UNIVERSIDADE FEDERAL DO CEARÁ
Departamento de Computação
Mestrado e Doutorado em Ciência da Computação

Uma abordagem semi-automática para geração incremental de correspondências entre ontologias

Dissertação de Mestrado

FERNANDO WAGNER BRITO HORTÊNCIO FILHO

Fortaleza, CE – Brasil/2011

UNIVERSIDADE FEDERAL DO CEARÁ
Departamento de Computação
Mestrado e Doutorado em Ciência da Computação

Uma abordagem semi-automática para geração incremental de correspondências entre ontologias

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. José Antônio F. de Macedo
Co-orientadora: Profa. Dra. Bernadette F. Lóscio

FERNANDO WAGNER BRITO HORTÊNCIO FILHO

Agradecimentos

O meu mestrado não se desenvolveu da maneira por mim esperada. Alguns acontecimentos tornaram o processo longo e deveras conturbado. Porém, mantive-me firme, e hoje posso ostentar, com todas as honras e méritos cabíveis, o título de Mestre em Ciências da Computação. Entretanto, não posso deixar de agradecer as pessoas que foram fundamentais na participação deste mestrado. Sendo assim, agradeço:

Primeiramente aos meus pais e irmãos, pela força, empenho e motivação concedidos a mim para a realização deste trabalho.

À minha eterna orientadora, Bernadette Farias Lóscio, pelo carinho e apoio, mesmo por diversas vezes, estando distante geograficamente.

Às minhas amigas Fernanda Lígia e Eveline Sacramento pela força motivacional e ajuda técnica no desenvolvimento deste trabalho.

Aos meus amigos biólogos Francisco Welves Maia, Célio Moura e Ivna Lins pela ajuda com os experimentos feitos nesta dissertação.

Aos meus queridos avôs e avós, em especial, ao meu avô paterno, Sr. Paulo Hortêncio de Medeiros Filho, pelos sábios conselhos e por sempre acreditar no meu potencial. Que você esteja bem, e vendo de um lugar bem melhor, tudo o que está acontecendo por aqui.

Aos meus amigos e colegas de mestrado, em especial, Ticianne Ribeiro, Clayton Maciel e Bruno de Carvalho Leal pelos bons momentos proporcionados.

Ao meu orientador José Macedo por me orientar, mesmo possuindo outros orientandos

À Nathália Lima Pedrosa, pela incrível força e paciência empregadas já na parte final da caminhada.

Ao Rock´N´Roll, Blues, e Heavy Metal, pela força, animação e auto-estima concedidas. Vocês, companheiros de longa data, mais uma vez foram fundamentais para meu êxito.

A todos que, de alguma forma, colaboraram para o desenvolvimento deste trabalho...

Resumo

A descoberta de correspondências semânticas entre esquemas é uma importante tarefa para diversos domínios de aplicações, tais como integração de dados, data warehouse e mashup de dados. Na maioria dos casos, as fontes de dados envolvidas são heterogêneas e dinâmicas, dificultando ainda mais a realização dessa tarefa. Ontologias vêm sendo utilizadas no intuito de definir vocabulários comuns usados para descrever os elementos dos esquemas envolvidos em uma determinada aplicação. O problema de matching entre ontologias, ou ontology matching, consiste na descoberta de correspondências entre os termos dos vocabulários (representados por ontologias) usados entre as diversas aplicações. As soluções propostas na literatura, apesar de serem totalmente automáticas possuem natureza heurística, podendo produzir resultados não-satisfatórios. O problema se intensifica quando se lida com grandes fontes de dados. O objetivo deste trabalho é propor um método para geração e refinamento incremental de correspondências entre ontologias. A abordagem proposta faz uso de técnicas de filtragem de ontologias, bem como do feedback do usuário para dar suporte à geração e ao refinamento dessas correspondências. Para fins de validação, uma ferramenta foi desenvolvida e experimentos foram realizados.

Palavras-Chave

Ontologias, matching, geração incremental, refinamento incremental, filtragem, feedback do usuário.

Abstract

The discovery of semantic correspondences between schemas is an important task for different fields of applications such as data integration, data warehousing and data mashup. In most cases, the data sources involved are heterogeneous and dynamic, making it even harder the performance of that task. Ontologies are being used in order to define common vocabulary used to describe the elements of the schemas involved in a particular application. The problem of matching between ontologies, or ontology matching, consists in the discovery of correspondences between terms of vocabularies (represented by ontologies) used between the various applications. The solutions proposed in the literature, despite being fully automatic have heuristic nature, and may produce non-satisfactory results. The problem intensifies when dealing with large data sources. The purpose of this paper is to propose a method for generation and incremental refinement of correspondences between ontologies. The proposed approach makes use of filtering techniques of ontologies, as well as user feedback to support the generation and refining these matches. For validation purposes, a tool was developed and experiments were conducted.

Keywords

Ontology matching, incremental generation, incremental refinement, filtering, user feedback.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

1.1. Motivação

Um dos requisitos para a realização da Web Semântica é a descrição semântica, formal e padronizada dos dados, com o objetivo de tornar possível a interpretação e inferência de informações por parte da máquina. Neste cenário, o conceito de ontologia, bem como boas práticas para sua construção e utilização, vem sendo alvo de estudos em diversos trabalhos. Uma ontologia é definida como sendo uma especificação formal, explícita e compartilhada acerca de um domínio [54]. Ontologias vêm sendo usadas com o intuito de descrever vocabulários. Tais vocabulários definem conceitos e relacionamentos dos termos das bases de dados das aplicações, relacionados a uma área de interesse qualquer.

Além da descrição formal e explícita dos dados, a Web Semântica tem como objetivo estabelecer interoperabilidade semântica entre diversas fontes de dados existentes na Web. Para isso, é necessário que haja uma integração entre as mesmas. Assim, estudos e soluções na área de integração de dados vêm ganhando a atenção da comunidade acadêmica. Nesse contexto, é necessário que haja uma maneira eficiente de integrar esses dados e exibi-los de forma transparente ao usuário.

O principal obstáculo encontrado na tarefa de integração é a criação de mapeamentos entre as fontes de dados, que é dificultada pela heterogeneidade entre as diversas bases de dados, uma vez que tais bases são modeladas com objetivos, formas e tecnologias distintas.

Apesar do advento das ontologias e do esforço por parte da W3C na padronização de tecnologias, a heterogeneidade semântica persiste nas diversas aplicações, uma vez que as mesmas são modeladas de forma independente possuindo assim eventuais divergências semânticas, além de estruturais e terminológicas.

Como exemplo de heterogeneidade semântica, suponha dois esquemas desenvolvidos independentemente que modelam regiões mundiais. Tais esquemas podem ter diferenças de granularidade (no qual um esquema leva em consideração detalhes como cidades, bairros, e o outro apenas as capitais e estados); de ponto de vista ou escopo (um esquema detalha os pontos geográficos e o outro destaca as características políticas); e de cobertura (um esquema dá ênfase para a região Sul e o

outro para a região Norte).

Para solucionar o problema da heterogeneidade semântica, é necessário que haja a construção de mapeamentos entre os termos das diferentes ontologias que descrevem os termos das bases de dados (*schema mapping*). Dessa forma, consultas poderão ser reescritas (*query rewriting*), dados poderão ser transformados (*data translation*) e resultados poderão ser corretamente integrados e retornados ao usuário. Como exemplo de mapeamento entre os esquemas das regiões mundiais, poderia ser citado que uma instância de um conceito REGIÃO de um esquema é equivalente a uma ou mais instâncias de um conceito PAÍS, do outro esquema, que possuem características semelhantes.

