esses, por sua vez, acabam apenas definindo suas próprias estruturas, e ao submeter um produto, o vendedor deve informar para qual categoria do *website* ele é destinado.

Na Figura 3 ainda podemos ver o principal problema levantado por esse estudo, as diferenças de nomenclatura através das diferentes fontes, das categorias presentes em seus catálogos. Nela em específico podemos ver as variações entre apenas três *websites* em relação à busca pela categoria de câmeras. Elas possuem variação nos termos utilizados, em caracteres que as compõem, por exemplo: “&” ao invés de “e”; “-“ ao invés de espaço em branco; maiúsculas e minúsculas alternadas; etc.

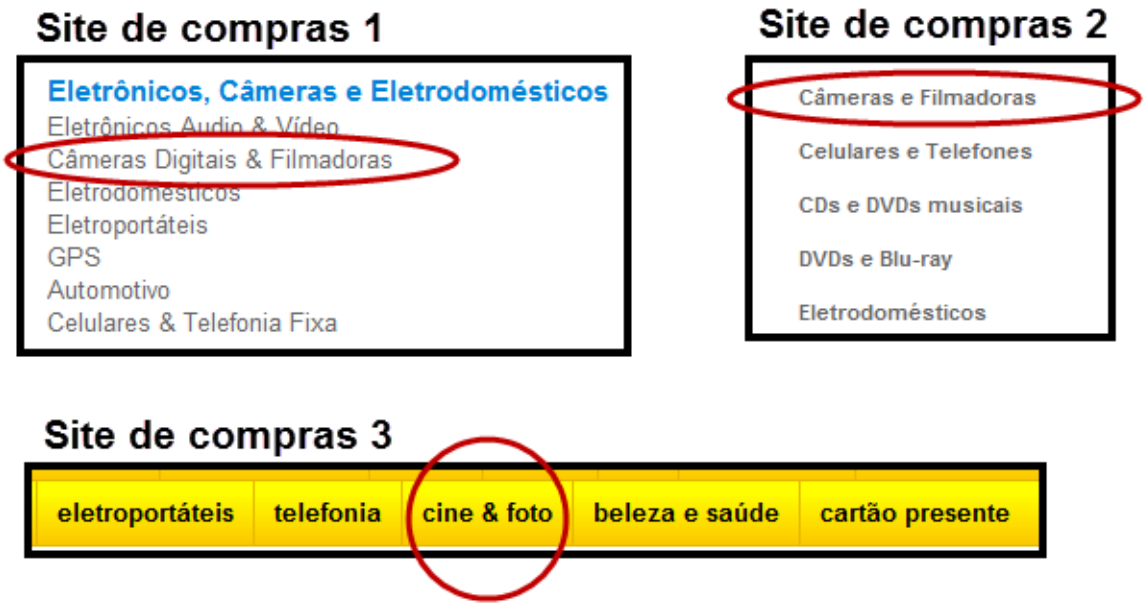

Figura 3 - Exemplos de diferentes nomenclaturas de categorias

1.2. OBJETIVO E PLANO DE TRABALHO

Propor e avaliar uma forma de facilitar a extração e alinhamento entre diferentes hierarquias, típicas na atual arquitetura de informação em páginas web, é o objetivo desse estudo. A solução será então capaz de analisar e processar tais informações de forma dinâmica.

Um dos módulos da aplicação visa extrair de páginas da internet a hierarquia por elas apresentada. É possível obter a hierarquia visualizada em qualquer que seja a estrutura de seu conteúdo (em linguagem HTML), mesmo que distribuída em mais de uma página. Como saída, a aplicação oferece a possibilidade de exportar tal hierarquia para formatos padronizados e amplamente difundidos pela internet para armazenar tais estruturas (OWL e SKOS).

Em outro módulo, essas hierarquias extraídas e padronizadas são analisadas, buscando um alinhamento entre suas entidades. Esse alinhamento produzido pode então ser exportado em um formato XML padronizado [17] e utilizado por diversas aplicações da área.

Em outros módulos auxiliares para as principais atividades, é possível criar um alinhamento igual ao gerado automaticamente, porém de forma manual; e extrair métricas

de avaliação de um alinhamento através da comparação do mesmo com um de referência.

Para abordar as soluções desenvolvidas nas etapas do trabalho e suas funções auxiliares, foi criada a ferramenta intitulada “**Plat**aforma de Extração e **Al**inhamento de Hierarquias” (PLATAL), a qual pode-se ver a interface inicial na Figura 4. Através da mesma e seus módulos, é possível atingir o objetivo desse trabalho.

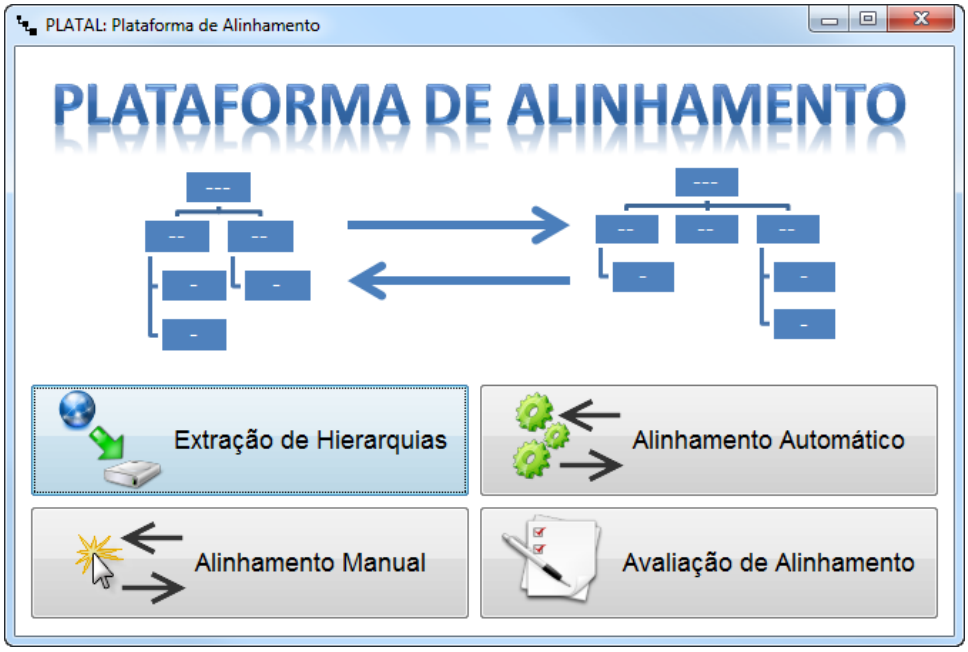

Figura 4 - Interface da aplicação PLATAL

Para os experimentos a serem realizados ao longo do estudo, foram utilizados sites de comércio eletrônico (*e-commerce*), como base para extração de hierarquias e análise de alinhamentos. O *e-commerce* consiste basicamente na compra e venda de produtos e serviços através de meios eletrônicos, tipicamente através da internet. Esta área está em constante expansão, tanto em países em desenvolvimento como nos de primeiro mundo, o que fortalece a importância que a extração efetiva de informações representa para essa área.

Pretende-se aplicar alguns dos conhecimentos obtidos dos trabalhos observados, porém em uma estrutura diferente (baseada no que o usuário encontra em um *website* de vendas usual), portanto flexível e customizável, que considera a adaptação aos formatos que vierem a surgir ao longo do tempo. Também terá como diferencial uma das maiores dificuldades nessa área, a aplicação da proposta em domínios em português (ainda que a solução seja modularizada a um ponto de permitir a portabilidade para outras línguas).

Acredita-se que com o estudo aqui desenvolvido, foi possível tratar significantemente um problema constante do *e-commerce*: a diversidade de formas de apresentação das mesmas categorias de produtos oferecidos ao longo dos diferentes *websites* de compras, em seus chamados catálogos eletrônicos (*e-catalogs*).

Avaliações dos resultados alcançados foram realizadas para comparar a precisão e abrangência obtidas frente a métodos mais básicos. Foram capturadas hierarquias de sites de *e-commerce* do Brasil e elaborados alinhamentos para eles, a fim de avaliar a qualidade da saída criada pela aplicação desenvolvida.

1.3. ORGANIZAÇÃO DA DISSERTAÇÃO

Este volume encontra-se organizado da seguinte forma:

- O Capítulo 2 apresenta a fundamentação teórica e o estado da arte relativo ao trabalho conduzido.

- O Capítulo 3 mostra o desenvolvimento do *framework* PLATAL, detalhando sua arquitetura e o funcionamento cada um de seus módulos.

- O Capítulo 4 explora o uso da ferramenta PLATAL aplicada à realidade de comércio eletrônico e os resultados obtidos dos experimentos realizados, os avaliando principalmente em relação a outras ferramentas.

- O Capítulo 5, por fim, encerra com algumas considerações finais relevantes para a proposta deste trabalho.

2. FUNDAMENTAÇÃO TEÓRICA E ESTADO DA ARTE

Neste capítulo serão apresentados conceitos para melhor compreensão do método produzido, como a web semântica, o alinhamento de ontologias, o *e-commerce* e a IC; além de alguns conceitos técnicos da área de PLN aplicados a esse trabalho. Ainda serão abordados ao longo dele alguns trabalhos relacionados aos tópicos supracitados.

2.1. WEB SEMÂNTICA

Tim Berners-Lee, o “inventor” da internet, originalmente expressou a ideia da web semântica como o seu sonho para a internet, onde os computadores se tornariam capazes de analisar todos os dados da rede (o conteúdo, atalhos e transações entre pessoas e computadores), como uma “rede semântica”. Ele acredita que, quando a mesma for disponibilizada por completo, os processos diários de negociação, a burocracia derivada deles e as nossas vidas diárias de forma geral serão tarefas gerenciadas por máquinas conversando com máquinas. Com isso, os agentes inteligentes que as pessoas imaginaram por muito tempo irão se tornar viáveis [5].

De acordo com a World Wide Web Consortium (W3C), organização internacional que promove formatos padronizados para dados na web, a web semântica é um *framework* padronizado que permite dados serem compartilhados e reutilizados por aplicações, empresas e pela comunidade [53].

No momento, a web semântica está focada em tornar a informação com que lidamos na web legível para máquinas, propondo a conversão de documentos para dados. Humanos são capazes de interpretar informações visíveis em uma página da internet, em um e-mail, um relatório, etc. Porém máquinas precisam que a informação esteja organizada e/ou anotada, de forma que seja possível para elas realizar a mesma interpretação que humanos. Para isso, se faz necessária a transformação de documentos não estruturados em dados semanticamente estruturados.

Analisando em um nível mais técnico, podemos partir da análise que um documento apresentado em Hypertext Markup Language (HTML), que tipicamente não possui a intenção de prover uma forma para máquinas categorizarem o conteúdo da página (ainda que existam iniciativas que anotam trechos do HTML para torná-lo mais

legível para máquinas), mas sim apresentar informação para humanos lerem. A solução proposta pela web semântica deixa que o HTML se encarregue de descrever documentos e relacionamentos (*links*) entre eles, enquanto aplica tecnologias (linguagens) próprias para o significado (dados), tais como *Extensible Markup Language* (XML), *Resource Description Framework* (RDF), *Web Ontology Language* (OWL), etc. Essas, por sua vez, atuam de forma a complementar ou substituir o conteúdo de documentos, adicionando semântica aos mesmos, e assim permitindo a uma máquina processar o documento de forma a ser capaz de realizar algumas operações, como por exemplo, a inferência.

2.1.1. ONTOLOGIAS E OWL

Uma ontologia, para ciência da computação, de acordo com Studer, Benjamins e Fensel [49], é uma especificação formal e explícita de um conceito compartilhado. Conceito se refere a um modelo comum de um fenômeno no mundo, através da identificação das entidades / classes relevantes do mesmo. Explícita significa que os tipos de conceitos utilizados, e as restrições de seu uso são definidos de forma explícita (declarados sempre). Formal se refere ao fato de que a ontologia deve ser legível por máquinas, em um formalismo apropriado. Compartilhado reflete a noção de que uma ontologia captura um consenso sobre um conhecimento (é uma ideia aceita por um grupo, não apenas por um indivíduo).

Atualmente, ontologias são amplamente usadas para engenharia de software e de conhecimento e inteligência artificial. Também são utilizadas em aplicações relacionadas a essas áreas, como gerência de conhecimento, *e-commerce*, PLN, integração de informação, recuperação de informação, projeto e integração de bancos de dados, bioinformática, educação, etc [9]. Fensel [22] ainda as define como um recurso “definitivo” para resolver problemas de várias aplicações, que além das previamente citadas, ainda inclui outras áreas mais recentes que necessitam mapear uma representação do conhecimento, como sistemas de compartilhamento (*peer-to-peer*), *web services* semânticos e até mesmo redes sociais.

Apesar das diferentes linguagens que existem para definir uma ontologia, cada uma provendo componentes capazes de implementar uma ontologia, Studer, Benjamins e Fensel [49] definem os itens a seguir como o conjunto mínimo de componentes comum a todas:

- **Classes**: representam conceitos do domínio, às vezes também são referidas como entidades ou conjuntos. Classes em uma ontologia normalmente são organizadas em taxonomias hierárquicas, nas quais mecanismos de herança podem ser aplicados. Por exemplo, em um domínio de viagens, classes poderiam ser locais (cidades, estados, etc.), alojamentos (hotéis, acampamentos, etc.) e meios de transporte (aviões, carros, etc.).

- **Relacionamentos**: representam algum tipo de associação entre conceitos do domínio. Ontologias normalmente contêm relacionamentos binários, onde o primeiro argumento é o domínio (*domain*) da relação, e o segundo é o "alcance" (*range*). Em um relacionamento típico, o alcance é um outro conceito. Por exemplo, no domínio de viagens, um relacionamento possível seria o de "localizado em", onde o domínio é a classe "Alojamento", e o alcance a classe "Local".

Existe ainda o relacionamento implícito em toda classe, derivada de sua hierarquia, que é denominado de relacionamento "é um", para com a entidade pai. Por exemplo, em uma taxonomia em que a classe "Avião" é filha de "Transporte aéreo", que por sua vez é filha de "Meio de Locomoção", podemos deduzir a relação transitiva de que um avião é um transporte aéreo, assim como é um meio de locomoção.

Relacionamentos binários às vezes são usados para representar atributos de classes, sendo que alguns autores os classificam como um componente à parte da ontologia. Atributos se diferenciam de um relacionamento normal, pois seu alcance é um tipo de dado (*string*, numérico, etc.). Por exemplo, a relação / atributo "número do voo", tem como domínio a classe "Avião", enquanto o alcance é uma *string*.

- **Instâncias**: representam indivíduos ou elementos de uma classe, são a forma "concreta" dela. Por exemplo, a instância "Hotel Modelo", da classe "Hotel", que possui o relacionamento "vizinho de" com a instância "Albergue de Todos" da classe "Albergue", e o relacionamento / atributo "telefone" com a *string* "555-123456".

Desde 2004, a W3C passou a adotar (ainda em sua primeira versão) uma linguagem padronizada para descrever essas ontologias e seus componentes. Denominada OWL, que vem desde então se disseminando ao redor do mundo com grande velocidade. A linguagem utiliza descrições semânticas formais, e é baseada em RDF / XML.

Em 2009 a W3C publicou a especificação da versão 2 da OWL, denominada OWL 2 Web Ontology Language (OWL 2), que mantém compatibilidade com a primeira versão, e adiciona novas funcionalidades, como tipos de dados mais ricos, com intervalos de dados, restrições de cardinalidade para relações, propriedades novas (assimétrica, reflexiva, entre outras), etc. [52].

A W3C ainda define a OWL em três “sublinguagens”, para uso de acordo com comunidades e usuários específicos, a partir do abaixo descrito [46]:

- **OWL Lite**: é direcionada para suportar usuários que necessitam principalmente de uma classificação hierárquica e restrições simples. Por exemplo, embora suporte restrições de cardinalidade, ela só permite valores de cardinalidade 0 ou 1. Permite um caminho de migração mais rápido de tesauros e taxonomias.

- **OWL DL**: é direcionada a usuários que querem a máxima capacidade de expressão, enquanto mantém a computabilidade da ontologia (garantindo que as implicações sejam computáveis) e garantindo decidibilidade (todas as execuções devem terminar em tempo finito). OWL DL inclui todas as construções da linguagem OWL, porém com algumas restrições (uma classe não pode ser um indivíduo ou uma propriedade, por exemplo).

- **OWL Full**: é direcionada para aqueles que querem a máxima capacidade de expressão, sem estar presos à sintaxe do RDF e sem garantia da computabilidade. Por exemplo, uma classe pode ser tratada simultaneamente como uma coleção de indivíduos e como um indivíduo próprio.

Um exemplo visual de uma ontologia pequena pode ser visto na Figura 5, que apresenta uma versão simplificada de uma ontologia de turismo, com todos os tipos de componentes previamente vistos. Um exemplo prático do uso da linguagem OWL para essa ontologia pode ser visto na Figura 6, onde para construção foi utilizada a ferramenta de modelagem de ontologias Protégé [47].

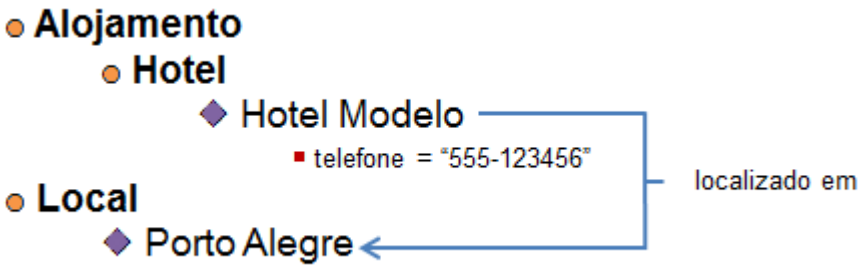

Figura 5 - Exemplo visual de ontologia de turismo simplificada

Na Figura 6, dentre as anotações relevantes, vemos o elemento (*tag*) “owl:Ontology”, que contém informações da ontologia, no caso contendo o mínimo de informações obrigatórias, que é URI único que a identifica. Em seguida temos as declarações das classes pertencentes a essa ontologia, com o elemento “owl:Class”. Vemos ainda que aninhado a esse elemento, tem-se o relacionamento de sub-classe (“é um”), que pode ser descrito com o elemento “rdfs:subClassOf“ (no exemplo mapeando um hotel como sub-classe de alojamento).

Vemos na Figura 6 também duas formas de armazenar relacionamentos entre classes, uma que utiliza o elemento “owl:ObjectProperty”, mapeando duas classes; e a outra utilizando “owl:DatatypeProperty”, mapeando um conceito para um tipo de dado. Esse último constituindo o caso que vimos de um relacionamento que representa um atributo da classe (telefone de alojamentos no exemplo).

Por fim, temos os alinhamentos de indivíduos, das classes “Hotel” e “Local”, com o uso do elemento “owl:NamedIndividual”. Junto ao indivíduo “Hotel_Modelo”, podemos ainda ver a notação dos seus relacionamentos, com os elementos de nome iguais aos relacionamentos, “telefone” e “localizado_em”.

```
<?xml version="1.0"?>

<rdf:RDF xmlns="http://exemplo.dissertacao/bernardo/turismo.owl#"
     xml:base="http://exemplo.dissertacao/bernardo/turismo.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:turismo="http://exemplo.dissertacao/bernardo/turismo.owl#">

    <!-- Ontologia -->
    <owl:Ontology rdf:about="http://exemplo.dissertacao/bernardo/turismo.owl"/>

    <!-- Classes -->
    <owl:Class rdf:about="Alojamento"/>
    <owl:Class rdf:about="Local"/>
    <owl:Class rdf:about="Hotel">
        <rdfs:subClassOf rdf:resource="Alojamento"/>
    </owl:Class>

    <!-- Relacionamentos -->
    <owl:ObjectProperty rdf:about="localizado_em">
        <rdfs:domain rdf:resource="Alojamento"/>
        <rdfs:range rdf:resource="Local"/>
    </owl:ObjectProperty>

    <owl:DatatypeProperty rdf:about="telefone">
        <rdfs:domain rdf:resource="Alojamento"/>
        <rdfs:range rdf:resource="string"/>
    </owl:DatatypeProperty>


    <!-- Indivíduos -->
    <owl:NamedIndividual rdf:about="Hotel_Modelo">
        <rdf:type rdf:resource="Hotel"/>
        <telefone>555-123456</telefone>
        <localizado_em rdf:resource="Porto_Alegre"/>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="Porto_Alegre">
        <rdf:type rdf:resource="Local"/>
    </owl:NamedIndividual>

</rdf:RDF>
```

Figura 6 - Exemplo em OWL de ontologia de turismo simplificada

2.2. ALINHAMENTO DE ONTOLOGIAS

Alinhamento de ontologias ou correspondência de ontologias (*Ontology Matching*), às vezes também referido como integração de ontologias, é a atividade de buscar correspondências entre conceitos de diferentes ontologias. Em sistemas distribuídos e abertos, como a web semântica e outras aplicações que utilizam ontologias, a heterogeneidade não pode ser evitada [18]. O conjunto das correspondências encontradas entre duas ontologias diferentes é chamado de alinhamento [18]. As correspondências podem ser usadas para diversas tarefas, como união (*merge*) de ontologias, resposta para buscas, tradução de dados, ou até mesmo para a navegação na web semântica; possibilitando assim a interoperabilidade para sistemas entre as diferentes ontologias.