Como pré-requisito para a elaboração dos mapeamentos entre as bases, tem-se a necessidade de identificar correspondências semânticas entre as ontologias envolvidas, ou seja, é preciso identificar como os conceitos dos esquemas estão relacionados entre si. Como exemplo de correspondência entre os esquemas das regiões mundiais, poderia ser citados que um conceito PAÍS de um esquema é equivalente a um conceito PAÍS do outro esquema, dependendo do contexto empregado em cada um dos esquemas. Dessa forma, uma correspondência é estabelecida através de uma sentença que envolve esses dois conceitos através de uma relação (equivalência para o caso deste exemplo).

O problema do alinhamento entre esquemas ou *schema alignment* consiste em descobrir e estabelecer correspondências entre os termos do esquema conceitual das bases de dados envolvidas em uma determinada aplicação [21]. No contexto das ontologias, tal problema é conhecido na literatura como alinhamento entre ontologias, *ontology alignment* ou *ontology matching*. Essa operação pode ser bastante complexa, considerando a já citada heterogeneidade semântica.

A definição completa do conjunto de correspondências entre os termos (ou alinhamento), bem como a manutenção das mesmas, exige grande esforço e consumo de tempo, especialmente quando se trabalha com grandes fontes de dados. O uso de ferramentas especializadas em gerar correspondências entre esquemas parece ser uma boa solução, porém os algoritmos utilizados por tais ferramentas são heurísticos, podendo gerar resultados contendo um percentual considerável de falsas correspondências e de ausência de correspondências verdadeiras [10]. Para evitar erros em tarefas que dependam da utilização dessas correspondências geradas automaticamente, faz-se necessário que haja uma correção manual. Porém, efetuar as correções necessárias em grandes bases de dados exige um grande esforço e tempo do

usuário. Deve-se também levar em consideração o gasto de tempo que o algoritmo necessita para gerar um conjunto de correspondências entre os esquemas. Obviamente, quanto maior o tamanho e mais profunda a hierarquia dos esquemas envolvidos, maior será o tempo necessário para gerar uma resposta.

Além disso, a geração de um alinhamento completo entre duas ontologias nem sempre se faz necessária. Como exemplo, cita-se a definição de correspondências entre ontologias multi-domínios (por exemplo, DBPedia[1] e Yago[2]), em que apenas algumas partes destas compartilham uma conceitualização comum. Outro exemplo está no domínio da bioinformática. A *Gene Ontology*[3], produto do esforço colaborativo de pesquisadores da área, é uma ontologia que visa dar suporte à anotação de genes. Considere o cenário em que se deseja obter um alinhamento entre a *Gene Ontology* e uma ontologia que seja mais específica e que descreve, por exemplo, genes de apenas uma espécie. Em ambos os casos, apenas partes dos esquemas tratam de um domínio em comum.

1.2. Requisitos e contribuições

A definição de correspondências entre ontologias é uma tarefa difícil, em especial para grandes bases de dados. Neste contexto, alguns requisitos são desejáveis em soluções que trabalhem neste tipo de problema. São eles:

- Eficácia: Obter um conjunto de correspondências com um bom índice de corretude (correspondências corretas dentre as recuperadas) e completude (índice de correspondências corretas dentre as possíveis) constitui um grande desafio e requisito para soluções de *matching*. Isto porque quanto maior o número de conceitos dos esquemas, maior é o tamanho do problema (espaço de busca dos algoritmos) e maior a probabilidade de erro por parte das soluções existentes. Neste trabalho, o processo de geração de correspondências é feito iterativamente, em que em cada iteração, partes dos esquemas são alinhados, diminuindo assim o

[1] *http://www.**dbpedia**.org*
[2] http://www.mpi-inf.mpg.de/yago-naga/yago/
[3] http://www.geneontology.org/

espaço de busca e a probabilidade de erros. Além disso, a abordagem proposta é interativa, onde o usuário poderá prover um *feedback* acerca dos resultados.

- Eficiência: Grandes esquemas geram grandes espaços de busca, que necessitam de mais tempo para serem processados por algoritmos. A resolução incremental proposta neste trabalho visa diminuir o tempo gasto no processo por parte dos algoritmos, diminuindo o tamanho do problema e alinhando partes relevantes dos esquemas.
- Minimizar o trabalho manual: Em grandes bases de dados, torna-se difícil efetuar acertos em um alinhamento devido à grande quantidade de informação envolvida (tamanho do alinhamento gerado e o grande número de conceitos). Neste sentido, o trabalho em questão propõe uma solução iterativa, que limita a quantidade de informações processadas pelo usuário em cada iteração. O alinhamento é gerado incrementalmente, à medida que o usuário explora os esquemas envolvidos.

As principais contribuições deste trabalho são: (i) Propor uma abordagem iterativa, semi-automática para a geração e refinamento incremental de correspondências entre ontologias que descrevem grandes bases de dados heterogêneas; (ii) Criação de um método para filtragem de ontologias dentro do contexto da tarefa de *matching*; (iii) Uma maneira de capturar e utilizar o *feedback* do usuário para refinar alinhamentos.

A abordagem proposta no trabalho terá como base a filtragem de termos das ontologias de entrada, bem como o *feedback* do usuário. Além disso, tal abordagem fará uso de algoritmos automáticos já existentes para geração de correspondências no intuito de auxiliar o usuário na obtenção de um alinhamento resultante de qualidade.

1.3. Aplicações

O *matching* entre ontologias é uma importante operação em aplicações tradicionais (por exemplo: *Ontology Engineering*, *Data integration*, *Data Warehouse*) e emergentes (composição de *Web Services*, aplicações *peer-to-peer, query answering*) dentro da Ciência da Computação. Tais aplicações são caracterizadas por conter bases

de dados heterogêneas, que devem ser analisadas e conciliadas, com o objetivo de prover interoperabilidade entre estas.

Euzenat e Shvaiko [21] listam diversas aplicações e casos nos quais o alinhamento entre ontologias vem sendo utilizado como uma solução eficaz para o problema da interoperabilidade. Algumas dessas aplicações são descritas a seguir.

1.3.1. Edição e Importação de Ontologias

O reuso é uma importante característica no processo de desenvolvimento de ontologias. Ao se criar uma ontologia, é importante (i) verificar a existência de outras ontologias relevantes que abordam o mesmo domínio de interesse; (ii) identificar as relações entre os termos das ontologias envolvidas para que se evite a multiplicação de ontologias e se mantenha a uniformidade do vocabulário. Assim, o alinhamento entre as ontologias envolvidas se faz necessário, uma vez que se almeja uma resolução para a heterogeneidade entre os vocabulários envolvidos.

1.3.2. Evolução de Ontologias

Ontologias estão sujeitas a constantes evoluções ao longo do tempo, seja por mudança de requisitos em aplicações ou na maneira como os desenvolvedores conceitualizam tais requisitos. Além disso, há casos em que uma determinada ontologia é desenvolvida e distribuída de forma colaborativa, resultando em diversas versões dessa mesma ontologia. Desenvolvedores precisam gerenciar essas mudanças.

Neste cenário, o alinhamento surge como maneira de relacionar as versões distintas de uma ontologia, focando na descoberta das diferenças entre as mesmas, como a inclusão, remoção e renomeação de conceitos.