Os diferentes modelos de ontologias surgem devido ao seu uso e objetivo particular para uma aplicação, estando sujeitos muitas vezes à visão aplicada no momento da engenharia da ontologia, que visa os objetivos para o qual foi projetada. As diferentes formas em que as ontologias aplicadas se encontram (diferentes realidades de negócio) também contribuem para as diferenças surgirem, uma vez que elas podem ter diferentes origens (bancos de dados, diretórios, etc.) e até mesmo diferentes linguagens (que tipicamente compartilham os mesmo conceitos).

O principal objetivo do alinhamento de ontologias é reduzir as diferenças entre as ontologias, que podem ocorrer de diversas formas e em mais de uma forma ao mesmo tempo. Euzenat e Shvaiko [18] categorizam algumas dessas formas dos tipos de heterogeneidade da seguinte forma:

- **Heterogeneidade sintática**: quando duas ontologias não são expressas na mesma linguagem. Pode ocorrer quando se compara estruturas de origens completamente diferentes, por exemplo, um diretório e um modelo conceitual de banco de dados. Mas o caso mais comum de se encontrar o problema em um cenário real é o de que as ontologias foram escritas em duas formas de representação do formalismo diferentes. Por exemplo, uma em OWL e outra em F-Logic. Assim, muitas vezes é possível “traduzir” ontologias de diferentes linguagens preservando o significado [19].

- **Heterogeneidade terminológica:** determina a ocorrência de variações de nomes quando se referindo à mesma entidade em diferentes ontologias. Isso pode ocorrer pelo

uso de diferentes linguagens naturais (por exemplo, português e inglês), diferentes formas técnicas de tratamento da entidade (por exemplo, “Papel” e “Documento”), ou até mesmo o uso de sinônimos (por exemplo, “Carro” e “Automóvel”).

- **Heterogeneidade conceitual:** também chamada de heterogeneidade semântica, se refere à diferença que ocorre ao modelar um mesmo domínio. Isso pode ocorrer devido ao uso de diferentes axiomas para definir os conceitos. Pode ocorrer também quando duas ontologias descrevem um mesmo domínio, porém com um nível de detalhe diferente, ou de perspectiva.

- **Heterogeneidade semiótica:** também chamada de heterogeneidade pragmática, está relacionada ao modo como as entidades são interpretadas por pessoas. Dois diferentes analistas podem classificar uma determinada entidade de forma diferente. Esse tipo de heterogeneidade ainda é muito difícil de detectar e resolver automaticamente (por um computador).

Para esse trabalho será adotado o processo de alinhamento de ontologias definido por Euzenat e Shvaiko [18], que podemos descrever como uma função:

$$\mathbf{A'} = f(O_1, O_2, A, P, R)$$

Essa função recebe como entrada duas ontologias ' O_1' e ' O_2' para alinhamento, e opcionalmente um conjunto de parâmetros 'P' e um conjunto de recursos 'R'. Como entrada é possível também prover um alinhamento ‘A’, que caso possua correspondências pode ser utilizado durante o processo para decisões e posteriormente complementado com as novas descobertas, ou, caso “vazio”, simplesmente servir para ser completado e se tornar a saída da função. Vemos a organização do processo descrito pela função na Figura 7**.** É possível ainda trabalhar com um processo de alinhamento de múltiplas ontologias (mais de duas), onde as mesmas passam a serem entradas para a função descrita também.

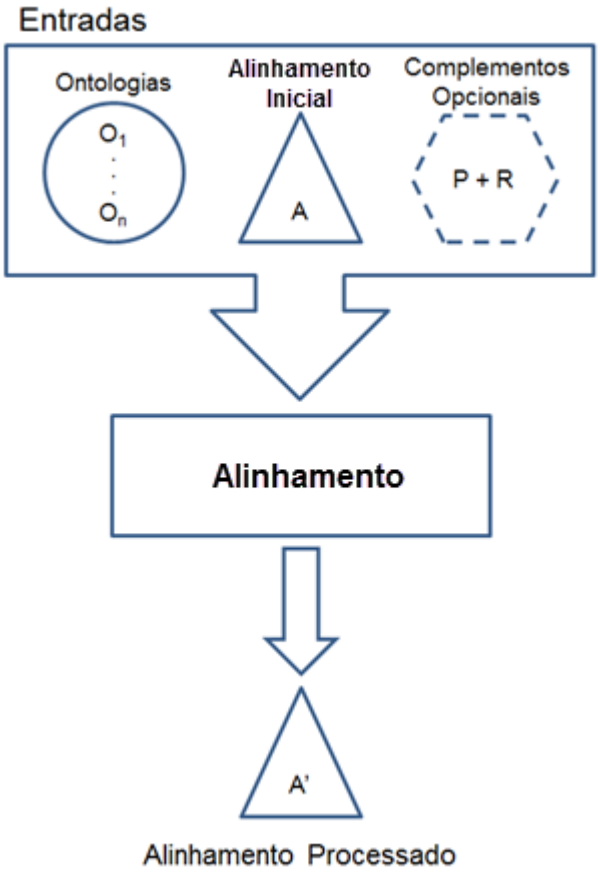

Figura 7 - Processo de alinhamento de ontologias

O alinhamento resultante de um processo de alinhamento de ontologias é composto por um conjunto de correspondências entre as ontologias envolvidas. Um fator importante nisso é a relação existente entre as entidades das diferentes ontologias. Apesar de os algoritmos de comparação mais simples utilizarem tipicamente o relacionamento mais comum, o de equivalência (=), ele não é o único possível. Relações como as definidas em OWL, por exemplo, a de disjunção (etiqueta “owl:disjointWith”) ou de generalização (etiqueta “rdfs:subClassOf”), podem ser utilizadas. Por fim, qualquer relacionamento pode ser descrito, independente de existir em uma linguagem formal.

Como parte do alinhamento, ainda temos associado a cada relação identificada um fator de confiança, que expressa (mede) o quanto podemos acreditar no fato que aquela relação contém. Por exemplo, podemos dizer que uma classe “Telefone Móvel” é equivalente a “Telefone” com uma confiança de 70%. Possibilitando assim que se utilize essa medida como forma de “filtrar” os relacionamentos mais relevantes, ou aceitáveis.

A cardinalidade de um relacionamento ainda pode ser mais complexa que o simples 1:1. Alinhamentos múltiplos são possíveis, por exemplo, quando mais de duas

ontologias estão envolvidas, ou quando alinhamentos são encontrados para mais de uma entidade destino.

2.2.1. TÉCNICAS PARA ALINHAMENTO DE ONTOLOGIAS

Diversas abordagens são possíveis quando se trata de descobrir alinhamentos entre ontologias. A Figura 8 e a Figura 9 mostram uma estrutura que classifica as técnicas de acordo os tipos de técnicas para alinhamento de ontologias (baseadas na classificação dada por Euzenat e Shvaiko [18]), em nível de elemento e em nível de estrutura, respectivamente. Podemos descrever essas diferentes classificações da seguinte forma:

- **Nível de elemento**: técnicas que processam correspondências a partir da análise de entidades das ontologias ou suas instâncias apenas. Isto é, as relações com outras entidades ou instâncias não são levadas em conta.

- **Nível de estrutura**: ao contrário das de nível de elemento, são técnicas que não desprezam os relacionamentos. Elas os utilizam para analisar como as entidades e instâncias são compostas em suas redes de relacionamentos.

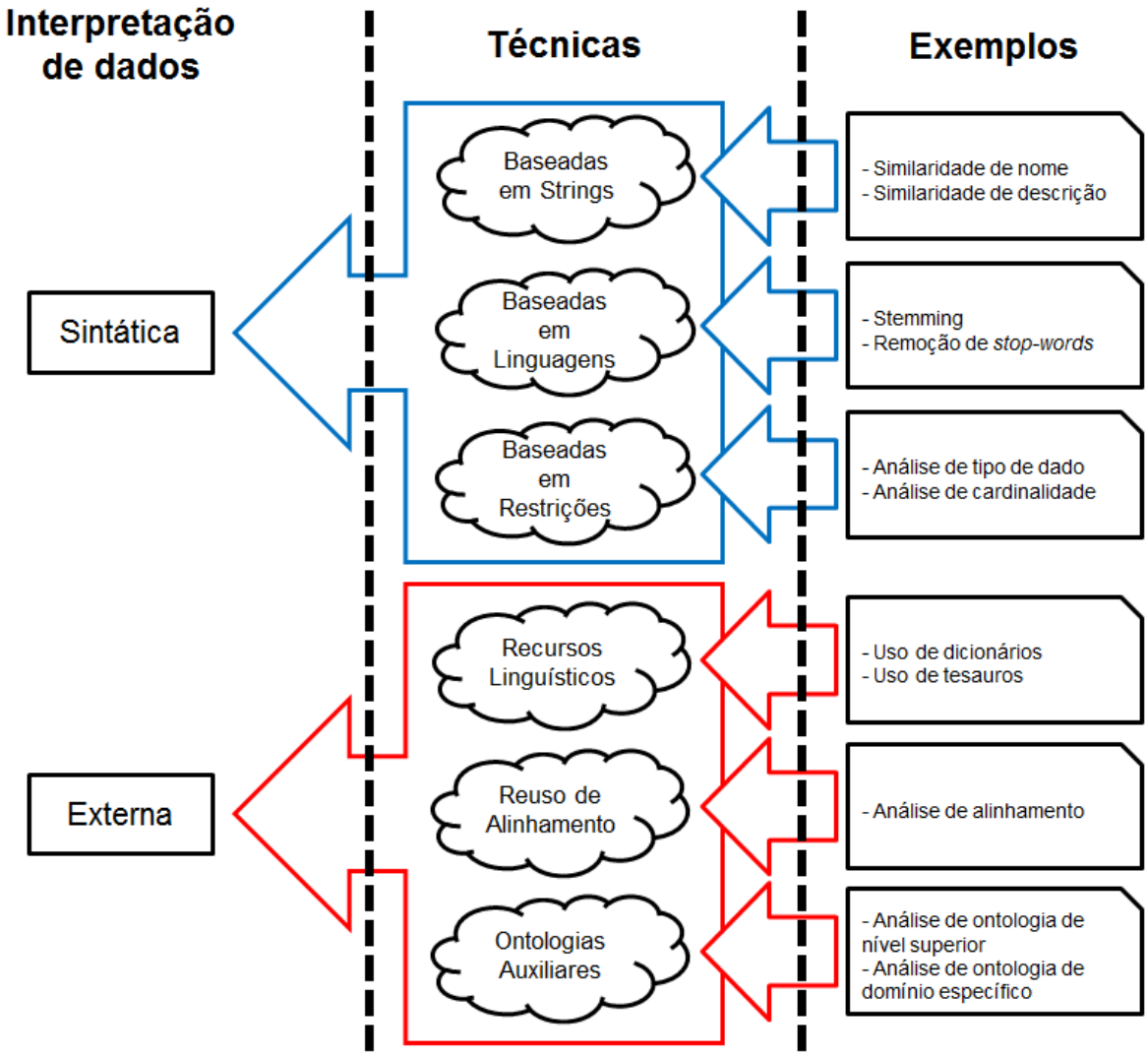

Figura 8 - Classificação de técnicas de alinhamento de ontologias em nível de elemento, baseada na classificação de Euzenat e Shvaiko

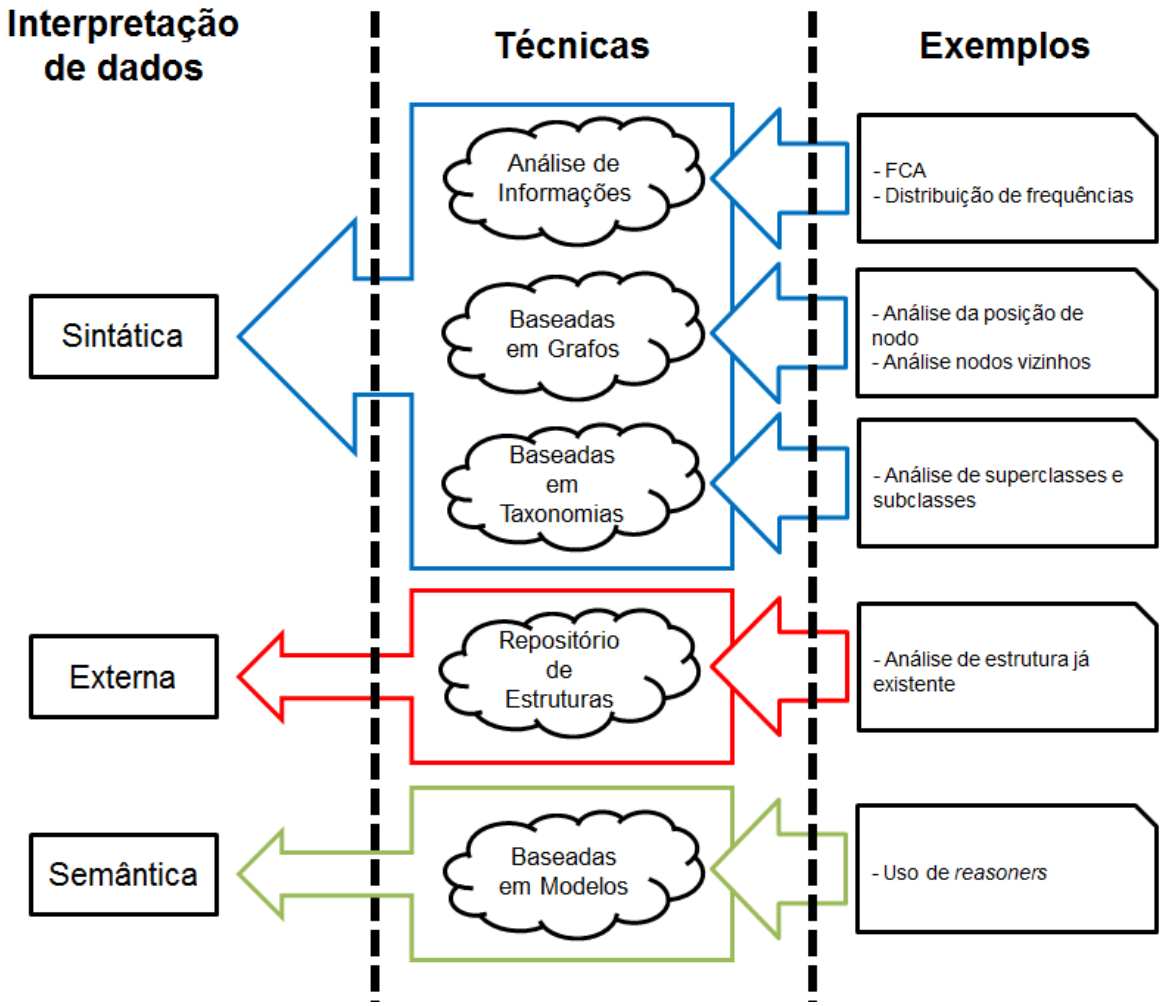

Figura 9 - Classificação de técnicas de alinhamento de ontologias em nível de estrutura, baseada na classificação de Euzenat e Shvaiko

Para a classificação da interpretação de dados (aplicada tanto em nível de elemento quanto de estrutura), podemos analisar as três variações presentes nas classificações:

- **Técnicas sintáticas**: interpretam a entrada de dados levando em consideração apenas sua estrutura gramatical, seguindo um processo algorítmico bem definido.

- **Técnicas externas**: tiram proveito de um conhecimento comum ou informação auxiliar, providos através de um recurso extra na entrada de dados além das ontologias para interpretar as relações.

- **Técnicas semânticas**: utilizam, como o nome sugere, recursos formais de semântica, para interpretar a entrada de dados e justificar a saída encontrada. De forma que seus algoritmos são capazes de descobrir todos os relacionamentos possíveis (fortemente relacionados a web semântica).

Pode-se então observar a descrição detalhada de cada uma das classificações de técnicas apresentadas para classificação **em nível de elemento**:

- **Técnicas baseadas em *strings***: são tipicamente voltadas para fazer a correspondência de nomes ou descrições de entidades das ontologias. O mais comum é realizar uma análise das *strings*, de forma que quanto mais próximas (similares) elas forem, maior a probabilidade de que elas representem o mesmo conceito. Essas técnicas tratam uma *string* simplesmente como uma sequência de caracteres, sem análise semântica.

As diferenças entre os nomes podem ocorrer basicamente por quarto motivos: as entidades são diferentes e representam diferentes classes; as entidades apresentam diferentes nomes (por exemplo, por serem sinônimos ou homônimos); estão escritas em diferentes línguas; são apresentadas de forma abreviada. O que se busca com essas técnicas é resolver o problema para quando as entidades representam a mesma entidade.

Tipicamente implementações desse tipo de técnica mapeiam um par de *strings* para um número real, que indica o quanto as duas *strings* são similares. Exemplos de técnicas seriam as que se baseiam em, além dos nomes como um todo, normalizar o texto através de: *subtrings* dos nomes para comparação; remoção de acentos; conversão para minúsculo; remoção de espaços em branco excedentes; remoção de números; remoção de pontuação; entre outras.

Outros exemplos de métodos mais robustos seriam: o uso de *edit-distance* (como a distância de Levenstein), que calcula o número mínimo de alterações necessárias para transformar uma *string* em outra, auxiliando a tratar erros de digitação; a distância de Hamming, que calcula o número de caracteres diferentes entre duas *strings*; *n-gram-distance*; entre outras medidas baseadas em distância, similaridade, e análise de *tokens*.

Devido aos exemplos acima mencionados, esse grupo de técnicas está fortemente relacionado à língua em que está escrita a ontologia. Além de que, algumas das implementações das técnicas podem levar a uma perda de informação (devido a redução visando aproximar *strings*, por exemplo), situação essa que deve ser levada em consideração e pesada ao elaborar os métodos.

- **Técnicas baseadas em linguagens**: são baseadas em técnicas de PLN, analisando propriedades morfológicas das palavras ou sentenças dadas como entrada. Portanto, nessas técnicas, um nome / *string* é considerado uma palavra em uma língua natural, como o português ou inglês. Podem ainda analisar tanto as classes como as instâncias.

Tipicamente, essas técnicas são aplicadas aos nomes das entidades, antes de executar técnicas baseadas em *strings* ou dicionários, visando melhorar os resultados. Podendo ser consideradas uma classe à parte de técnicas de alinhamento, visto que elas podem ser estendidas para, por exemplo, um cálculo de distância (comparando as *strings* resultantes de um conjuntos de *strings*).

Alguns exemplos de métodos baseados em linguagens são: o de comparação do "caminho" das entidades (*path distance*), que está relacionado a técnicas baseadas em taxonomias, através da concatenação de todas as subclasses e superclasses de uma entidade e análise disso como um todo; análise de *tokens*; lematização e *stemming,* que consistem em reduzir ou converter uma palavra a um radical ou forma básica comum; remoção de *stop words*; entre outras.

- **Técnicas baseadas em restrições**: são algoritmos que lidam com restrições internas aplicadas na definição de entidades da ontologia, como tipos, cardinalidade de atributos, identificadores, etc.

- **Técnicas de recursos linguísticos**: utilizam recursos linguísticos, como dicionários e tesauros específicos do domínio. Com o suporte desses recursos, procuram encontrar associações entre nomes de entidades (aqui consideradas palavras / sentenças em língua natural). Um exemplo de análise que pode ser feita é a comparação de sinônimos ou hiperônimos.

Alguns exemplos dessas técnicas incluem o uso auxiliar de recursos externos às estruturas conhecidas, como: dicionários para sinônimos; dicionários de tradução de línguas, para quando se tratar de mais de uma língua a análise; tesauros, para análise de hiperônimos / hipônimos; terminologias; entre outros.

- **Técnicas de reuso de alinhamento**: é um meio de utilizar recursos externos de outra forma, onde alinhamentos de ontologias previamente mapeadas são aproveitados para análise desse novo alinhamento. São indicadas pela premissa que muitas ontologias

já alinhadas são similares às que serão mapeadas, principalmente se tratando de um mesmo domínio.

Por exemplo, se possuímos de antemão o alinhamento de uma ontologia O_1 com uma ontologia O_2, e o alinhamento da ontologia O_1 com O_3; podemos utilizar esse conhecimento para "passar" pelos alinhamentos conhecidos de O_1 para auxiliar a deduzir os alinhamentos entre O_2 e O_3.