1.3.3. Integração de Dados Envolvendo Ontologias

A integração de dados é um problema onde o alinhamento é visto como uma possível solução. O típico cenário envolve bases de dados locais (possivelmente usando diferentes tecnologias para armazenamento de dados) descritas por ontologias (locais) e

uma interface de consulta (esquema de mediação) também descrita por uma ontologia (global) e relacionada com as ontologias locais. Tal arquitetura permite ao usuário interagir apenas com o esquema de mediação para obter as respostas advindas das bases de dados locais.

Neste contexto, alinhamentos são necessários para estabelecer correspondências entre os conceitos das ontologias e auxiliar na reescrita de consultas (em termos do esquema de mediação) bem como na reconciliação dos resultados advindos das múltiplas bases locais, eliminando redundâncias e duplicações antes de retornar o resultado ao usuário final.

1.3.4. Composição de serviços Web

Serviços Web são processos nos quais as interfaces são descritas e publicadas na Web. Através das descrições, o usuário tem conhecimento dos detalhes de cada serviço, bem como suas entradas e saídas, podendo invocá-lo através das interfaces públicas disponíveis. Serviços Web semânticos utilizam ontologias para descrever os seus serviços, entradas e saídas, objetivando fornecer uma descrição mais rica e detalhada do processo que efetuam.

A descoberta e integração de serviços Web é a atividade de encontrar e integrar serviços Web disponíveis, adequados para realizar determinadas tarefas, com o objetivo de alcançar uma meta particular. Assim, serviços podem ser incorporados em *workflows* em que a saída de um constitui a entrada do outro. Para o caso dos serviços Web semânticos, o alinhamento entre as ontologias que os descrevem é peça chave para estabelecer relações com novos serviços disponíveis, bem como interconectar os formatos de entrada e saída entre serviços utilizados.

1.3.5. Sistemas Peer-To-Peer

Peer-to-Peer (P2P) é um modelo de comunicação distribuída na qual as partes envolvidas (*peers*) provêem dados e serviços umas as outras. Com o advento da Web 2.0, redes P2P tornaram-se muito populares devido ao paradigma colaborativo existente. Neste tipo de rede, cada *peer* possui sua própria autonomia para representação dos dados. Redes P2P semânticas utilizam ontologias para a descrição mais rica e detalhada

dos dados, no intuito de aperfeiçoar tarefas com a busca de dados na rede P2P. Nesse contexto, o alinhamento entre as ontologias que descrevem os *peers* facilitam a comunicação entre os mesmos, possibilitando a realização de atividades como a tradução de dados e consultas, auxiliando na interoperabilidade da rede.

1.4. Organização da dissertação

O restante deste trabalho está organizado como se segue: o Capítulo 2 trata do problema de *matching* bem como técnicas e estratégias existentes; o Capítulo 3 descreve a estratégia proposta neste trabalho; o Capítulo 4 apresenta os experimentos empíricos realizados; o Capítulo 5 apresenta uma discussão acerca das vantagens e limitações da abordagem, bem como as conclusões e possíveis trabalhos futuros;

Capítulo 2

Matching entre ontologias

Gerar um alinhamento de qualidade entre grandes ontologias constitui um campo de pesquisa em aberto que vem chamando atenção da comunidade acadêmica, especialmente devido ao advento da Web dos Dados. Neste contexto, várias técnicas foram desenvolvidas, no intuito de prover soluções eficazes e eficientes. Este capítulo lista algumas definições prévias, bem como as técnicas e estratégias específicas para *matching* entre ontologias, e técnicas auxiliares que facilitam esta tarefa, como particionamento de esquemas e o provimento de *feedback* pelo usuário. O capítulo está organizado da seguinte forma: Seção 2.1 descreve o problema de *matching*; Seção 2.2 aborda as principais técnicas para geração de correspondências entre ontologias; Seção 2.3 aborda as estratégias de *matching*; Seção 2.4 trata do estado da arte no uso de duas técnicas para melhoria da qualidade de um alinhamento: particionamento de ontologias e provimento do *feedback* acerca dos resultados por parte do usuário.

2.1 O Problema

Matching é uma operação que tem como entrada duas ontologias ***O*** e ***O'***, um conjunto de correspondências (ou alinhamento) inicial ***A*** (vazio ou não) e um conjunto de parâmetros e recursos, retornando um novo alinhamento ***A'***. A operação de *matching* pode ser ilustrada na Figura **1**.

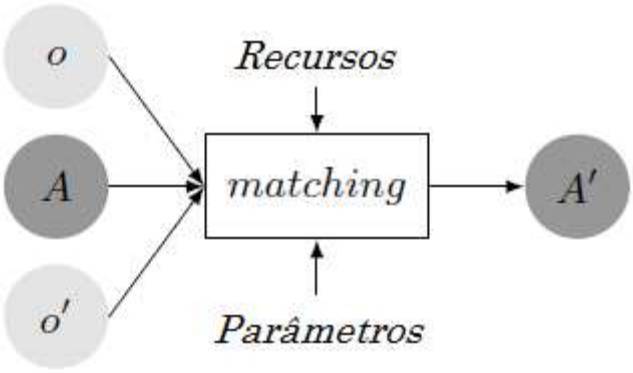

Figura 1. A operação de *matching* [21]

Como exemplo de possíveis **parâmetros**, cita-se o limiar, que funciona como um valor de corte de correspondências, ou seja, em uma abordagem que use limiar, toda

e qualquer correspondência com valor de confiança abaixo do especificado pelo limiar será desconsiderada. Por outro lado, um possível **recurso** a ser usado em uma abordagem de *matching* são as chamadas bases de conhecimento auxiliares, que geralmente consistem de ontologias de domínio. Tais ontologias são consultadas e as informações extraídas auxiliam na construção do conjunto de correspondências de saída. **Ontologias de domínio** são vocabulários que tratam de um domínio específico. Geralmente são ontologias grandes, que possuem uma vasta gama de conceitos relacionados ao domínio. Diferentemente dos dicionários e tesauros, ontologias possuem um rigor formal. Assim, é possível, por exemplo, fazer uso de um motor de inferência para auxiliar a execução de certas tarefas.

A literatura [21, 3] apresenta diversas soluções de *matching*. Tais soluções podem ser classificadas em técnicas básicas para geração de correspondências entre termos de um esquema (sintáticas, estruturais, extensionais, uso de elementos externos, semânticas) e estratégias de *matching* (sequencial, paralela, mista). As duas subseções a seguir abordam e discutem cada uma dessas duas classificações.

2.2 Técnicas básicas de *Matching*

As técnicas básicas para geração de correspondências entre termos de um esquema, ou *matchers*, são algoritmos que, recebem como entrada dois esquemas, e fazem uso de uma técnica específica sobre os conceitos desses esquemas, com o objetivo de gerar correspondências entre as mesmas. A Figura **2** exibe a classificação das técnicas básicas existentes para *matching*, segundo Euzenat [21].

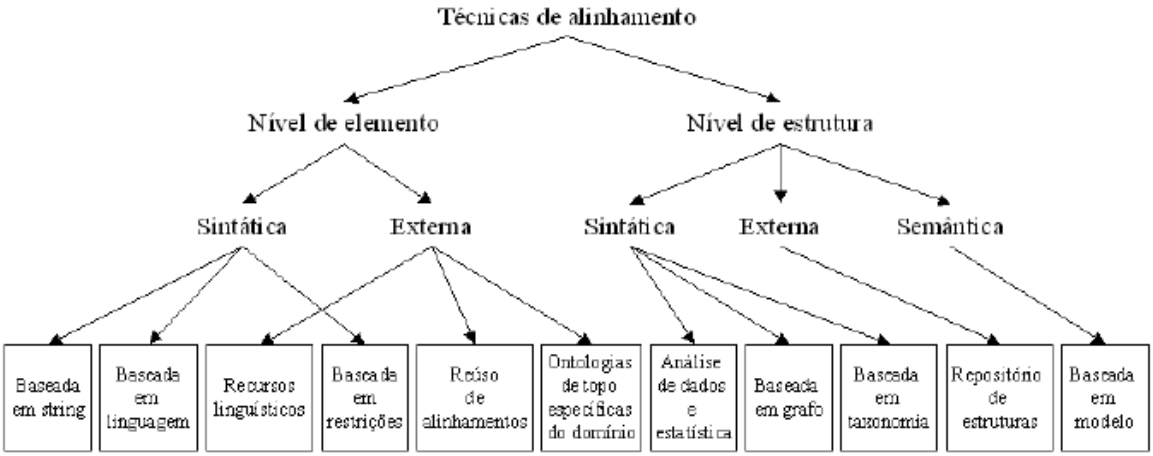

Figura 2. Classificação das técnicas básicas de *matching*. Adaptado de [21]

Euzenat [21] separa as técnicas em duas vertentes: em níveis de elemento e de

estrutura da ontologia. As técnicas em nível de elemento comparam os conceitos isoladamente, gerando valores que definem o grau de similaridade entre cada um desses conceitos. As técnicas em nível de estrutura comparam os conceitos levando em consideração como eles estão apresentados, ou seja, levam-se em consideração as relações existentes com outros conceitos. As próximas cinco subseções abordarão as principais classificações (sintáticas, estruturais, extensionais, uso de elementos externos, semânticas) [21].

2.2.1 Técnicas sintáticas

São métodos que efetuam comparações léxicas entre os valores das cadeias de caracteres relacionadas aos conceitos em questão. Como exemplos dessas cadeias citam-se nomes, comentários e anotações.

Com o intuito de prover uma correta comparação e atingir resultados mais precisos, estas geralmente são acompanhadas por uma etapa anterior, chamada normalização linguística.

A normalização linguística consiste em editar as *strings* dos conceitos de forma a padronizar suas escritas. O processo de normalização é baseado na língua padrão definida pelo usuário para ser usada na concepção do esquema (ex: inglês, português, francês, etc.), e consiste em eliminar alguns termos e caracteres que não trazem significado, como vírgulas, hífens, pontos, etc., análises morfológicas, como supressão de gêneros e verificação de possíveis flexões e derivações da raiz de uma determinada string, além da eventual tradução para a língua padrão.

Uma vez que os termos estão normalizados, existem inúmeras medidas de similaridade que podem ser usadas para a comparação entre duas strings. Dentre as mais usadas, citam-se Leveshtein, Jaro, Jaro-Winkler e os n-gramas, detalhadas em [21].

Apesar da relativa simplicidade por parte das técnicas sintáticas, as mesmas possuem dificuldades na identificação de certos tipos de correspondência. O maior problema reside na existência dos

Sinônimos: Nomes diferentes usados para referenciar os mesmos conceitos. Em alguns contextos, foto e retrato referem-se ao mesmo conceito do mundo real.

Homônimos: Nomes iguais usados para referenciar conceitos diferentes. Como exemplo, cita-se a palavra manga, que, dependendo do contexto, pode referir-se a uma fruta ou a uma parte de uma peça de roupa.

Como conseqüência, nem sempre conceitos com alta similaridade sintática

possuem o mesmo significado e vice-versa. Logo, o uso dessas técnicas é adequado quando as ontologias envolvidas utilizam cadeias de caracteres similares para denominar um mesmo conceito do mundo real, uma vez que, para este caso, irá ser encontrado um grande número de correspondências corretas.

2.2.2 Técnicas Estruturais

Além das cadeias de caracteres, as estruturas dos conceitos também podem ser comparadas com o intuito de identificar correspondências entre ontologias. Os métodos estruturais são construídos sob a intuição de que a similaridade dos conceitos entre duas ontologias é influenciada por suas características estruturais. Essas características podem ser divididas em

- **Estrutura interna**: Nesta categoria, o cálculo da similaridade entre conceitos é baseado nas características de suas propriedades (quantidade, domínio, alvo, cardinalidade, se há transitividades, simetrias, etc.). Suponha os seguintes conceitos contidos na Figura **3**, oriundos das ontologias fonte e alvo respectivamente. Um *matcher* sintático teria dificuldades em identificar que tais conceitos são similares. Porém, um *matcher* estrutural identificaria tal correspondência através de uma análise na similaridade das propriedades envolvidas.

  Apesar deste tipo de *matcher* possuir baixa complexidade e ser de fácil implementação, a estrutura interna dos conceitos nem sempre fornece informações necessárias para se produzir um alinhamento de qualidade. Conceitos não-correspondentes podem conter propriedades similares. Além disso, algumas ontologias existentes são constituídas de pouquíssimas ou até mesmo nenhuma propriedade, como é o caso da *Gene Ontology*. Para estes casos, o *matcher* não será eficaz.

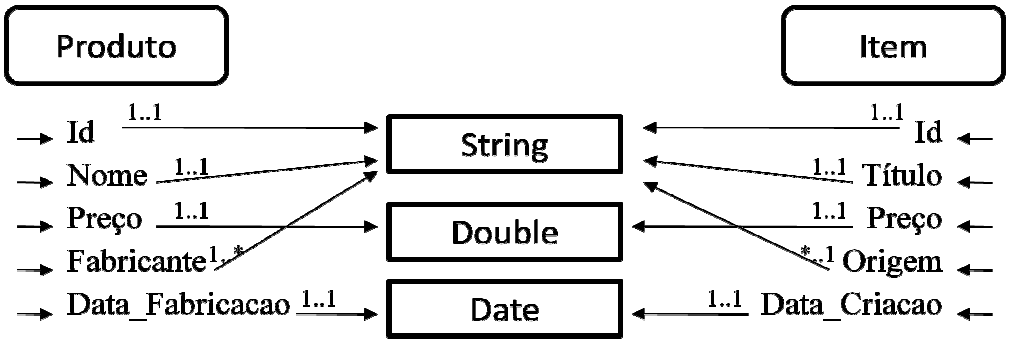

Figura 3. Propriedades de dado de dois conceitos

- **Estrutura externa (ou relacional)**: A similaridade entre dois conceitos é calculada levando-se em consideração as relações externas (Exemplo: generalização, especialização, etc.) e a similaridade entre os conceitos relacionados a estes. Intuitivamente, se um conceito A possui um pai e um filho que são correspondentes ao pai e ao filho de um conceito B respectivamente, então é provável que os conceitos A e B também sejam correspondentes.

Algumas abordagens estruturais requerem um alinhamento inicial para o cálculo de um novo alinhamento. Isso se deve ao fato dessas soluções basearem-se em correspondências previamente estabelecidas para efetuar o cálculo de novas correspondências. Outras abordagens não exigem um alinhamento inicial, mas confiam na existência e disponibilidade de instâncias (nem toda ontologia possui instâncias; neste caso, tal solução não terá efeito), calculando as similaridades entre as estruturas e valores das instâncias, com o intuito de auxiliar na identificação de correspondências entre conceitos.