- **Técnicas de ontologias auxiliares:** englobam o uso de ontologias de nível superior e de domínio específico. Ambas são baseadas em lógica e, portanto, técnicas que as exploram são baseadas em semântica.

Ontologias de nível superior podem ser usadas como fonte de conhecimento comum, uma vez que comportam conceitos gerais, que devem ser os mesmos em todos os domínios de conhecimento.

Ontologias de domínio específico, ao contrário, utilizam termos de um domínio em particular, de uma forma que são relevantes apenas para esse domínio, e não relacionadas a conceitos similares em outros domínios. Por exemplo, uma ontologia de biblioteca pode ser usada como contexto para uma ontologia de livros, provendo parte da estrutura.

E por fim, é possível observar a descrição detalhada de cada uma das classificações de técnicas apresentadas para classificação **em nível de estrutura**:

- **Técnicas de análise de informações:** realizam uma análise de dados e estatísticas disponíveis para o contexto, a fim de encontrar conformidades e discrepâncias para calcular a similaridade de entidades. Para isso levam em consideração inclusive as instâncias da ontologia em análise.

Exemplos de técnicas desse tipo são classificações baseadas em distância, *formal concept analysis* (FCA), distribuição de frequências (técnica estatística usada para apresentar uma coleção de objetos classificados de modo a mostrar o número existente em cada classe).

- **Técnicas baseadas em grafos:** são técnicas que consideram as ontologias de entrada como uma estrutura de grafos rotulados. Existem ainda abordagens que consideram formas de grafos mais específicos, como as que utilizam a estrutura de árvores. Tipicamente a análise de similaridade ocorre baseada na posição dos nodos

dentro dos grafos montados. Por exemplo, se dois nodos são similares, seus vizinhos devem ser de alguma forma similares também.

- **Técnicas baseadas em taxonomias:** também são técnicas baseadas em grafos, porém considerando somente a relação hierárquica de especialização. Portanto, se comparados os subconjuntos de termos superiores e inferiores de um determinado termo, com os mesmos conjuntos de outros, espera-se que eles sejam similares.

Por exemplo, se o conjunto de subclasses de duas entidades ou a superclasse da entidade têm o mesmo valor, é extremamente provável que elas sejam similares. Um problema dessa abordagem reside no fato de que, se analisada por completa a taxonomia de uma classe, eventualmente classes de distantes caminhos na árvore acabarão tendo uma ou mais classes bases em comum (a raiz, por exemplo); sendo necessário nesses casos utilizar algum outro critério para discriminar.

- **Técnicas de repositórios de estruturas:** utilizam repositórios de estruturas de ontologias e / ou fragmentos de ontologias, onde a cada uma dessas estruturas está associado um valor de referência para a similaridade entre as ontologias comparadas. Quando se deseja que novas estruturas (ontologias ou partes de uma) sejam mapeadas, é primeiro realizada uma consulta a esses repositórios, para avaliar a possível relevância de comparar essas duas estruturas. Essas técnicas se diferenciam das de reuso de alinhamento por não guardarem o alinhamento entre duas ontologias, mas apenas a similaridade de duas ontologias; para tal, a similaridade de duas estruturas deve ser computacionalmente mais “barata” de se medir que análise das mesmas por completo.

Um exemplo de implementação de tais técnicas pode ser a abordagem de Rahm [38], que para mapear duas estruturas, propõe usar alguns meta-dados de descrição, tais como nome da estrutura, número de nodos, nome do(s) nodo(s) raiz, profundidade máxima da árvore, etc. Ele então agrega essas informações para gerar um coeficiente de similaridade.

- **Técnicas baseadas em modelos:** interpretam os dados baseadas em seu significado semântico. São portando métodos fortemente dedutivos, baseados em informações já conhecidas das entidades, pois se duas entidades são a mesma, elas compartilham a mesma interpretação. Um exemplo de implementação seria o uso de *reasoners* para inferência de equivalências entre entidades.

2.3. ONTOLOGY ALIGNMENT EVALUATION INITIATIVE

O International Semantic Web Conference (ISWC) [29] é o maior fórum internacional para as comunidades que trabalham com web semântica e dados relacionados (*Linked Data*, método de conectar dados distribuídos através da internet). Nele são apresentadas as mais recentes pesquisas e resultados, inovações técnicas, e discutido o futuro de soluções inovadoras dessas áreas.

Junto com a conferência ISWC, ocorre o International Workshop on Ontology Matching [37], *workshop* focado em atividades de alinhamento de ontologias. Esse *workshop* tem como objetivo reunir líderes de universidades, empresas e interessados para avaliar os avanços acadêmicos em como lidar com problemas do mundo real.

Por fim, associada a esse *workshop* está a Ontology Alignment Evaluation Initiative (OAEI) [36]. Desde 2004, a OAEI organiza campanhas com o objetivo de avaliar tecnologias de alinhamento de ontologias (*Ontology Matching*). Essa necessidade surgiu do crescimento do número de métodos disponíveis para realizar integrações entre ontologias e fazer correspondência de esquemas (*Schema Matching*) [36].

O objetivo dessa iniciativa é verificar e avaliar quais técnicas são capazes de encontrar correspondências corretas em um conjunto de ontologias que descrevem um domínio comum. As ontologias disponíveis são descritas em OWL-DL e os alinhamentos providos pelos participantes devem ser em um formato próprio (provido pela Alignment API), e ambos são serializados no formato RDF / XML. Os resultados enviados são então avaliados automaticamente através de um alinhamento de referência (construído e / ou revisado manualmente).

A OAEI recomenda aos participantes o uso da Alignment API, um framework criado para atividades comuns ao alinhamento de ontologias que é detalhado na Subseção 2.5.1. Alguns dos serviços providos e que são de uso para os participantes (e em parte da Etapa 2 desse estudo) são: desenvolvimento de algoritmos de alinhamento com a criação de processos de alinhamento e *matchers*, busca de alinhamentos nas ontologias, melhora de algoritmos, comparação de alinhamentos, manipulação de saída dos dados, entre outros.

2.3.1. MATCHERS DA OAEI

Alguns *matchers* usados em edições da OAEI serão utilizados nesse tranbalho para avaliações de resultados, portanto o mesmos são melhor detalhados nesse sub-capítulo.

O *matcher* de ontologias AROMA [4] é projetado para encontrar relações de equivalência e subbmissão entre as entidades, classes e propriedades derivadas de duas taxonomias textuais ou ontologias OWL. Ele utiliza o parafigma da regra de associação e uma medida estatística de interesse, que implica na intensidade da relação. O AROMA baseia-se no seguinte pressuposto: "Uma entidade A será mais específica ou equivalente a uma entidade B se o vocabulário usado para descrever A, seus descendentes e sua instâncias tender a a incluir o mesmo vocabuláio que B".

O *matcher* AFLOOD [2] utiliza como essência do seu algoritmo de corrêspondência a localização de refência, levando em consideração entidades vizinhas e relações para alinhar conceitos de ontologias. Ele começa de um ponto chamado "âncora", e a partir dele coleta dois blocos de conceitos vizinhos através da ontologia. Esses blocos são então alinhados e o processo se repete.

O *matcher* Falcon [20] propõe uma série de diferentes abordagens para encontrar corrêspondências para diferentes (porém relacionadas) ontologias. Atualmente estão inclusas técnicas estruturais iterativas, de corrêspondência linguística e de particionamentos de grandes ontologias.

2.4. SIMPLE KNOWLEDGE ORGANIZATION SYSTEM

O Simple Knowledge Organization System (SKOS) [45] é um modelo de dados comum para compartilhamento de vocabulários estruturados e controlados para web semântica. Assim como o OWL, o formato SKOS é um padrão da W3C, e construído baseado em RDF, podendo inclusive ser usado combinado com OWL e outras linguagens formais de representação de conhecimento.

Diversas aplicações compartilham uma estrutura similar para seus dados (esquemas de conceitos), como tesauros, taxonomias, esquemas de classificação, etc. O SKOS captura grande parte dessa similaridade e a torna explícita, para disponibilizar dados e compartilhamento de tecnologia para diversas aplicações [3].

No SKOS básico, conceitos são identificados com URIs (assim como OWL), nomeados com uma ou mais *strings*, em uma ou mais linguagens naturais. Podem ainda ser documentados com vários tipos de anotações, relacionados semanticamente entre si e agregados em esquemas de conceitos.

Em um SKOS mais avançado, os conceitos podem ser mapeados entre diferentes esquemas de conceitos, ou ainda agrupados de acordo com seus nomes, visto que um conceito pode ter diversos (chamados de nomes alternativos). Relacionamentos podem inclusive ser criados entre nomes específicos de conceitos. Por fim, o próprio vocabulário da linguagem pode ser estendido para se adaptar às necessidades do usuário.

Na Figura 10 vemos um exemplo do conceito “Celular” (onde a etiqueta “skos:Concept” assinala conceitos da estrutura), anotado com linguagem SKOS. Existem ainda outras etiquetas “superiores” que agrupam conjuntos de conceitos, que poderiam ser usados para mapear inclusive relacionamentos entre esses diferentes conjuntos.

As etiquetas “skos:prefLabel” e “skos:altLabel” são então usadas par armazenar formas de referência ao conceito, como vemos com os dois nomes típicos para o conceito, “Celular” e “Telefone Móvel”. Tipicamente a etiqueta “skos:altLabel” armazena sinônimos para o conceito, podendo ambas as etiquetas ainda armazenar nomes alternativos em outras línguas naturais, conforme vemos denotado pelo atributo “xml:lang”, onde temos o valor em inglês para o conceito: “Cell Phone”.

Vemos ainda duas etiquetas que representam relacionamentos, a “skos:broader”, que sinaliza um relacionamento em que o conceito indicado é diretamente superior na hierarquia, como a categoria “Telefonia”. Enquanto a etiqueta “skos:related” é usada para simplesmente declarar um relacionamento qualquer entre dois conceitos, como a relação de celulares com “Smartphones”.

```
<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#">

  <skos:Concept rdf:about="http://exemplo.dissertacao/bernardo/ecommerce/#Celular">
    <skos:definition>Categoria que contém aparelhos de telefonia móvel.</skos:definition>
    <skos:prefLabel xml:lang="pt">Celular</skos:prefLabel>
    <skos:altLabel xml:lang="pt">Telefone Móvel</skos:altLabel>
    <skos:altLabel xml:lang="en">Cell Phone</skos:altLabel>
    <skos:broader rdf:resource="http://exemplo.dissertacao/bernardo/ecommerce/#Telefonia"/>
    <skos:related rdf:resource="http://exemplo.dissertacao/bernardo/ecommerce/#Smartphone"/>
  </skos:Concept>

</rdf:RDF>
```

Figura 10 - Exemplo de esquema de classificação SKOS

2.5. OUTRAS TECNOLOGIAS

Nessa subseção, serão abordados alguns tópicos relacionados a tecnologias empregadas mais especificamente para a programação da solução descrita nesse trabalho.

2.5.1. ALIGNMENT API

A Alignment API é um *framework* para implementar, expressar e compartilhar alinhamentos de ontologias [12]. Desenvolvido em tecnologia Java, permite a manipulação de alinhamentos, desde a interpretação das ontologias, através do processo de alinhamento, avaliação de resultados (comparação de alinhamentos), até a saída resultante (o alinhamento das entidades).

O formato usado para exibir a saída do processamento, isso é, o alinhamento em si, adotado por mais aplicações, é definido por Euzenat [17]. O formato é escrito usando RDF / XML, sendo portanto, livre de uma linguagem de programação específica. Um exemplo desse formato pode ser visto na Figura 11.

Como detalhes da etiqueta “Alignment”, que engloba um alinhamento, temos o elemento “level”, que pode ter valor “0”, quando é apenas o alinhamento entre duas entidades, valor “1” quando é um alinhamento entre conjuntos de entidades e valor “2” para tipos de alinhamentos mais presos a uma linguagem específica, permitindo o uso de

lógica de primeira ordem para expressar os alinhamentos. Existe ainda a etiqueta ”type”, que indica o tipo de alinhamento (cardinalidade), como “11” (1:1), “**” (n:m), etc. Além dos URIs que identificam as ontologias sendo analisadas (etiquetas “onto1” e “onto2”).

Após esses elementos básicos, temos uma etiqueta “map” e uma série de etiquetas “Cell”, que armazenam os alinhamentos encontrados entre duas entidades das ontologias. Nelas, as etiquetas “entity1” e “entity2” armazenam os URIs das entidades que possuem o relacionamento. Enquanto a etiqueta “measure” contém um valor entre 0 e 1, com a confiança na relação mapeada, que por sua vez é descrita pela etiqueta “relation”.

```
<?xml version="1.0"?>

<rdf:RDF xmlns="http://knowledgeweb.semanticweb.org/heterogeneity/alignment"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

    <Alignment>
        <xml>yes</xml>
        <level>0</level>
        <type>**</type>
        <onto1>http://exemplo.dissertacao/ontologia1</onto1>
        <onto2>http://exemplo.dissertacao/ontologia2</onto2>
        <map>
            <Cell>
            <entity1 rdf:resource="http://exemplo.dissertacao/ontologia1#Literatura_Estrangeira"/>
            <entity2 rdf:resource="http://exemplo.dissertacao/ontologia2#Literatura_Estrangeira"/>
            <measure rdf:datatype="&xsd;float">1.0</measure>
            <relation>=</relation>
            </Cell>

            <Cell>
            <entity1 rdf:resource="http://exemplo.dissertacao/ontologia1#romance"/>
            <entity2 rdf:resource="http://exemplo.dissertacao/ontologia2#romance_policial"/>
            <measure rdf:datatype="&xsd;float">0.6725</measure>
            <relation>=</relation>
            </Cell>
        </map>
    </Alignment>
</rdf:RDF>
```

Figura 11 - Exemplo do formato de alinhamento da Alignment API

É possível ainda estender (implementando novas técnicas de correspondência, por exemplo) a API, através da customização de seus componentes, como:

- **AlignmentProcess:** é uma interface que deve ser implementada para cada algoritmo de alinhamento, com um processo bem definido para a realização do mesmo. Ela provê a lógica do método de alinhamento em si, que pode consultar *matchers* para tal;

- **Matcher**: implementa uma forma de alinhar um relacionamento entre duas entidades, com uma determinada confiança;

- **Relation**: pode ser estendida para declarar novos tipos de relações;

- **Evaluator**: é uma interface que descreve a comparação de dois alinhamentos (por exemplo, o primeiro pode ser utilizado como o de referência);

- **Renderer**: permite criar novas formas de renderizar o alinhamento resultado, podendo assim alterar o formato para um diferente do padrão.

2.5.2. XPATH

A XML Path Language (Xpath) é uma linguagem recomendada e padronizada pela W3C, projetada para a realização de consultas em arquivos XML, selecionando elementos do mesmo. A XPath opera na estrutura abstrata e lógica de um documento XML. O nome é originado da notação utilizada, que é semelhante a um caminho (*path*) de diretórios ou de URLs, para então realizar a navegação entre as estruturas hierárquicas do documento XML [10].

A principal unidade sintática do Xpath é a expressão. Uma expressão é capaz de simbolizar a busca por resultados e retorná-los em uma das quatro estruturas básicas: um conjunto de nodos (sem ordem e sem duplicados); um valor booleano, um valor numérico ou uma *string*.

Além das expressões para caminhos, a Xpath ainda possui mais de 100 funções incluídas na linguagem que possibilitam a manipulação de *strings*, números, datas, comparações de tipos, além da manipulação de nodos (como obter o próximo, filhos, etc).

```xml
<?xml version="1.0"?>

<categoria_livros>

<livro>
  <titulo ling="eng">Learning XPath</titulo>
  <titulo ling="pt">Aprendendo XPath</titulo>
  <preco>40.50</preco>
</livro>

<livro>
  <titulo ling="eng">Ontology Matching</titulo>
  <preco>16.90</preco>
</livro>

</categoria_livros>
```

Figura 12 - Exemplo XML de uma estrutura de classificação de livros

Alguns exemplos de expressões aplicadas a um documento XML, apresentado na Figura 12, podem ser visualizados na Tabela 1. Essa tabela apresenta as expressões simples para busca de nodos, onde mostra exemplos para uso dos recursos básicos de consulta, como:

- “/”: referencia o nodo raiz;
- “//”: indica um caminho qualquer;
- “.”: seleciona o nodo atual;
- “[]”: verifica se a condição interna é verdadeira (usado para construção de filtros);
- “@”: seleciona atributos;
- “text()”: seleciona o texto do nodo atual;
- “contains(texto_analise, texto_buscado)”: verifica se o texto “texto_analise” contém o valor da *string* “texto_buscado”.

Tabela 1 - Exemplos de expressões XPath para seleção

XPath	Descrição	Retorno do XML Exemplo
/categoria_livros	Nodo raiz	Elemento <categoria_livros>
/categoria_livros/livro	Todos os livros filhos da categoria livros	<livro> referente a Ontology Matching e <livro>referente a Learning XPath
//@ling	Todos os atributos com nome “ling”	ling="eng" e ling="pt" e ling="eng"
/categoria_livros/livro[last()]	O último elemento livro filho da categoria livros	<livro> referente a Ontology Matching
//titulo[@ling='eng']	Todos os títulos com língua em inglês	<titulo ling="eng"> Learning XPath </titulo> e <titulo ling="eng">Ontology Matching </titulo>
//*[@ling="pt"]/text()	O texto de qualquer elemento com língua em português	“Aprendendo XPath”
//titulo[contains(text(), 'XPath')]	Todos os títulos que contêm o texto ‘XPath’	<titulo ling="eng"> Learning XPath </titulo> e <titulo ling="pt"> Aprendendo XPath </titulo>

2.5.3. SELENIUM WEBDRIVER

O Selenium WebDriver é uma ferramenta para automação de testes em aplicações web, tipicamente para validar o seu funcionamento [43]. Seu objetivo é prover uma API (*framework*) para explorar as funcionalidades do sistema sendo construído, facilitando e tornando mais simples os testes que devem ser executados. Por exemplo, desde verificar obrigatoriedade de campos até mesmo regras de negócio, preenchendo determinados campos e realizando determinados fluxos de navegação.

Em um *script* de execução do WebDriver, é possível descrever através de linguagens de programação (como Java, PHP, C#, etc.) a ordem dos passos que devem ser executados. Cada passo dos testes pode ter diversas formas de interagir com a tela, como clicar em elementos, preencher campos, verificar a presença de elementos, navegar entre páginas, etc.

Para execução de ações sobre os elementos da tela (contidos no código HTML), é necessário utilizar alguns dentre diversos comandos para obtê-los: através do atributo “id” das etiquetas, ou do atributo “class”, e finalmente, o recurso mais dinâmico, através de comandos XPath. Apesar de ter sido projetado para arquivos XML, o WebDriver utiliza XPath para navegar entre os elementos HTML da mesma forma (sujeito a eventuais problemas da falta de integridade desse formato).

WebDriver é especificamente o nome da interface principal, onde os testes são implementados, porém existem diversas implementações para a mesmo, cada uma simulando navegadores (*browsers*) diferentes. É possível então tornar portáveis os testes e validar se a aplicação se comporta como esperado quando é alterado o navegador utilizado. Durante sua execução é possível visualizar o mesmo que um usuário ao acessar e navegar pela página web, visto que o *framework* abre uma instância do navegador.

A única limitação apresentada pelo WebDriver para o objetivo, descoberta ao longo do processo detalhado na Subseção 3.2, foi que devido ao fato de ser projetado para simular a interação de um usuário do sistema com a página apresentada, ele não permite a obtenção e interação (por exemplo, clicar) com elementos que não estão visíveis (visto que um usuário final não seria capaz de fazer o mesmo). Por ser um projeto *open source* (com código-fonte disponível), foi possível recompilar o *framework* por completo, implementando as modificações necessárias.

2.6. COMÉRCIO ELETRÔNICO

O termo comércio eletrônico (*e-commerce*), teve origem no final dos anos 70, quando era referenciado como o uso de tecnologia para transmissão de documentos comerciais de forma eletrônica. Nesses tempos o uso de cartões de crédito e até serviços de atendimento ao cliente eram considerados formas de *e-commerce*. Desde a popularização da internet, e o desenvolvimento de protocolos de segurança para

transações nela, o termo passou a se referir ao uso de um aparelho eletrônico (computador, celular, tablet) para realizar uma transação comercial.

Com a expansão desse segmento para consumidores, surgiu o espaço para que empresas passassem a disponibilizar produtos e serviços online. Desde essa época, autores já consideravam as empresas que operam exclusivamente na internet, a exemplo da Amazon, eBay, e Yahoo, como pioneiros de uma “nova economia”, capazes de ameaçar a liderança das empresas tradicionais [44]. E esse espaço é concorrido até os dias de hoje por empresas puramente virtuais, com preços competitivos, e por empresas com lojas físicas, que tentam manter a competitividade.