2.2.3 Técnicas Semânticas

A característica principal das técnicas semânticas é o uso de técnicas de lógica dedutiva baseadas em modelos para a definição de correspondências. Dada a natureza indutiva da tarefa de *matching*, as abordagens semânticas requerem um alinhamento inicial (também chamados de *anchors*) para o cálculo de um novo alinhamento. Dessa forma, o *matcher* semântico trabalha "expandindo" o alinhamento inicial, através da descoberta de novas correspondências por inferências e deduções.

Na prática, tal expansão acontece da seguinte forma: (i) a partir do alinhamento inicial existente, a operação de *merge* entre duas ontologias *o* e *o'* é feita. A operação de

merge consiste em receber como entradas um alinhamento inicial e duas ontologias, e produzir como saída uma única ontologia que representa a fusão das duas ontologias passadas como entrada [55]. Dessa forma, (ii) inferências lógicas são feitas na ontologia resultante do *merge*, a fim de descobrir novos axiomas e expandir o alinhamento inicial. Em outras palavras, o alinhamento resultante do processo é *A*, tal que:

$o, o' \models A$ (*o* e *o'* implicam logicamente em *A*)

Considere o seguinte exemplo: Sejam os conceitos **Mansao** tal que **Mansao** ≡ ***Moradia*** ***\| \| ≤ 8 quarto*** e ***Casa*** tal que ***Casa*** ≡ ***Teto*** ***\| ≤ 3 comodo***. Suponha que as seguintes correspondências tenham sido passadas como entrada para o *matcher* semântico: ***Teto*** ≡ ***Moradia*** e ***Quarto*** ⊆ ***cômodo***. Assim, pode-se deduzir que a seguinte correspondência é válida: ***Mansao*** ⊆ ***Casa***.

Apesar da necessidade de um alinhamento inicial, o uso de técnicas semânticas torna-se importante, uma vez que, com estas, é possível descobrir e eliminar inconsistências no alinhamento bem como diminuir o número de correspondências corretas ausentes no alinhamento resultante (um bom nível de completude do alinhamento), através de deduções e inferências lógicas. Como principal desvantagem desta técnica, cita-se o tempo de processamento para a descoberta de inferências, pois em grandes ontologias, esta operação pode-se tornar ineficiente. Este tipo de *matcher* possui poucas implementações e ainda é um tema com muita pesquisa envolvida [21].

2.2.4 Técnicas Extensionais

As técnicas extensionais focam na análise das instâncias associadas aos conceitos, para a geração de correspondências. Quando dois conceitos compartilham o mesmo conjunto de instâncias, pode-se supor que há uma relação entre os mesmos. Mesmo quando conceitos não compartilham o mesmo conjunto de instâncias, técnicas comparativas são aptas a encontrar regularidades e discrepâncias em cada conjunto, ajudando na descoberta de correspondências. Dessa forma, o uso de técnicas extensionais torna-se adequado quando há ontologias que possuem ou compartilham um conjunto razoável de instâncias. Suponha por exemplo as classes ***Produto*** e ***Item***, pertencentes a *o* e *o'* respectivamente. Seus nomes são sintaticamente distintos, o que dificulta a descoberta de um relacionamento por parte de um *matcher* sintático.

Entretanto, considere a Figura **4** que contém instâncias relativas a cada um dos conceitos.

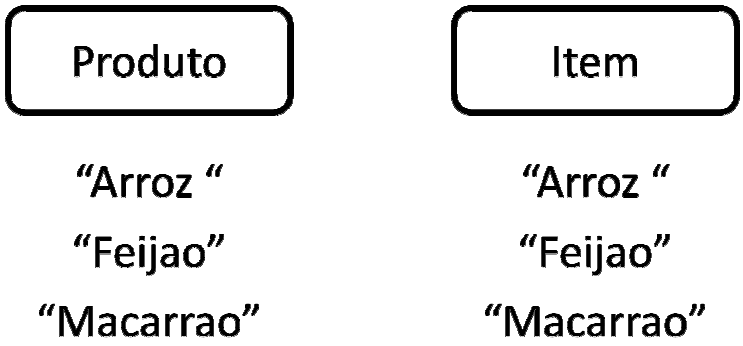

Figura 4. Conjunto de instâncias para dois conceitos

Devido à similaridade entre as instâncias de ***Produto*** e ***Item,*** é possível deduzir que estes conceitos são correspondentes.

Um cenário alternativo consiste do conhecimento sobre as instâncias estarem previamente armazenados. Suponha o cenário da Figura **5** em que ***Produto*** e ***Item*** possuem instâncias em comum com instâncias previamente armazenadas de um conceito ***Mercadoria***. Dessa maneira, é possível inferir uma correspondência entre os conceitos ***Produto*** e ***Item***, através da transitividade de similaridades, visto que ambos os conceitos possuem similaridades entre suas respectivas instâncias e as instâncias de ***Mercadoria***.

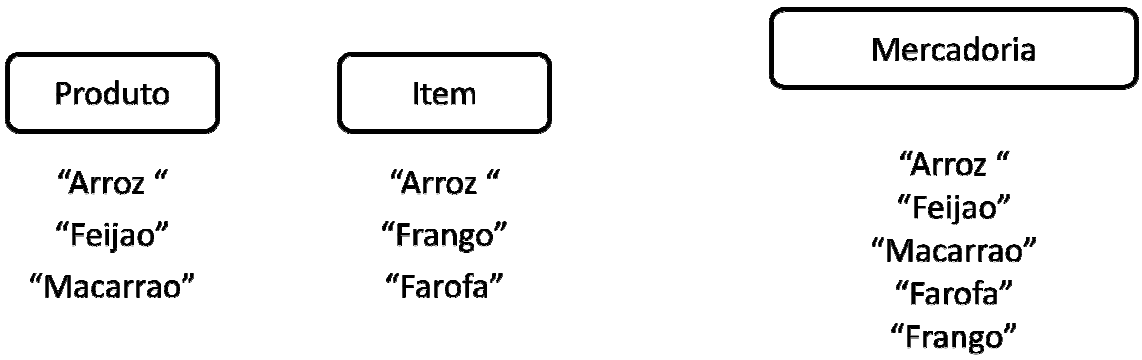

Figura 5. Instâncias previamente armazenadas

Entretanto, existem casos de ontologias que não possuem instâncias ou as mesmas não estão disponíveis. Nesse caso, as técnicas extensionais tornam-se ineficazes.

2.2.5 Uso de elementos externos

Algumas técnicas consistem no uso de recursos externos para auxiliar o processo de *matching*. Tais recursos são passados como entrada para esse tipo de *matcher*. Através destes recursos, o *matcher* é capaz de definir similaridades entre esses conceitos. Como exemplo desses recursos, cita-se:

Dicionários: São repositórios de palavras acompanhadas de uma definição em uma ou mais linguagens naturais (Inglês, Francês, etc.). Para cada palavra, pode haver mais de uma definição. Um tesauro é um tipo especial de dicionário no qual possui informações relacionais de termos com outros. Como exemplos dessas relações citam-se a hiperonímia (generalização), hiponímia (especialização), meronímia (parte de), holônimia (inverso do "parte de") antônimo (oposto) e o sinônimo (equivalência). Logo, suponha dois termos t e t' nos quais se quer definir uma correspondência; pode-se concluir que:

$t \geq t'$: Se t é hiponímia ou metonímia de t'.
$t \leq t'$: Se t é hiperonímia ou holonímia de t'.
$t = t'$: Se t é sinônimo de t'.
$t \neq t'$: Se t é antônimo de t' ou se t e t' não compartilham da mesma relação parte de.

Existem também medidas que estipulam um valor aproximado de similaridade entre dois termos. Como exemplo, cita-se a medida de co-similaridade, na qual se calcula o número de interseções entre os sinônimos dos dois termos, dividido pelo total de termos sinônimos de cada um.

O *WordNet* [51] é um exemplo de tesauro que abrange uma vasta gama de termos, juntamente com sinônimos, hiperônimos, hipônimos além de merônimos (parte de) associados a cada um.

Uma técnica existente é fazer o uso de ontologias de domínio como "base de conhecimento". Dada duas ontologias *o'* e *o"* e uma ontologia de domínio *o*, tal técnica consiste de gerar correspondências de *o'* para *o* e *o"* para *o* (*anchoring*) e em seguida obter correspondências de *o'* para *o"*. Essa derivação de *o'* para *o"* pode ser feita com o auxílio de um motor de inferência.

Por se tratar de um domínio específico, as ontologias de domínio provêem um contexto para a tarefa de gerar as correspondências. Por exemplo, seja uma ontologia de domínio sobre anatomia. Logo, um conceito "cabeça" de uma ontologia-alvo

relacionado a um conceito cabeça da ontologia de domínio (e posteriormente a um conceito da ontologia alvo) refere-se a uma parte de corpo humano, ao invés, por exemplo, de se referir ao chefe de uma empresa. Isso não fica claro quando dicionários são usados, visto que se desconhece qual definição de “cabeça” quer-se referenciar.

Além de dicionários e ontologias de domínio, outros tipos de recursos estão sendo usados para auxiliar o processo de *matching*. Como exemplo, cita-se o trabalho em [16], que utiliza o conhecimento colaborativo da *WikiPedia* como instrumento auxiliar na geração de correspondências.

2.3 Estratégias de *Matching*

As estratégias de *matching* são definidas a partir de uma combinação de uma ou mais técnicas básicas (*matchers*) de *matching*. Uma das ideias principais é tentar aproveitar as características das várias técnicas, visto que o uso de apenas uma delas tende a restringir bastante a qualidade do resultado final. Além disso, o uso de uma combinação de *matchers* permite adotar tratamentos adicionais freqüentemente chamados de tratamentos globais.

Como visto anteriormente, um *matcher* recebe duas ontologias de entrada e fornece como saída um conjunto de correspondências entre as mesmas. Em uma estratégia de *matching*, vários *matchers* são executados, gerando vários alinhamentos. Tais alinhamentos precisam ser agregados a fim de gerar para o usuário um único alinhamento final resultante. Com o objetivo de facilitar a agregação dos alinhamentos provenientes dos *matchers*, Euzenat [21] propõe a conversão de um alinhamento em uma chamada matriz de similaridade, ou seja, cada alinhamento terá uma respectiva matriz de similaridade. Tal matriz é construída da seguinte forma:

1. A matriz de similaridade *M* terá dimensões *n* e *m*, onde *n* é o número de conceitos de uma ontologia *O* (linhas da matriz) e *m* é o número de conceitos de uma outra ontologia *O’* (colunas da matriz) a qual se está efetuando o *matching*.
2. A Matriz *M* é inicializada com zero em todas as suas posições
3. Cada posição ***(i, j)*** será igual ao peso da correspondência entre o elemento associado a *i* e o elemento associado a *j*.

M:

	publication	reviewedarticle	journalarticle
article	0.0	0.75	0.44
reference	0.0	0.11	0.0
journalarticle	0.22	0.66	1.0
journalarticlevip	0.05	0.36	0.66

A:

article	reviewedarticle	0.75	=
article	journalarticle	0.44	=
reference	reviewedarticle	0.11	=
journalarticle	publication	0.22	=
journalarticle	reviewedarticle	0.66	=
journalarticle	journalarticle	1.00	=
journalarticlevip	publication	0.05	=
journalarticlevip	reviewedarticle	0.36	=
journalarticlevip	journalarticle	0.66	=

Figura 6. Matriz de Similaridade *M* e seu respectivo alinhamento *A*.

A Figura **6** exibe um exemplo de uma matriz de similaridade e seu respectivo alinhamento *A*. Neste exemplo foram omitidos os eventuais *id´s* das correspondências, bem como foram consideradas apenas relações de igualdade. Em caso de haver mais de um tipo de relação, pode-se construir uma segunda matriz em que cada posição ***(i, j)*** armazenará o tipo de relação da correspondência.

Dada duas matrizes, uma chamada função de agregação recebe como entrada duas matrizes de similaridade (que podem, por exemplo, representar a saída de dois *matchers*) e retorna uma única matriz de similaridade com os valores agregados. A seção 2.3.2 aborda mais detalhes sobre as funções de agregação.

2.3.1 Formas de combinação

Uma forma natural de combinar *matchers* é em forma seqüencial. A ideia reside em combinar os alinhamentos produzidos em seqüência. Como exemplo, pode-se usar um *matcher* baseado em similaridade entre os nomes dos conceitos, seguido de um baseado na estrutura destes conceitos. CUPID [42] e Falcon-AO [35] são exemplos de sistemas que utilizam uma abordagem seqüencial. Outra forma de combinar *matchers* é em forma paralela, que consiste em executar vários *matchers* independentemente, seguida de uma agregação entre suas saídas. Essa agregação pode ser feita combinando as várias matrizes de uma vez só, diferentemente da forma seqüencial em que a agregação é feita de duas em duas matrizes. Como exemplo de sistemas que utilizam a estratégia paralela, cita-se o COMA [53] e o COMA++ [36]. A Figura 7 mostra a forma geral das abordagens seqüencial e paralela.