Além do fator de aumento da quantidade de acessos por diversos nichos da sociedade a internet na última década, as empresas passaram a investir nos seus segmentos de *e-commerce* devido às vantagens que o mesmo apresenta, tais como:

- o negócio estar disponível 24h por dia, 7 dias por semana;

- fatores de economia em relação aos empreendimentos físicos (lojas), tais como gasto com empregados, aluguel, organização e manutenção do espaço;

- divulgação mais rápida de promoções, assim se tornando ainda mais competitivas;

- possibilidade de aplicarem descontos sobre os preços praticados em lojas, devido as economias aplicadas nos itens anteriores.

As negociações tipicamente envolvem uma forma de pagamento digital, que caminha junto com a evolução das tecnologias para pagamento na internet. Atualmente as mais populares e utilizadas no Brasil são os cartões de crédito (em torno de 67% optam por esse meio), boleto bancário (em torno de 20%) e a Transferência Eletrônica de Fundos (2%) [16].

As negociações ainda são divididas em dois grandes grupos, as do tipo B2C (*Business-to-Consumer*, negócio-a-consumidor), e o B2B (*Business-to-Business*, negócio-a-negócio) [41]. Como o nome sugere, as do tipo B2C são as transações voltadas para o consumidor final, como a venda de produtos e serviços online. Enquanto as do tipo B2B, são aquelas que determinada empresa realiza para com outras, por exemplo, fornecedores de produtos para vendas, ou de material necessário para suas atividades e operação.

Com os fatores previamente apresentados que se aplicam sobre o *e-commerce*, o crescimento exponencial que vemos na Figura 13 representa o resultado esperado do faturamento resultante de tamanha movimentação de finanças [15]. Com um crescimento

médio de 44,2% em por ano, analisando a última década, podemos afirmar que essa é uma área que deve ser analisada, estudada e trabalhada com atenção.

Figura 13 - Faturamento do *e-commerce* no Brasil, adaptado de eCommerceOrg

2.7. INTELIGÊNCIA COMPETITIVA

A Inteligência Competitiva, às vezes também referenciada como 'Inteligência de Mercado', é um conjunto de diversas ações para identificação de riscos e oportunidades para a atuação no mercado por parte de uma organização. Fleisher e Blenkhorn [23] defendem que a IC é o processo pelo qual as organizações obtêm informações sobre concorrentes e o ambiente competitivo e, idealmente, as aplicam ao seu processo de tomada de decisões e planejamento.

A Associação Brasileira dos Analistas de Inteligência Competitiva (ABRAIC) [1] define ainda a Inteligência Competitiva como um processo informacional proativo que conduz à melhor tomada de decisão, seja ela estratégica ou operacional. É um processo sistemático que visa descobrir as forças que regem os negócios, reduzir o risco e conduzir o tomador de decisão a agir antecipadamente, bem como proteger o conhecimento gerado.

A Society of Competitive Intelligence Professionals (SCIP) [42] complementa ainda as definições anteriores dizendo que a IC efetiva é um processo contínuo envolvendo uma coleção de informações ética e legal, uma análise que não evita conclusões que não são bem-vindas, e a disseminação controlada de inteligência para tomadores de decisão.

Uma vez levantada a definição de IC, pode-se ainda verificar a abrangência da sua aplicação, isso é, a utilização de tais métodos não está limitada apenas a grandes empresas multinacionais ou mesmo a setores específicos de uma determinada empresa. Uma empresa pequena precisa de uma análise das condições do mercado em que está envolvida, tanto quanto as grandes empresas, para suas tomadas de decisões. Assim como não é um técnica limitada a um departamento específico, visto que pode ser aplicada a áreas de marketing, P&D, recursos humanos, gerência de projetos a longo e curto prazo, entre outras.

A IC busca analisar e estruturar dados encontrados em diversos ambientes e em diversos formatos, visando construir assim a base para posteriores consultas (ainda que não necessariamente computacionalmente). Essas bases para consulta construídas idealmente passam pelo processo de integração de informações, que é definido como uma forma de acessar um conjunto de informações heterogêneas de forma unificada.

Para um sistema de IC, ter a integração de informações significa acessar através do mesmo, em uma estrutura única, na forma de um mediador, diversas informações relevantes para o objetivo declarado. Como parte desse desafio, surge a integração semântica, que busca garantir que informações realmente relevantes ao contexto (equivalentes às estruturas / conceitos declarados) serão consideras e absorvidas pelo sistema.

2.8. POSICIONAMENTO DESTE TRABALHO EM RELAÇÃO AO ESTADO DA ARTE

Em relação aos tópicos já abordados nessa seção e relevantes a essa pesquisa, podemos encontrar diversos trabalhos que situam o mesmo no estado da arte.

No que se refere à análise de informações para IC, existem alguns estudos acadêmicos já aplicados à área usando técnicas de PLN, como análise de sentimentos [48], extração de ontologias [31], reconhecimento de entidades nomeadas (REN) [11]. Neste estudo, o foco será em um contexto de dados apresentados em uma estrutura hierárquica, realidade não ainda explorada para IC.

Voltando a analisar a campanha da OAEI detalhada na Subseção 2.3, pode-se perceber que é muito importante para analisar o estado da arte no âmbito de alinhamento, por ser um evento internacional onde as mais recentes soluções são apresentadas e testadas. Muitas delas apresentam abordagens com técnicas recentes e diferentes das convencionais, como as que utilizam programação genética [28], outras que trabalham com *Linked Data* [35] [34], algumas ainda proporcionam comparações de alinhamentos com a possibilidade de fusão (*merge*) de ontologias [33]. Por fim ainda temos técnicas propostas que analisam a similaridade semântica para alinhamentos, estudo pouco abordado especificamente para esse fim.

Outros trabalhos acadêmicos da área para o propósito de analisar similaridades, proposta similar a desse estudo, podem ser destacados, como a aplicação de métricas para avaliar a similaridade de estruturas [7] [13] e similaridade semântica de duas ontologias. Pode-se mencionar para tal a aplicação de métricas para avaliar a similaridade semântica de duas ontologias, como a SiSe [24], e posteriormente o estudo feito sobre a mesma por Brandolt [8], esses especificamente aplicados a ontologias em português.

Esse trabalho surge então um contexto que é motivado pelos problemas da IC, abordando uma estrutura de dados diferente de outros trabalhos (mas com objetivos similares). Enquanto isso se baseia nas práticas, estruturas e metodologias propostas pela OAEI e praticadas nos trabalhos publicados na mesma.

Devido ao contexto de *e-commerce*, algumas técnicas de trabalhos apresentados na Subseção 2.8.1 foram utilizadas para inspirar parte do projeto desenvolvido. Projeto esse que se propõe ainda a ser dinâmico o suficiente para futura utilização de trabalhos como os de similaridade de estruturas e similaridade semântica, e diversos outros encontrados no meio acadêmico.

2.8.1. HIERARQUIAS E CLASSIFICAÇÃO NO COMÉRCIO ELETRÔNICO

Diversos trabalhos e iniciativas de pesquisa realizadas nos últimos anos foram voltados a tópicos relacionados ao *e-commerce* e *e-catalogs*, visando tipicamente a construção de algum modelo padrão de referência, comum a várias aplicações.

Vegetti et al. [51] abordam a necessidade de um modelo de produtos integrado e compartilhado para as organizações utilizarem, porém de uma forma muito mais "industrial", focando na composição dos produtos até o nível de matéria-prima, baseando-se nas listas de materiais das empresas.

Trabalhos iniciais nesse tema também abordaram meios de unificar padrões da indústria para classificação de produtos [26] [39], como os mais populares UNSPSC [50], eCl@ss [14] e RosettaNet Technical Dictionary [40]. Essas abordagens são interessantes pelos métodos utilizados para combinação das diferentes estruturas, mas não são muito práticas para os usuários finais, pois possuem extensos níveis de hierarquia extremamente detalhados e com uma categorização muito mais próxima da linguagem da indústria.

Algumas das pesquisas sobre o tema são realizadas buscando ainda métodos de recomendação de produtos otimizados [21] [6]. Outros trabalhos acabam por atingir objetivos mais específicos na interação com produtos, como comunicação entre agentes para negociação [25] [32]. Lógicas essas que encontramos em *sites* de vendas e negociação *online*, e extremamente valorizadas pelo mercado corporativo.

Lee e Shim [30] propõem um modelo formal de ontologia para modelagem de produtos, incluindo seus respectivos esquemas de classificação e atributos, assim como uma grande variedade de relacionamentos.

3. FRAMEWORK PLATAL

Aqui será abordado o desenvolvimento do *framework* PLATAL. Será visto a arquitetura da solução em nível de módulos para o usuário assim como o uso de tecnologias para compor o *software* final.

3.1. ARQUITETURA DA SOLUÇÃO

A aplicação disponibilizada é composta por quarto módulos, cada um agregando diferentes funcionalidades relacionadas a atividades de alinhamento de hierarquias / ontologias. Por ser uma aplicação escrita em tecnologia Java, se fez uso de diversos outros *frameworks* disponíveis, detalhados no Capítulo 2. Na Figura 14 **– Arquitetura de software da solução** é exibida a relação de cada módulo com as tecnologias e customizações usadas.

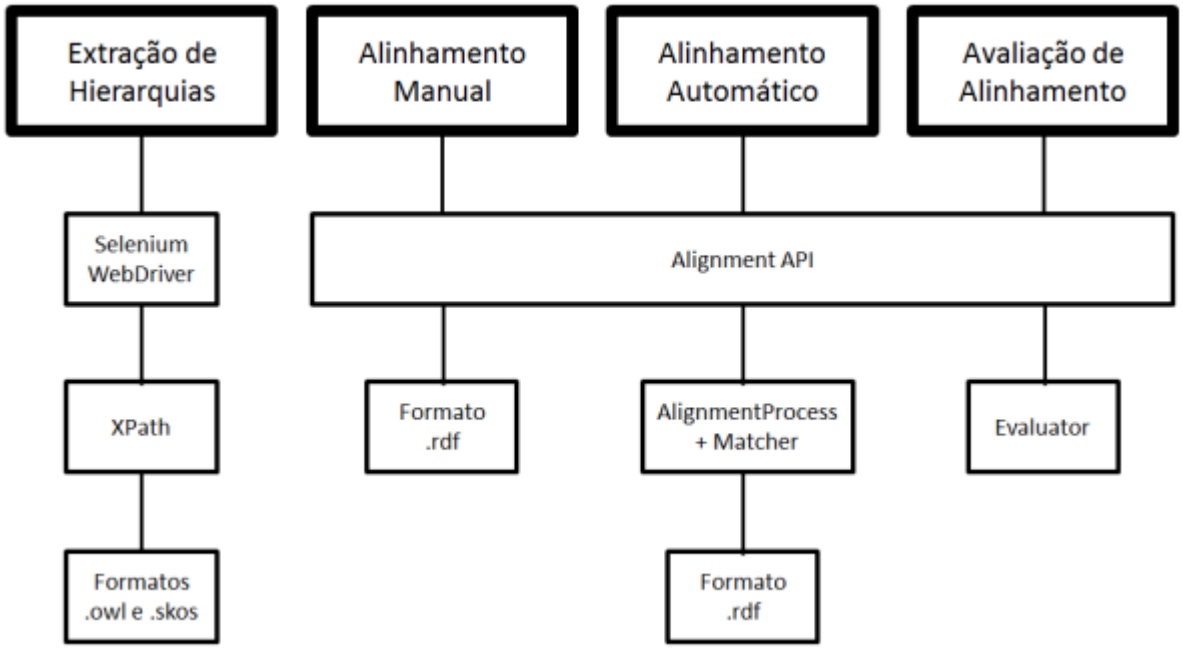

Figura 14 – Arquitetura de software da solução

Percebe-se então que o módulo de extração de hierarquias possui uma dependência um pouco diferente, pois utiliza o Selenium WebDriver e expressões XPath para navegar e selecionar trechos de páginas, para por fim exportar para formatos SKOS e OWL. Os demais módulos estendem funcionalidades disponibilizadas principalmente na Aligment API, customizando componentes para servir as necessidades desse trabalho. Isso ocorre através da customização de *matchers* para torna-los flexíveis e avaliadores para exibir os resultados desejados.

Todos os módulos da solução desenvolvida possuem também como grande diferencial apresentarem interface gráfica para as ações. Isto é, a entrada dos dados e até mesmo a leitura de resultados pode ser feita completamente em janelas, possibilitando um usuário operar as funções sem programação caso deseje.

3.2. EXTRAÇÃO DE HIERARQUIAS

A primeira etapa neste trabalho é a extração das hierarquias exibidas em páginas da internet, para sua posterior análise, de uma forma similar à ação de um *crawler*. Após a captura das hierarquias, é possível realizar a exportação para os formatos padrões para visualizar essas informações.

Como parte da ferramenta disponibilizada nesse trabalho, foi desenvolvida a interface do módulo que vemos na Figura 15 para realizar as operações necessárias ao longo dessa etapa. Os passos para a implementação desse módulo são descritos de acordo com o fluxo que vemos na Figura 16.

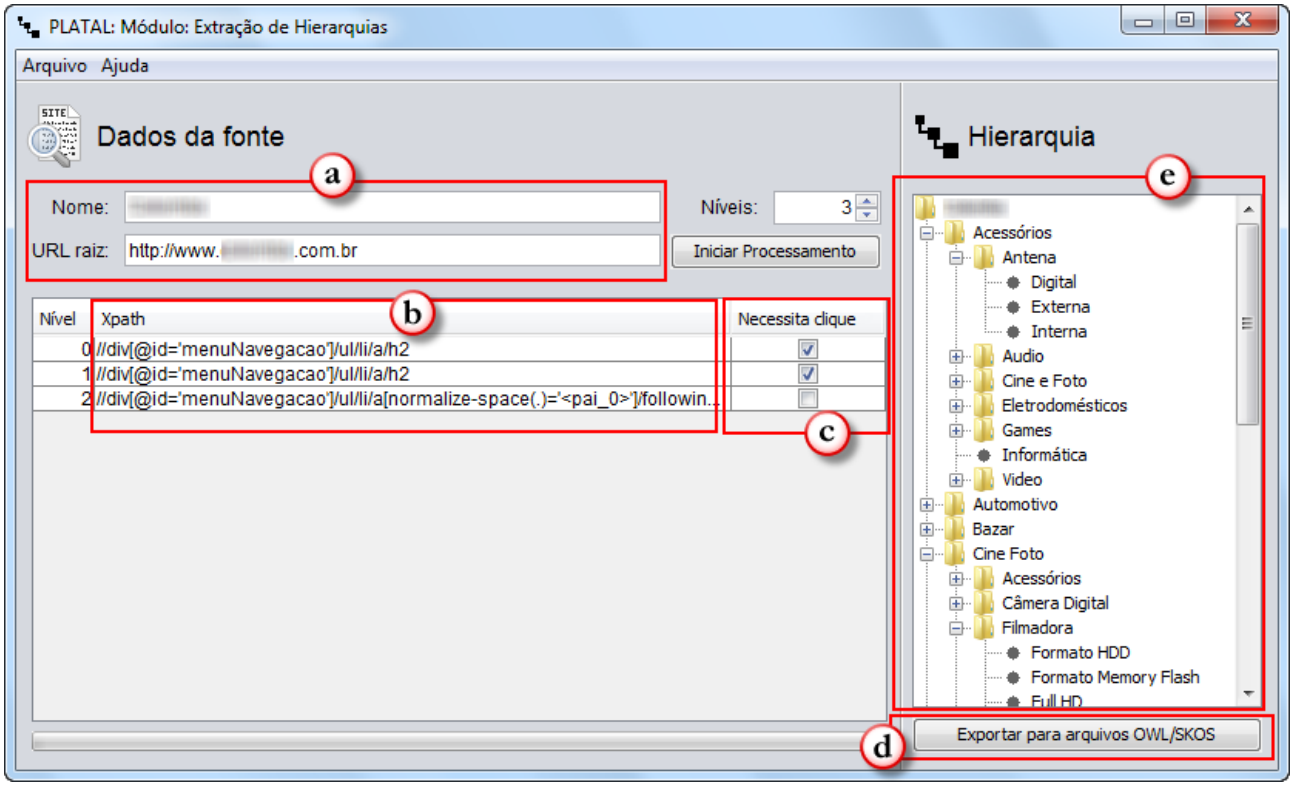

Figura 15 - Interface do módulo para extrair hierarquias

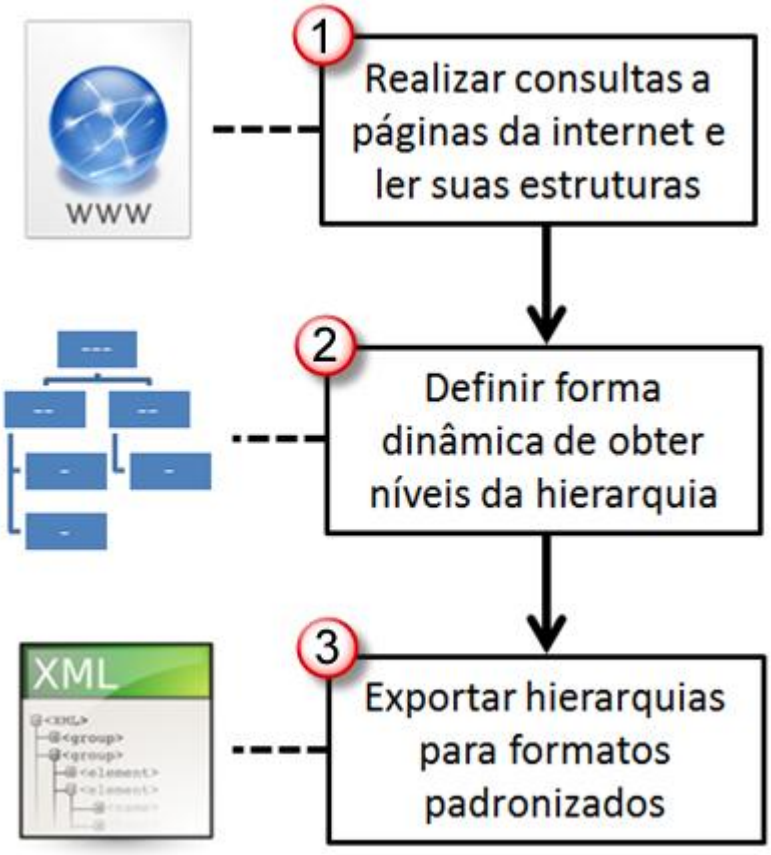

Figura 16 - Passos para a extração de hierarquias

3.2.1. PASSO 1: CONSULTAR PÁGINAS DA INTERNET E OBTER ESTRUTURAS

A ferramenta construída possibilita manipular partes de um documento HTML, para ler os elementos contidos em sua estrutura, e também permite navegar entre diferentes páginas para buscar o restante de uma hierarquia (nem sempre presente em uma única página), conforme ilustrado na Figura 17.

Figura 17 - Exemplo de hierarquia disposta em mais de uma página web

Foram analisadas algumas ferramentas / *frameworks* e tecnologias para acesso e manipulação de conteúdo de páginas da internet, muitas dessas APIs e projetos lidavam com arquivos HTML de forma bem otimizada, porém poucos proporcionavam o controle de navegação que era necessário para a organização comumente encontrada nas páginas estudadas. Por exemplo, era necessário obter a URL de um atalho da página e direcionar o navegador para ela, porém alguns desses endereços se encontram em sua forma relativa (partem do endereço já presente na página do navegador). Piores cenários

incluíam casos em que os redirecionamentos a partir de interação do usuário com a tela eram realizados através de código JavaScript, que necessitavam de um navegador para os interpretar, o que onerava a tarefa de navegação a um ponto de torná-la praticamente inviável.

Através do estudo de ferramentas de automação de navegação em páginas, foi identificado o Selenium WebDriver, que conforme foi detalhado na Subseção 2.5.3, é originalmente um projeto para testes automatizados. Ele, contudo, atende às necessidades deste trabalho por ser capaz de simular um navegador por completo, e realizar operações sobre a página como se um usuário humano as estivesse fazendo. Por apresentar essas características, o WebDriver foi adotado como ferramenta de obtenção das páginas HTML.

Foi construída então uma forma de utilizar o WebDriver para abrir páginas web, obter elementos presentes na página, interagir com esses elementos, e realizar navegação para qualquer endereço presente na tela. Na Figura 18 vemos um trecho de um arquivo HTML acessado através da ferramenta. A partir da etiqueta “div”, vemos a montagem de parte da hierarquia exibida para um usuário, onde cada etiqueta “h2” dentro do caminho de etiquetas “div / ul / li” contém o nível pai; enquanto as etiquetas “a”, contidas dentro do caminho “ul / li” irmão ao elemento “h2” da pai, representam os filhos da mesma.

```
<div class="menucat-categorias">
    <span class="bordaInicio"></span>
  <ul>
    <li class="linha-produto aberto">
        <h2> Notebooks e Netbooks </h2>
      <ul>
        <li class="aberto">
            <a href="notebooks-e-netbooks/notebooks"> Notebooks </a>
        </li>
        <li class="aberto">
            <a href="notebooks-e-netbooks/netbooks"> Netbooks </a>
        </li>
      </ul>
    </li>
    <li class="linha-produto aberto">
        <h2> Computadores </h2>
      <ul>
        <li class="aberto">
            <a href="computadores/computadores-com-monitor"> Computadores com Monitor </a>
        </li>
        <li class="aberto">
            <a href="computadores/computadores-13099"> Computadores sem Monitor </a>
        </li>
      </ul>
    </li>
```

Figura 18 - Trecho HTML de hierarquia em página web

Para realizar tais operações de navegação é necessário determinar o endereço inicial (onde os nodos raiz da hierarquia se encontram), e um nome único para essa hierarquia ser referenciada, conforme vemos na seção “a” da Figura 15.

3.2.2. PASSO 2: OBTER NÍVEIS DE HIERARQUIA DINAMICAMENTE

Uma vez atingida a meta de obter os arquivos necessários para investigar as hierarquias contidas nas páginas, se torna necessária a definição de uma notação comum que possa ser informada para a extração da estrutura de tais arquivos. A notação precisa ser capaz de obter qualquer estrutura hierárquica de elementos HTML, de forma parametrizável para cada *e-catalog*. Com isso, em cada estrutura de página pode-se flexibilizar a forma de seleção dos nodos da hierarquia de acordo com sua estrutura de apresentação, independente do formato.

Apoiada pela ferramenta escolhida para acessar as páginas no passo 1, foram adotadas as expressões XPath, que permitem expressar uma forma dinâmica de obter os diferentes níveis da hierarquia definida. Na Figura 19 vemos como a execução de uma expressão XPath sobre o código HTML apresentado na Figura 18 pode selecionar determinado conjunto de elementos que representam um nível da hierarquia. Em posse dessa seleção de nodos, o texto puro contido em cada um é extraído como o nome de entidades do nível em análise.

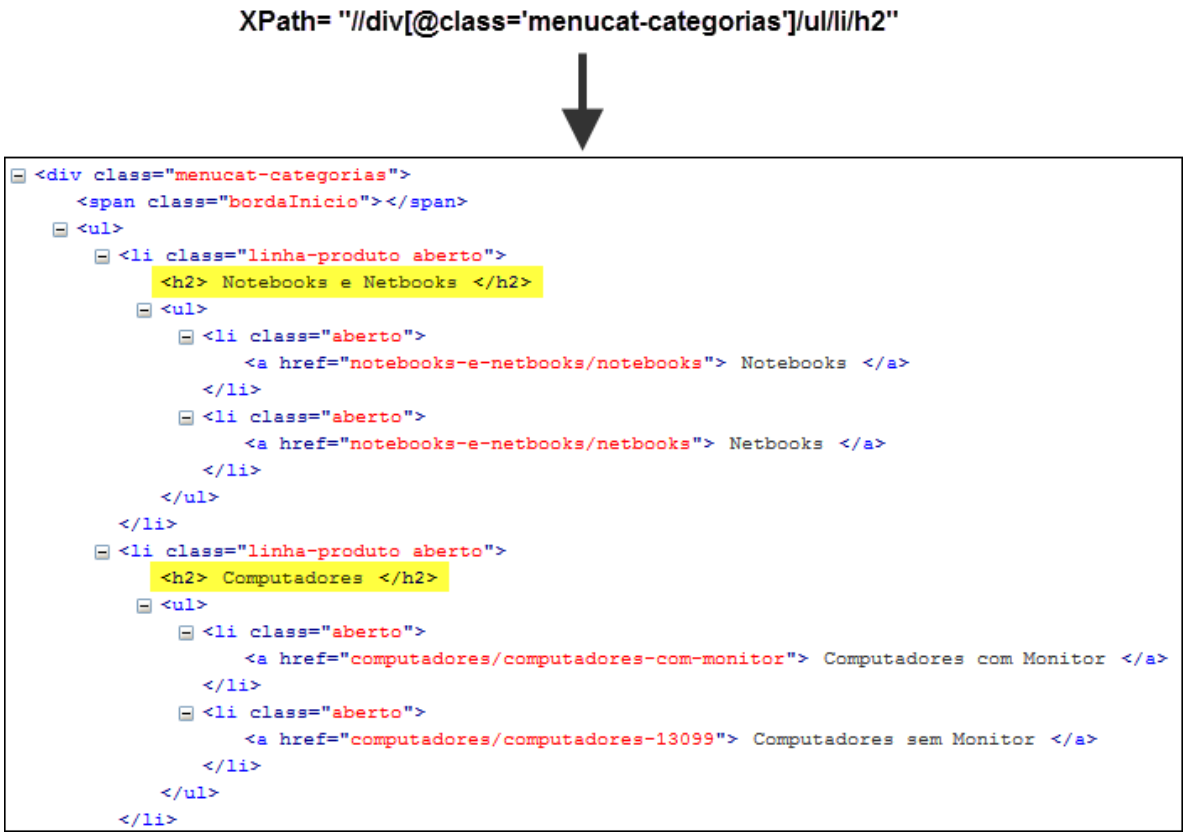

Figura 19 - Expressão XPath selecionando elementos HTML

Com a capacidade de selecionar coleções de elementos com apenas uma expressão, foi necessário definir uma estrutura que permita uma série de expressões, uma para cada nível da hierarquia, conforme vemos na seção “b” da Figura 15. Nela, de acordo com o número de níveis informado, serão exibidas linhas que representam cada nível da hierarquia a ser extraída, de forma que ao buscar pelas categorias do nível correspondente em tempo de execução, o programa executará a expressão ali informada para obter os elementos.

Resta agora detalhar a forma de relacionar uma série de elementos selecionados do HTML com a entidade “pai” (nível da hierarquias imediatamente superior). Na Figura 20 vemos um exemplo da solução elaborada, através de especificação da entidade pai no próprio XPath. Foi criada como parte da solução as variáveis “<pai_**n**>”, onde *n* representa qualquer nível de entidade pai superior (começando com o valor 0 para a entidade pai imediata, 1 para a entidade pai da pai, etc). Com isso se tornou possível modificar em tempo de execução a expressão de cada conjunto de entidades filhas de acordo com a entidade pai. Desta forma, uma expressão para um determinado nível, ao

retornar todos os elementos do mesmo, pode filtrar apenas os que se referem à entidade pai corrente.

- XPath usando variável para nodo pai:
"//div[@class='menucat-categorias']/ul/li/h2[text()='**<pai_0>**']/following-sibling::ul/li/a"

- XPath após interpretação:
"//div[@class='menucat-categorias']/ul/li/h2[text()='**Computadores**']/following-sibling::ul/li/a"

```
<div class="menucat-categorias">
    <span class="bordaInicio"></span>
  <ul>
    <li class="linha-produto aberto">
        <h2> Notebooks e Netbooks </h2>
      <ul>
        <li class="aberto">
            <a href="notebooks-e-netbooks/notebooks"> Notebooks </a>
        </li>
        <li class="aberto">
            <a href="notebooks-e-netbooks/netbooks"> Netbooks </a>
        </li>
      </ul>
    </li>
    <li class="linha-produto aberto">
        <h2> Computadores </h2>
      <ul>
        <li class="aberto">
            <a href="computadores/computadores-com-monitor"> Computadores com Monitor </a>
        </li>
        <li class="aberto">
            <a href="computadores/computadores-13099"> Computadores sem Monitor </a>
        </li>
      </ul>
    </li>
```

Figura 20 - Uso de expressões XPath para selecionar nodos filhos

Após a definição da forma de associar nodos filhos a pais em diferentes níveis, o foco foi direcionado para resolver o problema relativo à disposição das hierarquias em diferentes páginas. É necessário que para capturar uma hierarquia por completo, sejam realizadas navegações entre diferentes páginas, de forma que seja possível dizer quais níveis estão em diferentes páginas. Tirou-se proveito da estrutura do WebDriver, possibilitando clicar em elementos para fazer essa navegação. Chegando à solução final que vemos na seção “c” da Figura 15, que na coluna “Necessita clique” indica se o elemento capturado naquele nível da hierarquia deve ser clicado a fim de acessar sua sub-hierarquia (estrutura de menu / sub-menu).

Por exemplo, se temos uma parte da estrutura numa página HTML, e outro numa segunda página, dizemos que o nível da primeira página necessita de clique para acessar os subníveis. Com isso, o *framework* utilizado se encarrega de realizar um clique sobre o

elemento capturado no momento de leitura, que independente da etiqueta HTML, se comportará como se o usuário realizasse o clique e aguardasse a página seguinte carregar.

3.2.3. PASSO 3: EXPORTAR HIERARQUIA PARA FORMATO PADRONIZADO

Com os passos previamente descritos, o usuário pode executar a extração da hierarquia, através da seção “d” destacada na Figura 15, e visualizar o resultado na seção “e” da mesma figura.

Finalmente, com a estrutura hierárquica adquirida, resta a operação de salvar a mesma em um formato padronizado, que sirva tanto para a próxima etapa desse trabalho quanto para reuso em demais aplicações. Durante o estudo de técnicas que realizem o alinhamento, foi percebido que a maior parte trabalha a partir de ontologias, e por consequência a linguagem OWL.

Esse é, portanto, o primeiro formato possível para exportação disponibilizado pela aplicação construída. Esse formato serve de entrada para as técnicas que veremos mais adiante no estudo. O diferencial desse estudo, porém, é que o mesmo se baseia em estruturas de *websites* sem uma organização semântica minimamente rígida, ou seja, não possuem características de uma ontologia. Para casos como o que encontramos na análise de *e-catalogs*, o formato OWL não pode ser aplicado formalmente para armazenar a estrutura.

Dois são os principais fatores que acarretam nessa conclusão, primeiro o de que as estruturas de classificação dos *websites* não possuem necessariamente a relação “é um” de suas entidades filhas para as entidades pais. Como vemos na Figura 21, a entidade “Limpeza” não é um “Automotivo”. Na mesma ilustração observamos o segundo fator mencionado, que é o caso de duas entidades com o mesmo nome nesse domínio representarem conceitos diferentes: “Acessórios” abaixo da categoria “Automotivos” não pode ser agrupado ou alinhado como equivalente da categoria de mesmo nome abaixo da categoria “Equipamentos para DJ”.

Figura 21 - Hierarquia que não respeita as definições de uma ontologia

Com esse novo cenário, se estudaram formas alternativas e corretamente formais (visto que OWL não seria o correto) de armazenar estruturas, e chegou-se à conclusão de que a que mais se adaptava às necessidades e ao formato em que se encontravam os dados seria a do SKOS, voltado para classificação de esquemas. Concluindo assim os formatos disponíveis e interessantes para esse trabalho.

Durante a exportação, alguns processamentos são feitos em cima dos nomes das entidades capturadas. Por exemplo, são tratados caracteres que são reservados para o formato de saída, assim como os espaços em branco são substituídos pelo caractere *underline* (“_”). Ainda para as entidades que possuem um nome duplicado na mesma hierarquia, foi definido que para cada instância encontrada com o nome seria utilizado o padrão “Nome_Entidade[**n**]”, como observado na Figura 22, onde *n* representa um número único sequencial por entidade para identificar aquela aparição do nome na estrutura. Sendo necessário tratar o nome da entidade apropriadamente para correta comparação durante a Etapa 2 também.

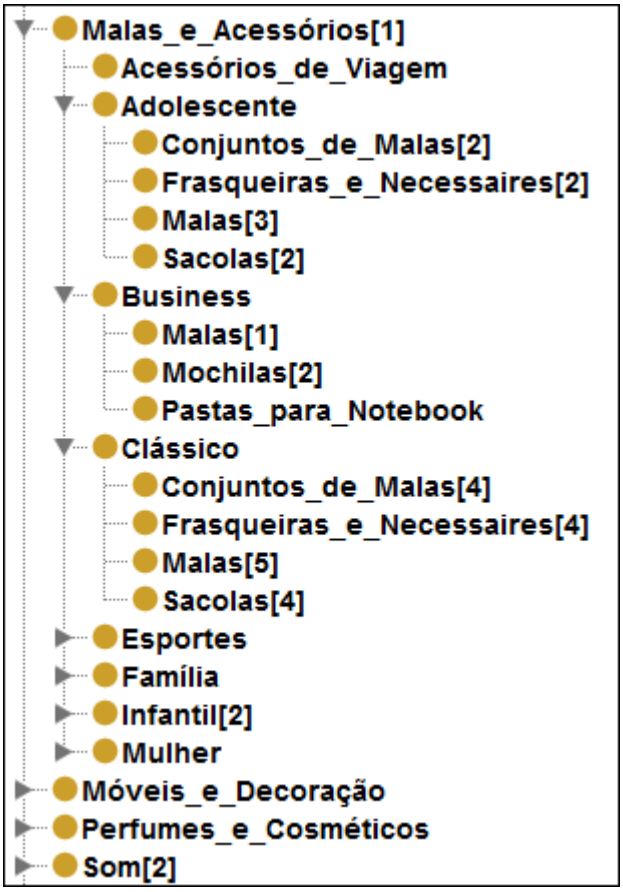

Figura 22 - Trecho de hierarquia com categorias duplicadas

Com essa ultima operação, se conclui essa primeira etapa do estudo e da ferramenta construída. Está disponível para o usuário final a possibilidade de acessar uma hierarquia em um *site*, descrever uma forma dinâmica para obtê-la em seus diversos níveis e exportar os dados lidos para os formatos padronizados pela W3C.

3.3. ALINHAMENTO DE HIERARQUIAS

Nessa segunda fase, será utilizado como entrada as hierarquias obtidas durante a etapa anterior. Elas serão processadas ao longo de uma série de heurísticas com o intuito de analisar a equivalência entre as entidades de duas hierarquias diferentes. Por fim, os resultados serão exportados para um formato de alinhamento padronizado. A Figura 23 ilustra a arquitetura desse módulo, que possibilita a configuração dos métodos usados para o alinhamento.

Figura 23 - Arquitetura da solução para alinhamento de hierarquias

3.3.1. ENTRADA DE DADOS: HIERARQUIAS

As ferramentas disponíveis e métodos conhecidos para alinhamento de entidades em sua maioria utilizam ontologias como entrada, inclusive a Alignment API, *framework* usado como base para realizar o alinhamento nesse projeto. Como entrada para essa etapa serão informadas duas hierarquias em formato OWL, que não necessariamente representarão ontologias.

Devido à utilização do formato OWL como entrada, foi apenas descartada a dedução da relação de subtipo de uma entidade com sua classe pai, sendo que as hierarquias representadas podem não respeitar às regras formais de uma ontologia. Métodos de alinhamento que considerem esse tipo de relação – 'é um' – também não puderam ser levados em conta no desenvolvimento dessa etapa.

3.3.2. PROCESSOS DE ALINHAMENTO

Para o processamento das hierarquias visando seu alinhamento, foi utilizado o conceito de "Processo", que aqui se refere à sequência de ações que resulta na sugestão dos alinhamentos mais relevantes.

Ao se entrar com as hierarquias no módulo de alinhamento automático (Figura 24), todos os processos disponíveis são levantados e aplicados sobre cada par de entidades possíveis de se alinhar entre as hierarquias, a fim de encontrar o processo que retorna os melhores resultados.

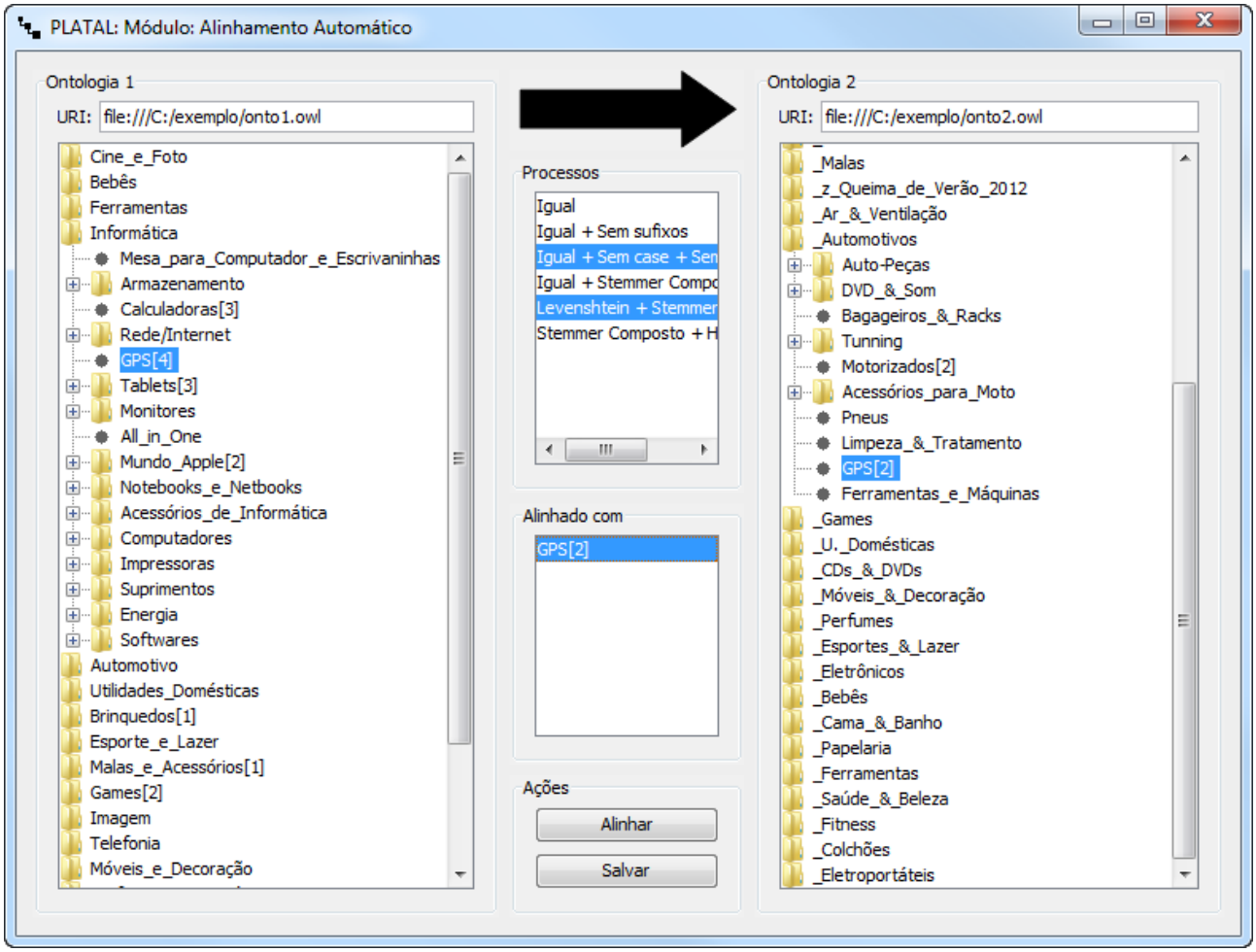

Figura 24 - Interface do módulo de alinhamento automático

Um processo é composto de um conjunto de filtros, que serão detalhados na Subseção 3.3.3, e um conjunto de avaliadores, que por sua vez serão detalhados na Subseção 3.3.4. Um processo começa por aplicar cada filtro registrado para ele sobre as duas entidades que se deseja analisar, visando aproximar o texto de ambas. Os textos resultantes desse filtro são submetidos a todos os avaliadores registrados para o processo em questão, e esses elencam (no encontrou maior similaridade) o quão parecidas as duas entidades são.

Ao final dessa execução, se obtém como resultado um conjunto de entidades mapeadas entre as duas diferentes hierarquias, junto com o grau de confiança na relação encontrada. O módulo de alinhamento automático determina então qual processo dentre

os aplicados encontrou maior similaridade entre as entidades (caso alguma seja encontrada) e determina para a saída do programa esse valor como o final.

3.3.3. FILTROS

O conceito de “Filtro” aqui é determinado como a entrada de um valor textual para processamento (o nome da entidade sob análise) e a produção como saída de um novo valor texto, após a aplicação de alguma heurística. Um filtro pode ser definido como uma heurística, ou uma heurística combinada com outro filtro, visando processar o resultado já obtido de outro filtro. Em uma forma similar à definição de gramáticas livres de contexto, podemos descrevê-los da seguinte forma:

FILTRO = HEURÍSTICA
FILTRO = FILTRO + HEURÍSTICA

Como exemplo de alguns filtros que aplicam heurísticas simples, podem-se mencionar filtros que:

- convertem todo o texto para letras minúsculas;
- removem o sufixo de repetição do nome (criado na Etapa 1);
- removem acentuação;
- substituem caracteres com significado (por exemplo, “&” por “e”);
- removem *stop-words*;
- aplicam técnicas de *stemming*.