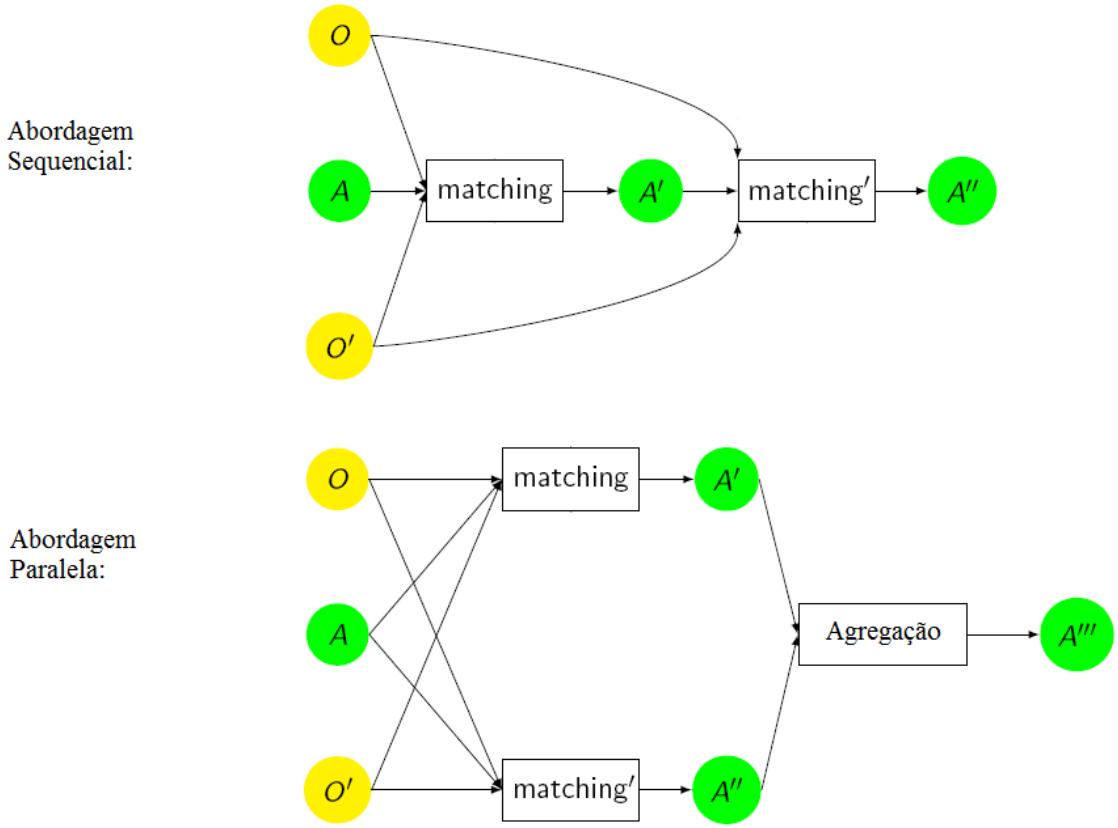

Figura 7. Abordagens Sequencial e Paralela

2.3.2 Medidas de agregação

Como mencionado na Seção 2.3.1, algumas estratégias de *matching* utilizam vários *matchers*, com diferentes características, tendo o objetivo de obter um conjunto de alinhamentos. Tais alinhamentos são então combinados, gerando assim um único alinhamento de saída entre as ontologias. Para efetuar essa combinação, as abordagens utilizam as chamadas medidas de agregação.

Formalmente, uma medida de agregação pode ser então definida como sendo uma fórmula matemática que recebe *k* alinhamentos e retorna apenas um alinhamento, resultado da operação de agregação. Ao usar medidas de agregação, uma estratégia visa tirar o maior proveito possível das características inerentes aos *matchers.*

Um exemplo para o uso de uma medida de agregação dentro de uma abordagem de *matching* seria a combinação entre um alinhamento produzido por um *matcher* sintático (ex: similaridade entre os nomes dos conceitos) e outro alinhamento produzido por um *matcher* estrutural. Dessa forma, a saída da agregação apresentaria um alinhamento mais completo, pois mais características (no caso, sintática e estrutural) foram levadas em consideração [28].

Assim uma combinação de matrizes de similaridades pode ser visualizada como uma operação entre duas matrizes, gerando uma terceira matriz resultante.

Algumas medidas de agregação foram proposta pela comunidade. Entre elas estão MIN/MAX e as medidas ponderadas. Entre as medidas ponderadas cita-se o Produtório Ponderado [21], AVG, HADAPT [29], SIGMOID [30]**,** OWA [31], OPENII [32], NONLINEAR [33]. Peukert [28] exibe uma comparação entre tais abordagens em que são feitos testes utilizando diversas bases de dados. Em seus resultados, as abordagens AVG e HADAPT obtêm os melhores resultados, além de serem consideradas pela avaliação como as medidas mais robustas. Nesse contexto, entende-se por robustez a habilidade da estratégia em retornar os melhores resultados nas diferentes tarefas de *matching.* A seguir, algumas medidas são detalhadas.

Min/Max: Dado dois alinhamentos, a função de agregação MIN sempre escolhe o menor valor entre os pesos das correspondências entre dois elementos. Do ponto de vista da aplicação de um limiar, esta abordagem é considerada pessimista, pois requer a existência de correspondências com alto nível de similaridade (valores altos para os pesos) para que estas sejam consideradas válidas. Considere como exemplo as matrizes de similaridades da Figura 8.

	Book	Translator	Publisher	Writer
Product	.86	.8	.89	.86
Provider	.88	.8	.56	.5
Creator	.86	.5	.89	.57

	Book	Translator	Publisher	Writer
Product	.82	.88	.88	.85
Provider	.83	.89	.76	.71
Creator	.82	.53	.88	.85

Figura 8. Matrizes de similaridades

	Book	Translator	Publisher	Writer
Product	.82	.8	.88	.85
Provider	.83	.8	.56	.5
Creator	.82	.5	.88	.57

Figura 9. Matriz resultante utilizando a função MIN

A Figura **9** exibe a matriz resultante tendo como entradas para a função MIN as matrizes da Figura **8**, ou seja, é uma matriz preenchida com os menores valores dentre

as duas matrizes.

Já a combinação MAX sempre escolhe o maior valor dentre os pesos das correspondências entre dois elementos. Em contrapartida ao MIN, a combinação MAX é considerada otimista, pois basta que apenas um dos valores seja alto o suficiente para conseguir chegar ao resultado final, não importando os demais valores.

Produtório Ponderado: Efetua o produtório entre as raízes das *n* medidas. As raízes são definidas como sendo o peso de cada matriz. Formalmente, tem-se que:

$$\forall x, x' \in o, \delta(x, x') = \prod_{i=1}^{n} \delta(x_i, x_i')^{w_i}$$

Onde *x* e *x'* são conceitos das ontologias envolvidas (linha e coluna de uma matriz o_i), a função sigma (δ) retorna o peso da correspondência entre esses conceitos, e w_i é o peso associado à matriz o_i.

A desvantagem do uso dessa combinação é que se pelo menos uma das *n* medidas for zero, o resultado final será zero. A Figura **10** exibe a matriz resultante da aplicação do produtório, tendo como entrada as matrizes da Figura **8** e pesos de 1/2 para cada uma das duas matrizes.

	Book	Translator	Publisher	Writer
Product	.84	.84	.88	.85
Provider	.85	.84	.65	.60
Creator	.84	.51	.88	.70

Figura 10. Produtório ponderado com $w_1 = w_2 = ½$.

Somas Ponderadas: Dadas *n* matrizes de similaridade, as somas ponderadas efetuam a combinação das matrizes de similaridades na seguinte forma geral:

$$\mathrm{sim}_{\mathrm{agg}}(s,t) = \frac{\sum_{k=1..n} w_k \cdot adj_k(\mathrm{sim}_k(s,t))}{\sum_{k=1..n} w_k}$$

Em que *s* e *t* são os elementos de *o* e *o'* respectivamente, w_k é o peso atribuído para a matriz *k* e adj_k é uma função de ajuste para a matriz *k*. Algumas abordagens como HADAPT, OWA e OPENII preocupam-se em determinar o peso w_k para cada matriz, enquanto que o SIGMOID tem seu foco na função de ajuste.

AVG: Calcula-se a média aritmética entre os pesos das matrizes. É a versão mais simples de uma média ponderada, em que são assumidos pesos iguais para cada matriz, além de se usar um valor identidade (não alterando o valor dos pesos da matriz). A Figura **11** exibe a matriz resultante da média aritmética entre as matrizes da Figura **8**.