Esses filtros “básicos” podem então ser combinados para encontrar resultados ainda mais similares. Por exemplo, convertendo o texto para minúsculo e removendo a acentuação; melhorando as chances de encontrar uma equivalência.

3.3.4. AVALIADORES

Um avaliador dentro de um processo é uma estrutura capaz de receber duas entidades pós-processadas por um filtro e analisar o quão similares as mesmas são,

tipicamente através da análise dos nomes. Da mesma forma que filtros, avaliadores podem ser compostos a fim de reutilizar uma avaliação prévia como entrada para sua lógica.

Como exemplo de avaliadores, pode-se mencionar alguns filtros que aplicam heurísticas simples, como:

- analisar se os textos são exatamente iguais;

- aplicar algoritmos de *edit-distance* (como a distância de Levenstein) e determinar um ponto de corte (percentual por exemplo) para similaridade;

Avaliadores mais complexos incluem técnicas de alinhamento em nível de estrutura, muito convenientes à realidade abordada pelo trabalho (hierarquias). Um exemplo simples é o de um avaliador que analisa as superclasses e subclasses das entidades e calcula o percentual de classes semelhantes entre as estruturas. Esse avaliador idealmente deve ser antes combinado com um que atribua uma igualdade entre as entidades em si (pelo texto), constituindo um avaliador composto de dois avaliadores.

3.3.5. SAÍDA DE DADOS: ALINHAMENTO

Após a execução de todos os processos e a determinação do alinhamento completo das hierarquias, obtém-se um alinhamento como saída da execução. Esse alinhamento é então exportado para o formato XML / RDF suportado pela Alignment API, descrito na Subseção 2.5.1. O formato é então preenchido com a relação de igualdade, deduzido dos processos aplicados, possibilitando seu posterior reuso por outras aplicações que utilizem o mesmo formato.

4. EXPERIMENTOS E AVALIAÇÃO

As funcionalidades da ferramenta PLATAL foram testadas e avaliadas no domínio de comércio eletrônico. As hierarquias analisadas foram extraídas de *e-catalogs*. Foram extraídas 14 estruturas de *websites* de *e-commerce* do Brasil, das quais 8 foram utilizadas para estudo e desenvolvimento das heurísitcas; enquanto 6 foram separadas para a avaliação do método proposto. Na Tabela 2 **– Tamanho de estruturas para experimentos e avaliação** podemos ver a complexidade das estruturas, detalhas em número de entidades (nodos) e níveis existentes em cada.

Tabela 2 – Tamanho de estruturas para experimentos e avaliação

Estrutura	Número de nodos	Número de níveis
1	775	3
2	890	3
3	1150	3
4	1079	3
5	613	3
6	580	3
7	1060	3
8	2436	4
9	1380	4
10	2882	3
11	453	3
12	1095	3
13	1074	3
14	1835	3

Juízes humanos realizaram a elaboração de alinhamentos de referência. Para essa atividade lhes foi provido o módulo de alinhamento manual, detalhado na Subseção 4.1.

Para os conjuntos de hierarquias supracitados para avaliação foram realizadas, em um primeiro momento, comparações entre o alinhamento processado pela aplicação e os alinhamentos de referência elaborados por humanos. Outros *matchers* foram avaliados

em relação às mesmas estruturas, visando comparar os resultados deste trabalho com outras propostas da literatura.

Em todas as avaliações realizadas, foram analisadas as medidas de precisão, abrangência e medida-F, resultantes da comparação dos alinhamentos. Para cálculo das métricas, é necessário a análise dos seguintes termos:

- Verdadeiro positivo: alinhamentos encontrados que estão corretos.

- Falso positivo: alinhamentos encontrados que estão incorretos.

- Verdadeiro negativo: alinhamentos determinados como não existentes corretamente (não se aplica ao trabalho, pois ele não busca mapear relacionamentos que não existem).

- Falso negativo: alinhamentos determinados como não existentes incorretamente (não se aplica ao trabalho, pois ele não busca mapear relacionamentos que não existem).

Na área de recuperação da informação (RI), a precisão é a medida do conjunto de dados recuperados que são relevantes à busca em questão, calculada através da seguinte fórmula:

$$\textbf{precisão} = \frac{\text{verdadeiros\_positivos}}{\text{verdadeiros\_positivos} + \text{falsos\_positivos}} \quad \textbf{OU} \quad \textbf{precisão} = \frac{\text{quantidade\_dados\_recuperados\_corretos}}{\text{quantidade\_total\_dados\_recuperados}}$$

A precisão vem então a ser o percentual de alinhamentos classificados corretamente como positivos, dentre todos os que foram classificados como positivos. Portanto, pode-se dizer que a precisão é uma medida de qualidade sobre os dados sugeridos como positivos. Por exemplo, se foram sugeridos 70 alinhamentos, dos quais somente 45 estavam corretos, a precisão será 45 / 70 ≈ 0,64 = 64%.

Enquanto isso a abrangência para a RI vem a ser a medida do conjunto de dados relevantes para a busca em questão que conseguiram ser recuperados, calculada através da seguinte fórmula:

$$\textbf{abrangência} = \frac{\text{verdadeiros\_positivos}}{\text{verdadeiros\_positivos} + \text{falsos\_negativos}} \quad \textbf{OU} \quad \textbf{abrangência} = \frac{\text{quantidade\_dados\_recuperados\_corretos}}{\text{quantidade\_total\_dados\_corretos}}$$

A abrangência para esse trabalho vem a ser o percentual correto de alinhamentos descobertos dentre o total de alinhamentos corretos existentes, podendo ser classificada como uma medida de quantidade sobre os dados sugeridos como positivos. Por exemplo,

se existem 90 alinhamentos para serem descobertos, dos quais foram sugeridos somente 45 alinhamentos corretos, a abrangência será 45 / 90 = 0,50 = 50%.

Por fim, a medida-F é uma avaliação da corretude (exatidão) de um teste, levando em consideração tanto as medidas de precisão quanto abrangência. A medida-F pode ser interpretada então como a média ponderada da precisão e abrangência. Seu cálculo é dado pela seguinte fórmula:

$$\textbf{medida-F} = 2 \times \frac{\text{precisão x abrangência}}{\text{precisão} + \text{abrangência}}$$

Para auxiliar na obtenção das medidas para avaliação, foi desenvolvido um novo módulo para a aplicação PLATAL, voltado para análise de alinhamentos. Podemos ver a interface do módulo referido na Figura 25. Como entrada para ele são passados dois alinhamentos, o que se deseja analisar e o de referência, para por fim obter os valores calculados automaticamente.

Figura 25 – Interface do módulo de avaliação

4.1. ALINHAMENTO MANUAL

Com o propósito de construir manualmente alinhamentos de referência para as avaliações desse trabalho, devido à ausência de uma ferramenta visual, de simples uso e sem custo, foi desenvolvido o módulo de alinhamento manual. A interface provida para tal pode ser vista na Figura 26.

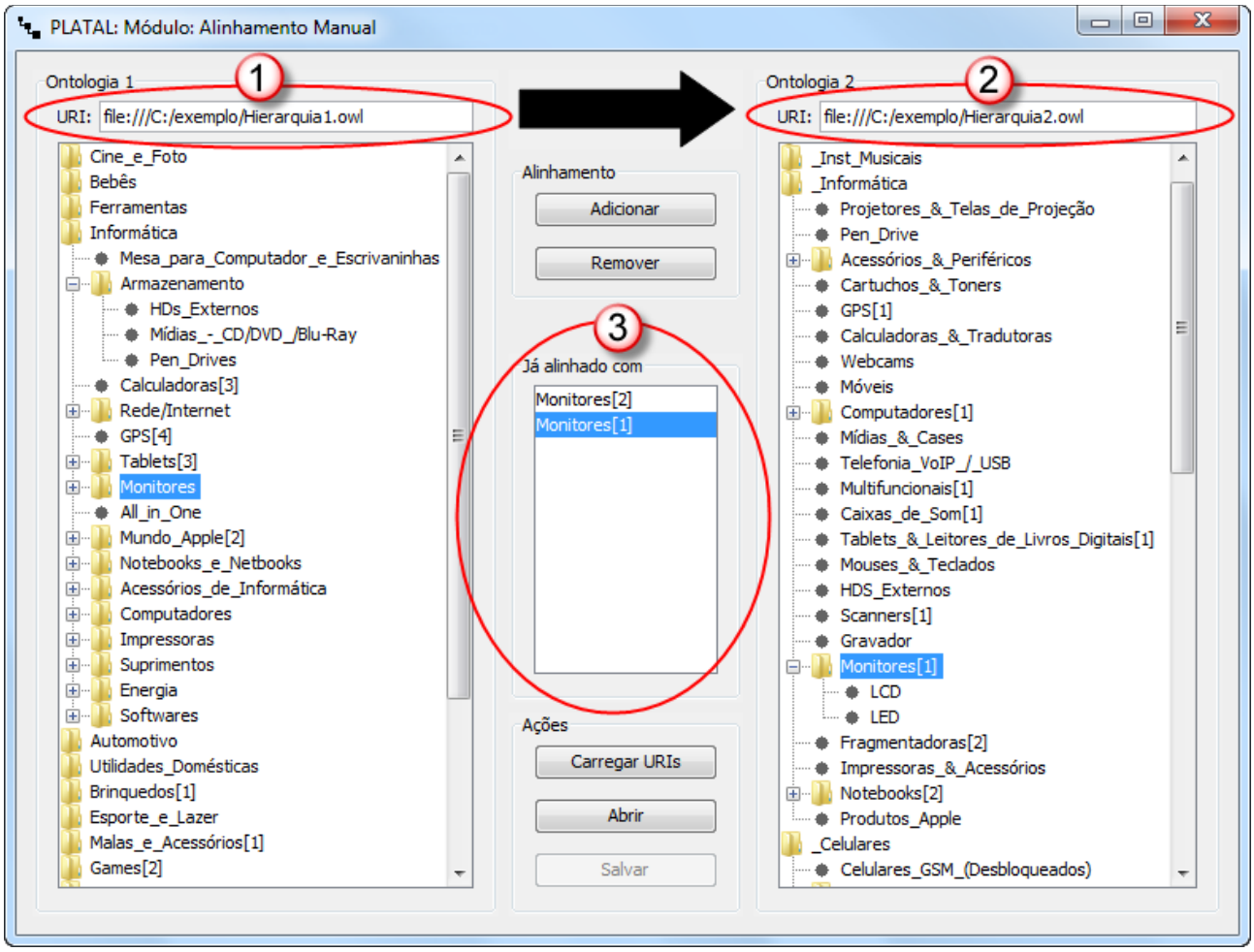

Figura 26 - Interface do módulo de alinhamento manual

Como entrada para início da atividade, nas seções 1 e 2 da Figura 26 temos os campos que comportam as URIs das duas hierarquias (OWL) a serem alinhadas. Através da ação “Carregar URIs” são trazidas para as respectivas árvores abaixo das seções 1 e 2. A partir disso, é possível selecionar um nodo da árvore correspondente à hierarquia 1, e selecionar 1 ou mais nodos da hierarquia 2.

Havendo pelo menos um nodo selecionado em cada árvore, é possível com o botão “Adicionar”, gravar em memória os nodos selecionados como alinhados. Com isso

os nodos da árvore da ontologia 2 alinhados passam ser exibidos na caixa “Já alinhado com” (seção 3), permitindo clicar nos mesmos para vê-los na árvore da ontologia 2, assim como remover o alinhamento com a entidade selecionada (na caixa “Ações”).

Por fim, completam as ações possíveis os botões de “Abrir”, que permite em vez de carregar ontologias a partir de URIs e começar seu alinhamento do início, carregar um alinhamento já existente (formato padrão da Alignment API), assim como as ontologias do mesmo, para edição. E por fim o botão de salvar permite exportar, para o mesmo formato de alinhamento utilizado para abrir arquivos, o alinhamento até então em memória.

4.2. EXPERIMENTOS

Conforme descrito no Subcapítulo 3.3, a organização da solução aqui proposta se baseia na aplicação de um conjunto de processos para identificação de alinhamento entre duas entidades. Filtros, avaliadores e processos testados são descritos na Tabela 3, na Tabela 4 e na Tabela 5 respectivamente.

Tabela 3 – Descrição de filtros desenvolvidos

Nome do filtro	Descrição
REMOVER_SUFIXO_REPETICAO	Remove o sufixo criado pelo módulo de extrair hierarquias para nomes repetidos, ou o contador de quantidade de produtos usado por alguns *websites*. Por exemplo: “Acessórios[9]” = “Acessórios”
REMOVER_ESPACOS_EXCEDENTES	Remove os espaços em branco duplicados ou no início / fim do texto.
CONVERTER_MINUSCULAS	Converte todo o texto para letras minúsculas.
SUBSTITUIR_UNDERLINE	Troca o caractere “_”, utilizado no momento de salvar nomes de entidade como URIs, por espaço em branco.
REMOVER_STOP_WORDS	Remove *stop-words* do texto.

SUSTITUIR_CARACTERE_SIGNIFICADO	Substitui caracteres especiais com significado por seu equivalente em texto puro. Por exemplo: “Cine & Foto” = “Cine e Foto”
REMOVER_ACENTUACAO	Remove caracteres específicos de línguas como o português.
STEMMER	Aplaca uma técnica de *stemming* em cada palavra do texto submetido.
STEMMER_COMPOSTO	Aplica os filtros SUBSTITUIR_UNDERLINE, REMOVER_SUFIXO_REPETICAO SUSTITUIR_CARACTERE_SIGNIFICADO, REMOVER_ACENTUACAO, CONVERTER_MINUSCULAS, STEMMER; nessa ordem.

Tabela 4 - Descrição de avaliadores desenvolvidos

Nome do avaliador	**Descrição**
IGUALDADE	Verifica se os nomes processados das entidades são exatamente idênticos.
COMUTATIVIDADE	Verifica se o nome processado é uma lista de nomes, e os avalia independente da ordem que são declarados. Por exemplo: “Esporte e Lazer” = “Lazer e Esporte”
DISTANCIA_LEVENSTEIN	Calcula a distância Levenstein entre os dois nomes das entidades.
HIERARQUIA	Avaliar o conjunto de categorias superiores e inferiores das entidades, buscando um percentual mínimo de similaridade entre eles.
LEVENSTEIN_HIERARQUIA	Aplica primeiro o avaliador da distância Levenstein. Se o resultado sugerir um possível alinhamento, executa o avaliador da hierarquia para somente assim afirmar positivamente a igualdade.

COMUTATIVIDADE_HIERARQUIA	Aplica primeiro o avaliador de comutatividade. Se o resultado sugerir um possível alinhamento, executa o avaliador da hierarquia para somente assim afirmar positivamente a igualdade.

Tabela 5 - Descrição de processos desenvolvidos

Nome do processo	Descrição
IGUAL	Não realiza nenhuma operação de filtros, e utiliza o avaliador IGUALDADE. Destinado a verificar o resultado mais básico possível, sem processamento.
REMOVE_SUFIXOS	Aplica o filtro REMOVER_SUFIXO_REPETICAO e utiliza o avaliador IGUALDADE. Permite verificar como seria o alinhamento sem a referência de repetição criada.
REMOVE_SUFIXOS_CASE	Aplica os filtros REMOVER_SUFIXO_REPETICAO e CONVERTER_MINUSCULAS, e utiliza o avaliador IGUALDADE. Tem como objetivo avaliar as entidades com seus nomes como seriam lidos diretamente, sem tratamentos avançados.
STEMMER_SIMPLES	Aplica o filtro STEMMER, e utiliza o avaliador IGUALDADE.
STEMMER_COMPOSTO	Aplica o filtro STEMMER_COMPOSTO, e utiliza o avaliador IGUALDADE.
STEMMER_COMUTATIVIDADE	Aplica o filtro STEMMER_COMPOSTO, e utiliza o avaliador CUMUTATIVIDADE.
STEMMER_LEVENSTEIN	Aplica o filtro STEMMER_COMPOSTO, e utiliza o avaliador DISTANCIA_LEVENSTEIN.
STEMMER_HIERARQUIA_COM	Aplica o filtro STEMMER_COMPOSTO, e utiliza o avaliador LEVENSTEIN_HIERARQUIA.
STEMMER_HIERARQUIA_LEV	Aplica o filtro STEMMER_COMPOSTO, e utiliza o avaliador CUMUTATIVIDADE_HIERARQUIA.

Nas 8 hierarquias separadas para estudo na fase de desenvolvimento deste trabalho, observou-se as diferenças entre os resultados obtidos em relação a uma técnica de simples comparação direta de *strings* (processo “IGUAL”, usado como *baseline*). A avaliação de algumas dessas medidas para os 4 conjuntos de alinhamentos (pares de hierarquias) de referência pode ser vista nas Figura 27 **- Resultados encontrados para o conjunto 1**, 28, 29 e 30.

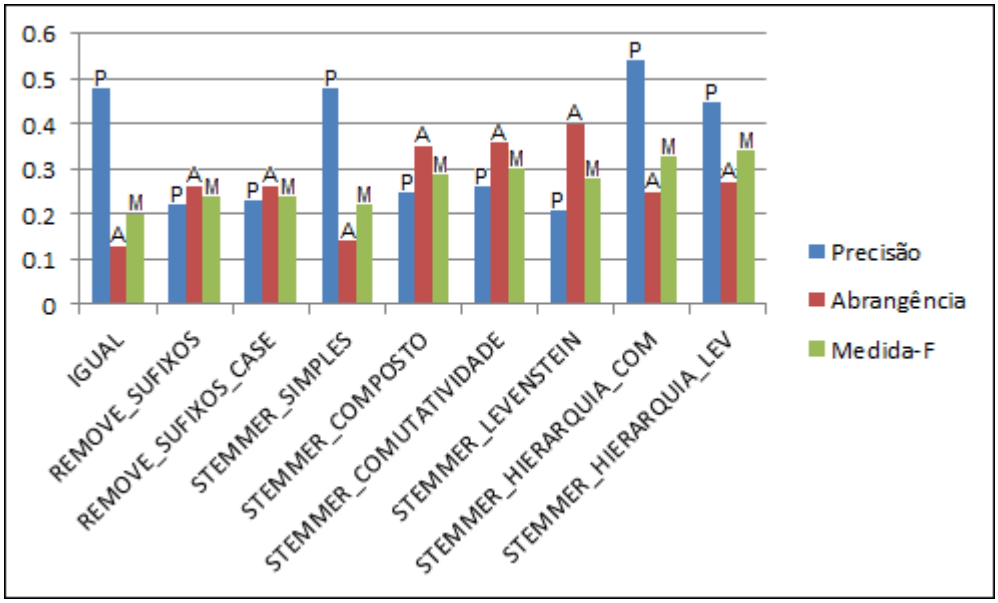

Figura 27 - Resultados encontrados para o conjunto 1

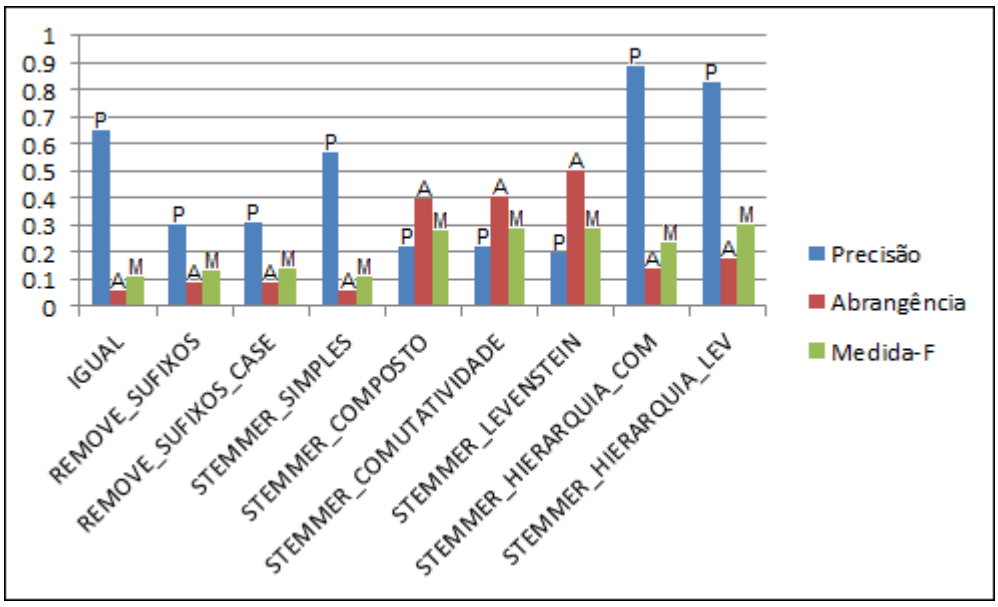

Figura 28 - Resultados encontrados para o conjunto 2

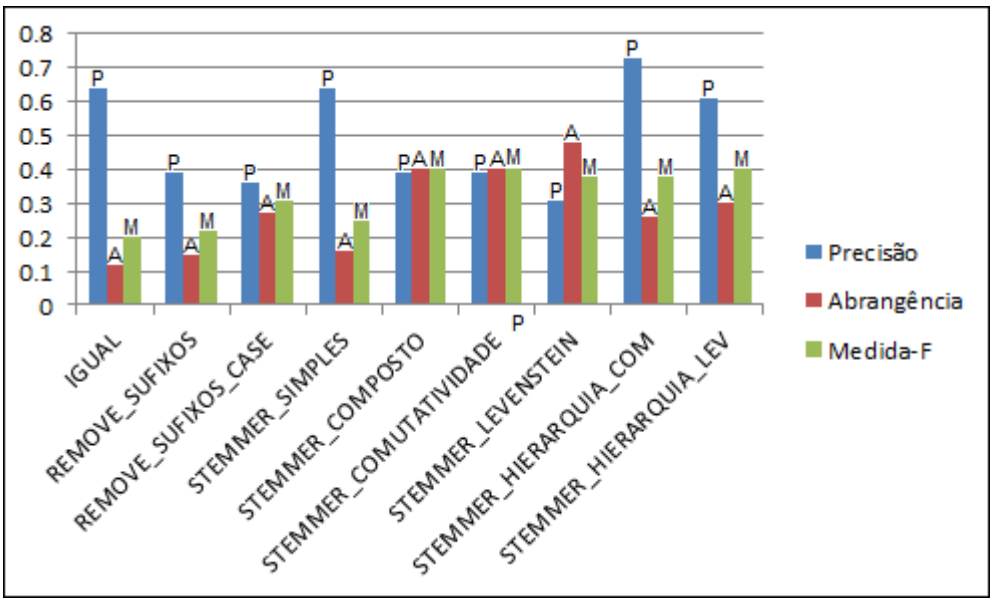

Figura 29 - Resultados encontrados para o conjunto 3

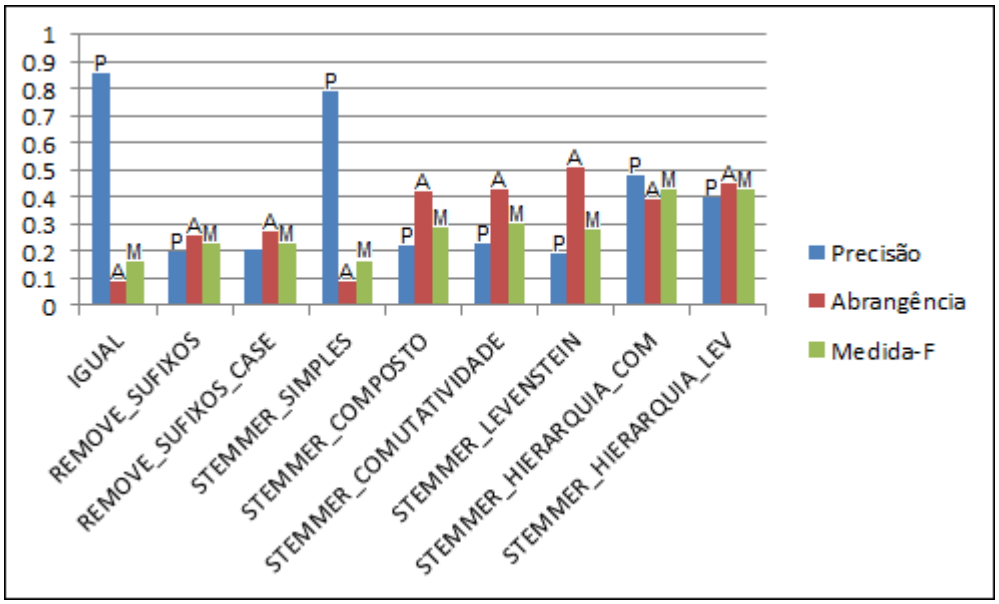

Figura 30 - Resultados encontrados para o conjunto 4

Analisando os resultados, podemos já verificar a eficiência dos métodos propostos mais completos (por exemplo o processo “STEMMER_HIERARQUIA_COM”) em comparação a outros simples disponível em algumas ferramentas (por exemplo os processos “IGUAL” ou “REMOVE_SUFIXOS_CASE”). Os métodos simples conseguem tipicamente atingir maiores valores de precisão, porém a um preço de uma abrangência muito próxima de 0%. Por fim na Figura 31 vemos a média desses resultados, dando

destaque para a melhor medida-F, encontrada com o processo “STEMMER_HIERARQUIA_LEV”.

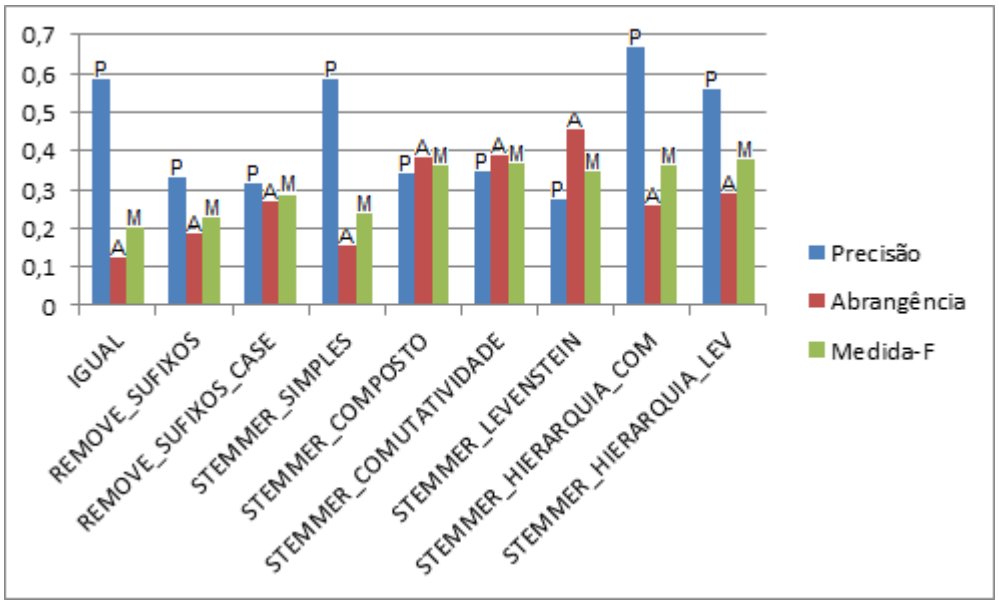

Figura 31 – Média dos resultados encontrados para os conjuntos 1, 2, 3 e 4

Observando as variações dos processos criados em comparação a um elementar como o processo “IGUAL”, a precisão é reduzida a fim de atingir uma melhoria significativa na abrangência, que teve aumento de até 36% no melhor caso.

Enquanto isso, analisando todas as variações da medida-F, que nos da à avaliação definitiva, temos uma melhora de até 27%. Percebe-se esse aumento principalmente nas técnicas que utilizam a análise da hierarquia no avaliador.

A justificativa para os melhores valores em processos que utilizam avaliadores hierárquicos, é baseada na maior confiança que se tem ao analisar também a posição de um elemento. Por exemplo, em um processo que utilize apenas técnicas em nível de elemento, como o “STEMMER_COMUTATIVIDADE”, alinhamentos como o da entidade “Malas → Acessórios” com “Informática → Acessórios” é erroneamente criado (junto com outros para diferentes nodos raiz), principalmente pela alta quantidade de categorias com esse nome. Enquanto um processo que analisa a estrutura junto com o texto, como o “STEMMER_HIERARQUIA_COM”, é capaz de mapear “Malas → Acessórios” somente com “Malas & Bolsas → Acessórios”, devido à similaridade da entidade pai.

Da análise dos resultados obtidos com os experimentos iniciais, pode-se destacar como os processos mais eficientes o “STEMMER_HIERARQUIA_COM” e “STEMMER_HIERARQUIA_LEV”. Em sua composição, ambos aplicam quase todos os filtros disponíveis no sistema, aproximando os nomes o máximo possível. Sua diferença

então está no uso de um avaliador que analisa (junto com a hierarquia, principal diferencial) ou a distância de Levenstein ou a igualdade de entidades listadas na expressão. Sendo suas variações bem pequenas, com os resultados do processo que utiliza a comutatividade tipicamente obtendo uma precisão maior, enquanto o que utiliza a análise da comutatividade de termos obtendo uma abrangência maior.

Com os resultados encontrados durante as avaliações do desenvolvimento (usando os 4 conjuntos separados para tal), foram selecionados os 9 processos vistos na Tabela 5, que permitem a visibilidade de melhora nos resultados, assim como obtenção dos mais otimizados. Os mesmos foram definidos como o conjunto de processos a ser executados pelo módulo de alinhamento automático, para buscar os melhores resultados durante a execução.

4.3. AVALIAÇÃO

Utilizando as 6 hierarquias separadas para a avaliação final dos processos, foram elaborados 3 alinhamentos de referência (conjuntos 5, 6 e 7), que foram utilizados para testar a eficiência da solução. Os resultados da execução desses processos podem ser vistos na Figura **32** - Resultados encontrados para o conjunto 5, na Figura **33** - Resultados encontrados para o conjunto 6 e na Figura **34** - Resultados encontrados para o conjunto 7.

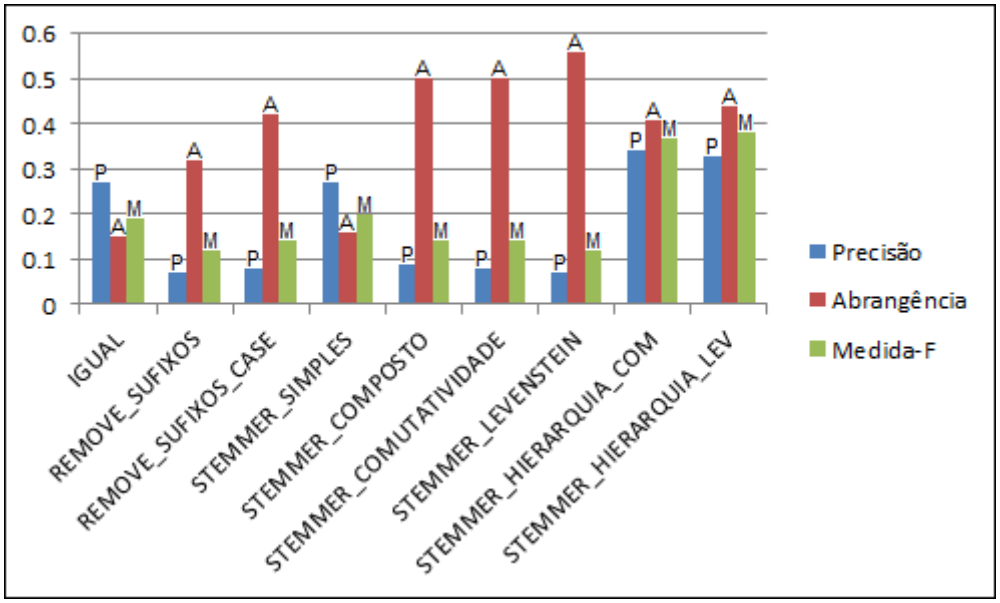

Figura 32 - Resultados encontrados para o conjunto 5

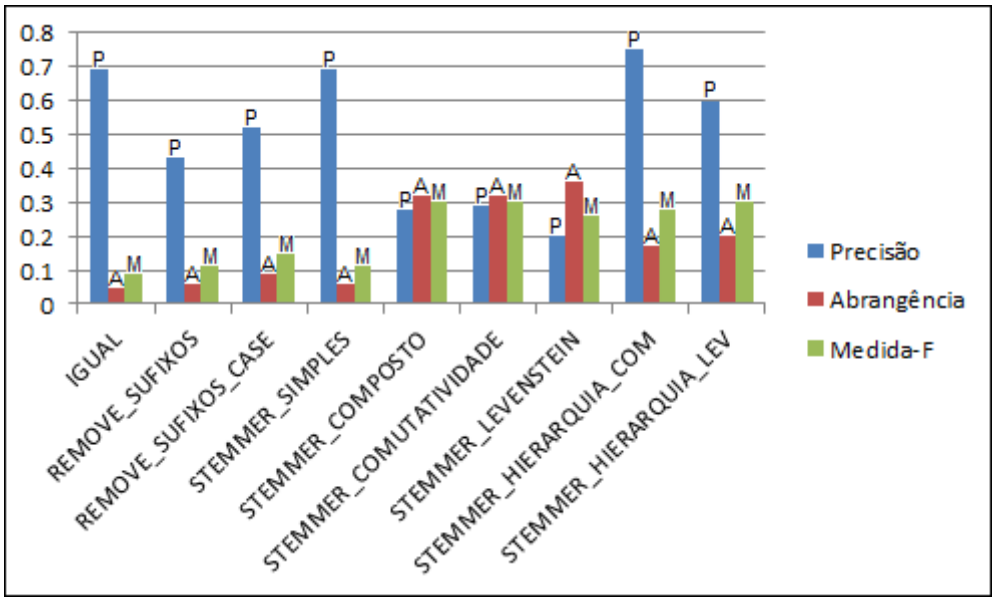

Figura 33 - Resultados encontrados para o conjunto 6

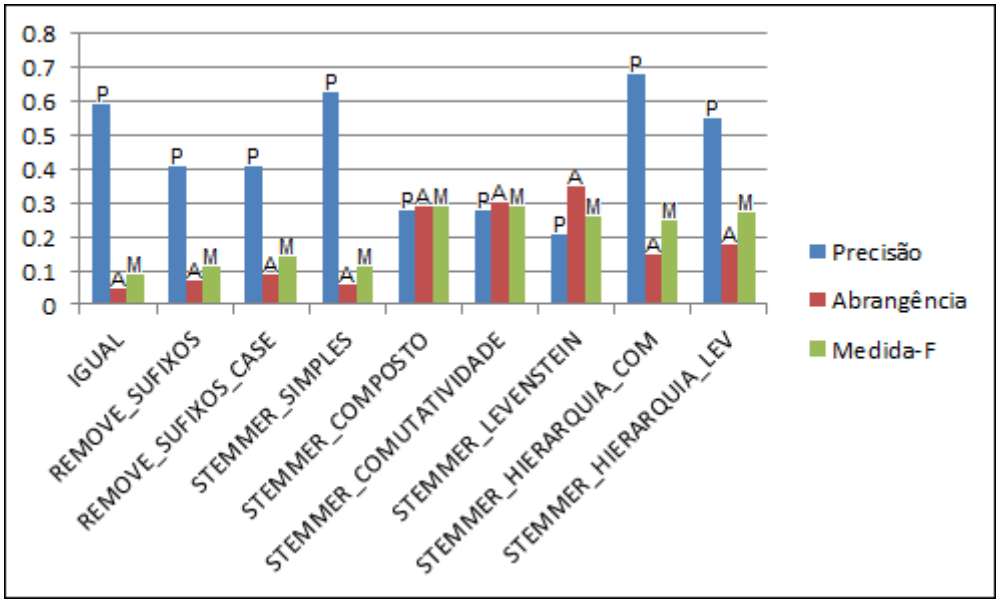

Figura 34 - Resultados encontrados para o conjunto 7

Conforme esperado, quando analisados os resultados do conjunto de testes para cada alinhamento, quando comparado ao processo “IGUAL”, temos uma melhora significativa dos números de avaliação. Em particular dos processos que aplicam o resultado do melhor conjunto de filtros encontrado, com os avaliadores que consideram a estrutura em que se encontra o elemento, mostrando a importância de tal consideração quando analisando hierarquias. Analisando a medida-F, temos um resultado de melhora de 0.19 para 0.38 no conjunto 5, de 0.09 para 0.30 no conjunto 6 e de 0.09 para 0.27 no conjunto 7. Por fim, podemos ainda analisar, conforme a Figura 35, a média dos resultados, que ressalta o processo “STEMMER_HIERARQUIA_LEV” como a melhor medida-F com uma margem de diferença significante (0.19 em comparação ao processo “IGUAL”).

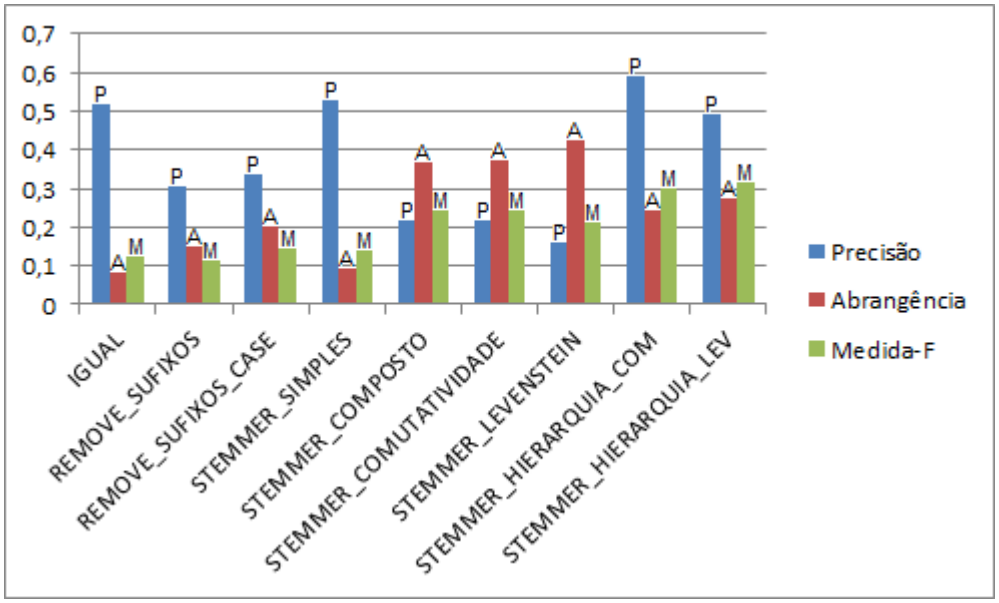

Figura 35 - Média dos resultados encontrados para os conjuntos 5, 6 e 7

Realizando uma análise mais detalhada dos alinhamentos, a partir do arquivo gerado, podemos verificar dois outros fatores que podem ser relevantes para os resultados. O primeiro é a questão da construção dos alinhamentos de referência estarem sujeitos ao ponto de vista de quem os construiu, podendo conduzir o programa a deduções que necessitariam de uma formalidade semântica. Por exemplo, o alinhamento de “ramais“ com “telefonia empresarial”. O segundo fator é praticamente o oposto ao primeiro, onde a aplicação consegue sugerir alinhamentos que um humano acaba por não perceber ao construir manualmente, o que eleva o valor da ferramenta construída.

4.4. AVALIAÇÃO USANDO OUTROS MATCHERS

Buscando comparar os resultados obtidos com as heurísticas construídas para aplicar os módulos da ferramenta PLATAL ao comércio eletrônico, descritas no Subcapítulo 4.3, foram executados outros *matchers* e analisados seus resultados. Para essa atividade, foram utilizados os *matchers* Aroma [4], Aflood [2] e Falcon [20], utilizados em edições da OAEI. Seus resultados para cada um dos conjuntos de teste (5, 6 e 7)

podem ser vistos na Figura **36** – Resultados de outros *matchers* para o conjunto 5, na Figura **37** e na Figura **38**.

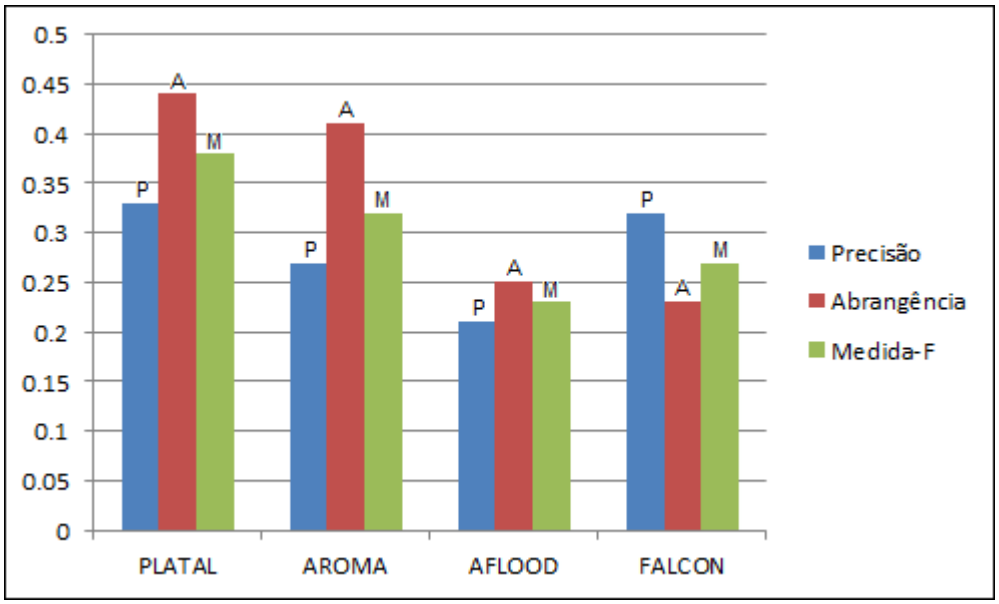

Figura 36 – Resultados de outros *matchers* para o conjunto 5

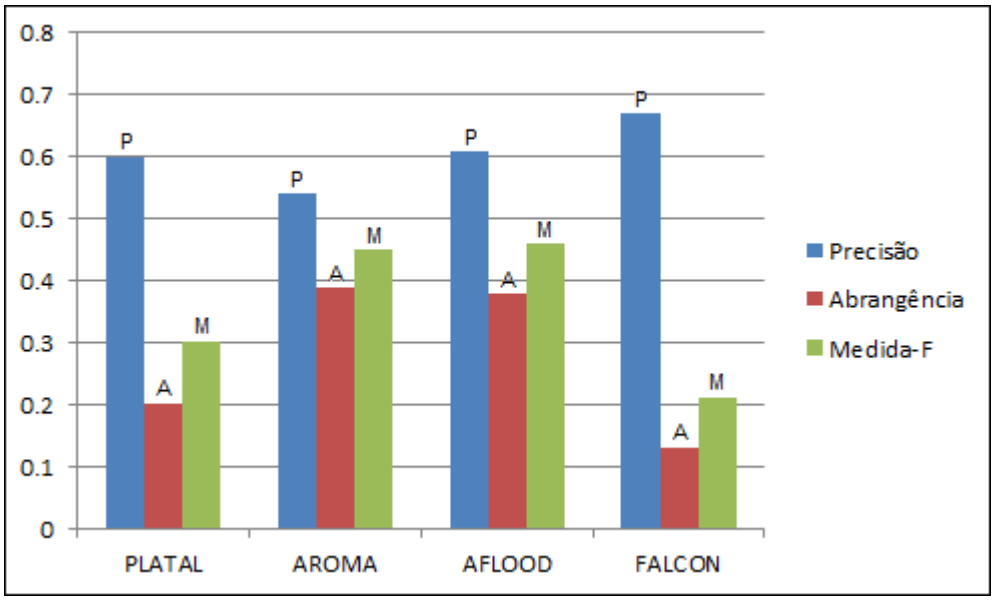

Figura 37 - Resultados de outros *matchers* para o conjunto 6

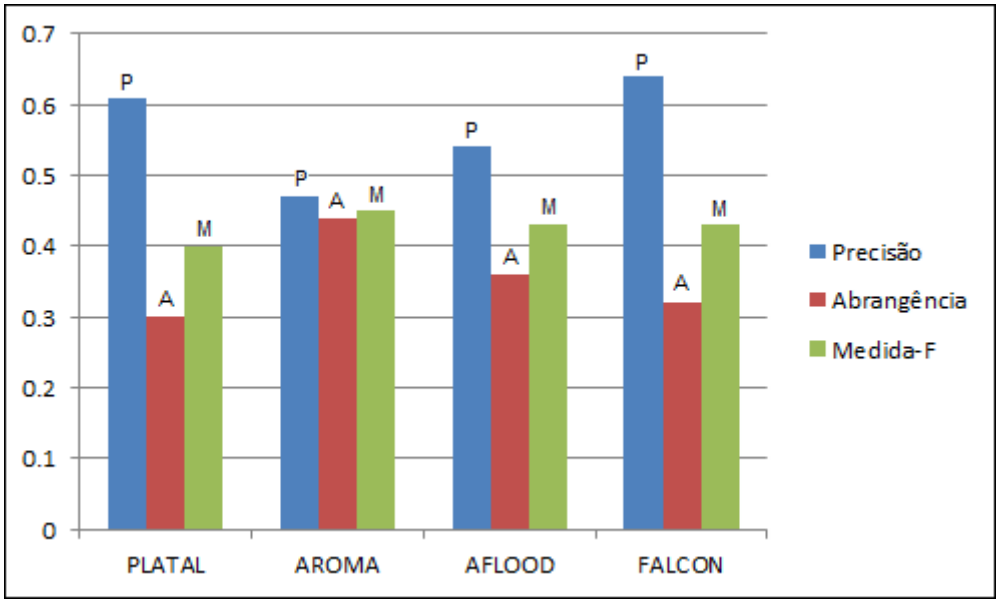

Figura 38 - Resultados de outros *matchers* para o conjunto 7

Através da comparação dos resultados da ferramenta PLATAL em relação as outras, podemos verificar que seus valores são satisfatórios, uma vez que se enquadram na média de resultados dos demais *matchers*. Em relação ao conjunto de testes 5, observamos que ela supera as outras nas três medidas; no conjunto 6 porém as ferramentas Aroma e Aflood a ultrapassam analisando a medida-F diretamente; e por fim no conjunto 7 ela acaba se apresentando a com menor medida-F, porém com diferença máxima de apenas 0.05.

Em uma última comparação, podemos analisar a Figura 39, onde temos a média dos resultados das diferentes ferramentas. Podemos ver que a maior variação da medida-F (entre a pior e a melhor ferramenta) é de apenas 0.11, sendo a diferença da ferramenta PLATAL para o melhor resultado de apenas 0.4.

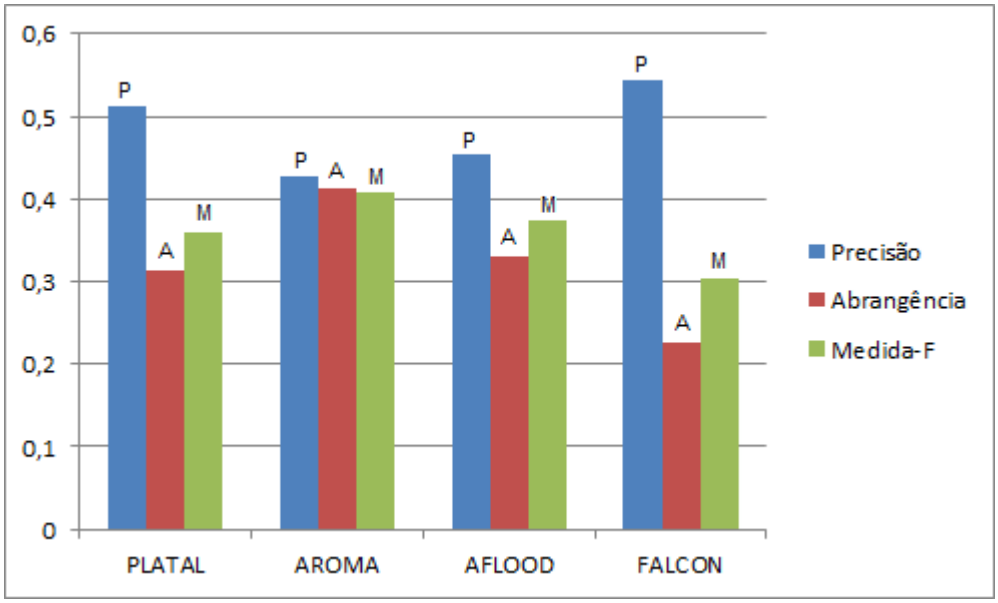

Figura 39 – Média dos resultados de outros matchers para os conjuntos 5, 6 e 7

De uma forma geral, pode-se afirmar que todas as diferenças foram pequenas no desempenho das ferramentas, sendo a maior diferença de medida-F nos conjuntos de teste de 0.15, quando a PLATAL se mostrou superior a todas as outras, caso provavelmente associado a dois fatores: primeiro, o maior volume de entidades presente nesse conjunto em relação aos outros, segundo as técnicas de análise da estrutura utilizados pela PLATAL, que tiram proveito de uma maior quantidades de dados para se aperfeiçoar nos alinhamentos. Tornando assim possível também afirmar que os métodos elaborados para testes da ferramenta PLATAL são competitivos no meio acadêmico atual. Ainda assim meios de melhorar esse desempenho podem ser encontrados na Subseção 5.2.

5. CONSIDERAÇÕES FINAIS

Nesta seção, são exploradas as contribuições do trabalho, assim como trabalhos futuros que podem ser derivados do mesmo.

5.1. CONTRIBUIÇÕES E CONCLUSÕES

Com o desenvolvimento deste trabalho, foram propostos e avaliados alguns métodos para a extração e alinhamento de hierarquias heterogêneas, estudo diferenciado na área de alinhamentos. Acredita-se que foi possível contribuir para que tarefas de IC que dependem da análise de informações distribuídas em diferentes fontes possam ser encontradas com mais facilidade.

Este trabalho dispõe de funcionalidades inexistentes ou de difícil acesso em demais aplicações, em um esforço de prover novas tecnologias, similar às aplicações submetidas à OAEI. Funcionalidade que a tornam um diferencial incluem: uma forma visual de tratar as operações de alinhamento, um repositório centralizado de heurísticas em português para aplicação, a possibilidade de se extrair de forma parametrizável estruturas hierárquicas da internet, entre outras.

Os quatro módulos desenvolvidos, fazem da ferramenta PLATAL um recurso completo e diferenciado para as atividades de alinhamento de hierarquias, começando desde o módulo de extração de hierarquias da web, passando pelos módulos de alinhamento de referência manualmente (visualmente) e alinhamento automático, até o módulo de avaliação que permite a comparação de alinhamentos para validar sua corretude.

No que se refere aos resultados obtidos com a avaliação realizada, deve-se destacar a eficiência dos métodos propostos, que, no melhor caso no cenário de teste comparado com a *baseline*, obtiveram uma melhora da medida-F de 0.09 para 0.30 (conjunto 6). Enquanto na relação com outros *matchers*, se obteve uma medida-F superior em 0.15 as demais no melhor cenário (conjunto 5). Para isso mais uma vez se mostrou a importância da análise estrutural de hierarquias aqui criada para seu alinhamento.

Com a arquitetura da solução implementada, é possível estender a ferramenta através da inclusão de novas técnicas ou mudanças nas hierarquias, e com isso mantê-la atualizada. Essa necessidade é observada não apenas no contexto de empresas, mas também na realidade dos usuários finais de sistemas de *e-commerce*, dada a constante variação das estruturas nos *websites*.

5.2. TRABALHOS FUTUROS

Conforme visto ao longo do texto, o projeto buscou ser modularizado e configurável, de forma que possa ser estendido facilmente. Trabalhos futuros podem partir da estrutura e métodos aqui utilizados para o desenvolvimento de combinações ainda mais precisas no processo de alinhamento.

Para melhorar a parte de avaliação dos resultados, pode-se medir a concordância entre alinhamentos de referência, que podem ser subjetivos.

Outra melhoria que pode ser realizada é o desenvolvimento de uma forma de determinar automaticamente as expressões necessária para identificar os níveis de uma hierarquia em uma página web, automatizando praticamente todo o processo de criação de alinhamentos.

Uma opção de estudo futuro seria a criação de novos filtros e avaliadores para o módulo de alinhamento automático que utilizassem novas técnicas. Por exemplo, técnicas de interpretação baseadas em recursos externos, como técnicas de reuso de alinhamento e técnicas de recursos linguísticos, para avaliadores. Até mesmo parametrizar os valores de aceitação das técnicas aqui desenvolvidas através de interface pode vir a contribuir na flexibilidade da execução.

Por fim, um trabalho importante pode ser desenvolvido no âmbito de usar os alinhamentos resultantes desse trabalho para criar um alinhamento único e disponibilizar o mesmo para o usuário, de forma a usar apenas esse para se referir as categorias de diversas fontes.

REFERÊNCIAS

[1] ABRAIC. Associação Brasileira dos Analistas de Inteligência Competitiva. Capturado em: http://www.abraic.org.br, Junho 2011.

[2] Aflood. “Anchor-Flood”. Capturado em: http://mindblast.informatik.uni-mannheim.de:8080/sealstools/aflood/matcherWS?wsdl, Dezembro 2011.

[3] Alistair, M.; Bechhofer, S. "SKOS Simple Knowledge Organization System Reference". Capturado em: http://www.w3.org/TR/skos-reference/, Julho 2011.

[4] Aroma. “AROMA: extensional and scalable ontology matcher”. Capturado em: http://exmo.inrialpes.fr/software/aroma/, Dezembro 2011.

[5] Berners-Lee, T.; Fischetti, M. “Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor”. Harper San Francisco, 1999.

[6] Bin, S.; Kuiyu, C. “Generating a concept hierarchy for sentiment analysis”. In: International Conference on Systems Man and Cybernetics, 2008, pp. 312-317.

[7] Bocassanta, F.; Dorneles, C. F. “Efetuando Junção por Similaridade de Documentos XML Armazenados em Sistemas de Arquivo”. Revista IEEE América Latina, vol. 8, 2010, p. 3.

[8] Brandolt, J. F. A. “Uso de Relações Semântico-Estruturais no Aprimoramento do Cálculo da Similaridade Semântica: um Estudo Voltado a estruturas Ontológicas em Língua Portuguesa”, Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2009.

[9] Calero, C.; Ruiz, F.; Piattini, M. “Ontologies for Software Engineering and Software Technology”. Springer, 2006.

[10] Clark, J.; DeRose, S. “XML Path Language (XPath) Version 1.0". Capturado em: http://www.w3.org/TR/xpath/, Julho 2011.

[11] Collovini, S,; Grando, F.; Souza, M.; Freitas, L.; Vieira, R. “Semantic Relations Extraction in the Organization Domain” . In: IADIS International Conference Applied Computing, 2011.

[12] David, J.; Euzenat, J.; Scharffe, F.; Santos, C. T. “The Alignment API 4.0”. Semantic web journal, Jun-2010.

[13] Dorneles, C. F; Heuser, C. A.; Lima, A. E. N.; Silva, A.; Moura, E. “Measuring similarity between collection of values”. In: 6th ACM International Workshop on Web Information and Data Management, 2004, pp. 56-63.

[14] eCl@ss. “eCl@ss: Classification and Product Description”.Capturado em: http://www.eclass.de/, Março 2011.

[15] eCommerceOrg. “Evolução da Internet e do e-commerce”. Capturado em: www.e-commerce.org.br/stats.php, Setembro 2011.

[16] eCommerceOrg. “Meios de pagamento ecommerce”. Capturado em: http://www.e-commerce.org.br/meiosdepagamento-ecommerce.php, Setembro 2011.

[17] Euzenat, J. “An API for ontology alignment”, ISWC, 2004.

[18] Euzenat, J.; Shvaiko, P. "Ontology Matching". Springer, 2007.

[19] Euzenat, J.; Stuckenschmidt, H. "The 'Family of Languages' Approach to Semantic Interoperability". In: Knowledge Transformation for the Semantic Web, 2003, pp; 49-63.

[20] Falcon. “Falcon-AO: A practical ontology matching tool”. Capturado em: http://ws.nju.edu.cn/falcon-ao/, Dezembro 2011.

[21] Feng, Y.; Xu, H.; Fang, X. “An Intelligent Recommendation Method of E-Commerce Based on Ontology”. In: Business Intelligence and Financial Engineering, 2009, pp 592-594.

[22] Fensel D. “The Semantic Web: Research and Applications”. Springer-Verlag, 2004.

[23] Fleisher, C. S.; Blenkhorn, D. L. “Managing frontiers in competitive intelligence”. Quorum Books, 2001.

[24] Freitas, J. B. “SiSe: Medida de Similaridade Semântica Entre Ontologias Em Português”, Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2007.

[25] Guo, W. “Ontology Design for Supporting Matchmaking in E-commerce”. In: International Symposium on Information Science and Engieering, 2008, pp. 410-413.

[26] Hepp, M. “A Methodology for Deriving OWL Ontologies from Products and Services Categorization Standards”. In: Int'l Journal on Semantic Web & Information Systems, vol. 2, Jan-Mar 2006, pp. 72-99.

[27] IBGE. “Pesquisa Mensal de Comércio”. Capturado em http://www.ibge.gov.br/home/estatistica/indicadores/comercio/pmc/pmc_201110_01.shtm, Novembro 2011.

[28] Isele, R.; Bizer, C. “Learning Linkage Rules using Genetic Programming”. In: Sixth International Workshop on Ontology Matching, 2011.

[29] ISWC. "The 10th International Semantic Web Conference". Capturado em: http://iswc2011.semanticweb.org/, Outubro 2011.

[30] Lee, H.; Shim, J. “Product Ontology and OWL Correspondence”. In: Studies in Computational Intelligence, 2008, pp. 1-14.

[31] Lopes, L.; Vieira, R.; Finatto, M. J.; Martins, D. “Extracting compound terms from domain corpora”, Journal of the Brazilian Computer Society, vol. 16-4, Nov-2010, pp 247-259.

[32] Malucelli, A.; Oliveira, E.; Palzer, D.” Combining Ontologies and Agents to help in Solving the Heterogeneity Problem in E-Commerce Negotiations”. In: International Workshop on Data Engineering Issues in E-Commerce, 2005, pp. 26-38.

[33] Massmann, S.; Raunich, S.; Aumuller, D.; Arnold, P.; Rahm, E. “Evolution of the COMA Match System”. In: Sixth International Workshop on Ontology Matching, 2011.

[34] Ngomo, A. N. “A Time-Efficient Hybrid Approach to LinkDiscovery”. In: Sixth International Workshop on Ontology Matching, 2011.

[35] Ngomo, A. N.; Lehmann, J.; Auer, S.; Höffner, K. “RAVEN - Active Learning of Link Specifications”. In: Sixth International Workshop on Ontology Matching, 2011.

[36] OAEI. "Ontology Alignment Evaluation Initiative 2011 Campaign". Capturado em: http://oaei.ontologymatching.org/2011/, Outubro 2011.

[37] OM-211. "The Sixth International Workshop on Ontology Matching". Capturado em: http://om2011.ontologymatching.org/, Outubro 2011.

[38] Rahm, E.; Do, H.; Maßmann, S. "Matching large XML schemas". In: ACM SIGMOD Record, 2004, pp. 26–31.

[39] Reusch, P. J. A.; Garcia, L. E. “Implementation of new ontologies for the classification of commodities in e-commerce”. In: Intelligent Data Acquisition and Advanced Computing Systems – IDAACS, 2009, pp 549-554.

[40] RosettaNet. “RosettaNet Technical Dictionary”. Capturado em: http://www.rosettanet.org/, Março 2011.

[41] Saccol, A. I. C. Z. “Alinhamento estratégico da utilização da internet e do comércio eletrônico: os casos Magazine Luiza e Fleury”. Revista de Administração Contemporânea, vol. 9, 2005, pp. 59-80.

[42] SCIP. “Society of Competitive Intelligence Professionals“. Capturado em: http://www.scip.org, Junho 2011.

[43] Selenium Project. “Selenium 2.0 and WebDriver”. Capturado em: http://seleniumhq.org/docs/03_webdriver.html, Agosto 2011.

[44] Shane, S.; Venkatraman, S. “The promise of entrepreneurship as a field of research”. In Academy of Management Review, 2000, pp. 217-226.

[45] SKOS. "SKOS Simple Knowledge Organization System". Capturado em: http://www.w3.org/2004/02/skos/, Outubro 2011.

[46] Smith, M. K.; Welty, C.; McGuinness, D. L. "OWL Web Ontology Language Guide". Capturado em: http://www.w3.org/TR/2004/REC-owl-guide-20040210, Julho 2011.

[47] Stanford University School of Medicine. "protégé". Capturado em: http://protege.stanford.edu/, Março 2011.

[48] Souza, M.; Vieira, R.; Busetti, D.; Chishman, R.; Alves, I. M. “Construction of a Portuguese Opinion Lexicon from multiple resources”. In: Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana, 2011.

[49] Studer, R.; Benjamins, V. R.; Fensel D. “Knowledge Engineering: Principles and Methods”, Data & Knowledge Engineering, vol. 25, Mar-1998, pp. 161-197.

[50] UNSPSC. “United Nations Standard Products and Services Code”. Capturado em: http://www.unspsc.org/, Março 2011.

[51] Vegetti, M.; *et al*. “PRONTO: An ontology for comprehensive and consistent representation of product information”. Engineering Applications of Artificial Intelligence, vol. 4-8, Fev-2011, pp. 1305–1327.

[52] W3C OWL Working Group. "OWL 2 Web Ontology Language Document Overview". Capturado em: http://www.w3.org/TR/owl2-overview/, Julho 2011.

[53] World Wide Web Consortium (W3C). “W3C Semantic Web Activity”. Capturado em http://www.w3.org/2001/sw/ , Novembro 2011.