	Book	Translator	Publisher	Writer
Product	.84	.84	.88	.85
Provider	.85	.84	.81	.6
Creator	.84	.51	.88	.71

Figura 11. Matriz resultante utilizando a função AVG

HADAPT: Determina os pesos para as matrizes de forma automática. O cálculo dos pesos é baseado em uma medida chamada harmonia (***harmony***) que é computada a partir dos valores de cada matriz, da seguinte forma:

$$h = \frac{\#s_{max}}{\min(n,m)}$$

Em que *h* é o valor harmônico, $\#S_{MAX}$ é a soma das similaridades máximas de cada linha e/ou coluna da matriz, *n* e *m* são as dimensões da matriz e a função ***min*** retorna o menor entre dois valores. Dessa forma, o cálculo da combinação final das matrizes é efetuado da seguinte forma:

$$\frac{\text{sum}(H_i * s_i)}{N}$$

Em que s_i é uma matriz de similaridade, H_i é o valor harmônico associado à matriz S_I, ***sum*** é a função de soma entre matrizes, *N* é o número de matrizes envolvidas e a saída é a matriz resultante da combinação. A Figura **12** mostra a matriz de saída da função HADAPT tendo como entrada as duas matrizes da Figura **8**.

	Book	Translator	Publisher	Writer
Product	.9	.9	.5	.92
Provider	.92	.91	.87	.65
Creator	.9	.55	.95	.76

Figura 12. Matriz resultante utilizando a função *HADAPT*

2.4 Técnicas para melhoria dos resultados

2.4.1 Recorte e Partições em ontologias

Encontrar o conjunto completo de correspondências entre duas ontologias é uma tarefa difícil e que demanda bastante tempo. Essa dificuldade se intensifica quando se lida com grandes ontologias, que envolvem milhares de conceitos. Tal dificuldade se estende inclusive à parte de validação do resultado final.

No intuito de diminuir a complexidade dessa tarefa, algumas técnicas foram propostas. Entre elas, estão o *matching* baseado em partições e o recorte ou filtragem de ontologias.

Rahm [39] define o *matching* baseado em partições como sendo uma estratégia de divisão e conquista que consiste em quebrar as ontologias de entrada, gerando várias partições. Em seguida, efetua-se o *matching* entre as partições geradas de cada ontologia. A ideia é que as partições sejam feitas de forma que cada partição da primeira ontologia "case" com uma partição da segunda ontologia. Isso resulta na redução do espaço de busca, uma vez que as comparações entre os conceitos serão feitas dentro do escopo das partições e não mais no esquema por completo, aumentando a eficiência do processo de *matching*. Além disso, há uma redução também nos requisitos de memória, uma vez que serão executadas várias pequenas tarefas de *matching*, em vez de apenas uma grande tarefa.

Algumas ferramentas como COMA++ [36] e FALCON-AO [35] incluem o *matching* baseado em partições em suas estratégias. No COMA++, o processo é conhecido como *fragment matching* e acontece em quatro passos: (1) Determinação das partições. Neste passo, o usuário é convidado a delinear as partições dos esquemas através de uma interface amigável; (2) Encontrar as similaridades entre as partições. É usado um algoritmo de baixa complexidade para realizar o *matching* entre os fragmentos e encontrar os que possuem mais relações (maior similaridade); (3) Realização do *matching* entre as partições. Uma vez encontrada as partições similares, um algoritmo mais elaborado e refinado é usado para encontrar as correspondências entre as partições similares; (4) As partições e os respectivos *matchings* são então reunidos (*merge* de partições e correspondências).

Resultados de experiências feitas [37] mostraram que a abordagem não apenas

melhora o desempenho do processo, mas também a qualidade dos resultados finais com relação à abordagem convencional de executar o *matching* nos esquemas por inteiro. Porém, ainda existem alguns problemas com essa abordagem. Em primeiro lugar, a definição da similaridade entre as partições não é precisa, pois é baseada nos resultados de um *matcher* de baixa complexidade. Em segundo lugar, definir as partições manualmente exige muito esforço do usuário, em especial para grandes esquemas. Por fim, nem sempre é necessária a definição completa das correspondências para o usuário. Apesar de particionar o esquema, a abordagem descrita não leva em consideração o caráter incremental na geração das correspondências, pois ao final de tudo sempre se visa a definição completa das correspondências para o esquema inteiro.

O Falcon-AO utiliza uma estratégia similar à do COMA++. A ferramenta usa uma estratégia de *cluster estrutural* para iniciar as partições. Após isso, utiliza os chamados *anchors* para determinar a similaridade entre as partições. *Anchors* são correspondências com um alto nível de similaridade (peso próximo a 1) determinadas antes do processo de partição. Tais *anchors* são determinados manualmente ou por um *matcher* de baixa complexidade. A Figura **13** mostra uma ilustração adaptada do processo de *matching* por partição usando *anchors*. As partições de mesma cor são candidatas a serem correspondentes.

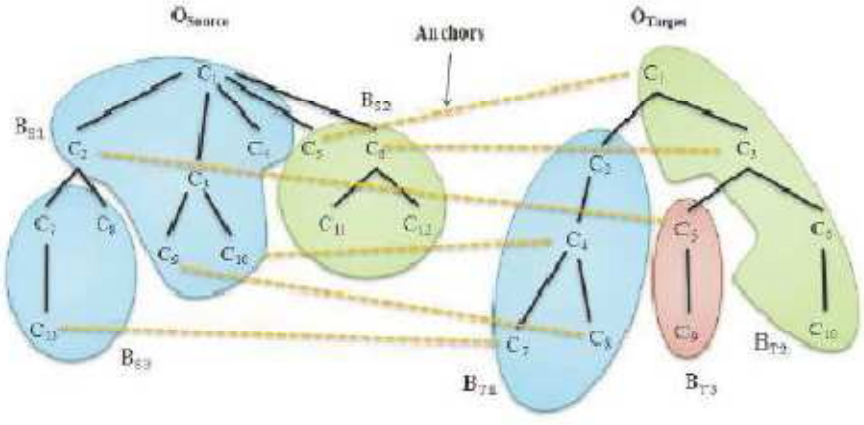

Figura 13. *Matching* por partição usando *anchors* [39]

Zhong [27] preocupa-se com o caso no qual é feito o *matching* entre uma ontologia de pequeno porte com uma de grande porte. Para isso, o autor propõe um particionamento na ontologia grande, tal que esta subontologia “casa” em sua maioria com a ontologia menor. Basicamente o processo segue três passos: (1) Seleciona os

conceitos candidatos a serem selecionados para a construção da subontologia. Tal seleção é baseada no cálculo das similaridades entre as duas ontologias (utiliza-se um *matcher* de baixa complexidade para a execução do cálculo, seguida de um refinamento na seleção consistindo a relevância estrutural dos conceitos); (2) Constrói-se a subontologia de acordo com os conceitos selecionados no passo 1; (3) Gera o conjunto-saída de correspondências entre a ontologia menor e a subontologia construída no passo 2 utilizando um algoritmo refinado.

O problema dessa abordagem é que os resultados das experiências demonstraram que a etapa de seleção de conceitos demanda muito tempo (da ordem de centenas de segundos), enquanto que os passos para se construir a ontologia e executar o *matcher* refinado entre a subontologia e a ontologia pequena demandam bem menos tempo (da ordem de poucos segundos). Isso se deve principalmente ao fato de executar-se um *matcher* nos esquemas inteiros logo no primeiro passo.

Olivier [34] faz um levantamento de métodos para seleção de conceitos para grandes esquemas. Tais métodos são baseados principalmente no número de relações (hierarquias e associações) e atributos de cada conceito para calcular o seu nível de importância dentro do esquema. Há métodos que consideram também a transitividade no cálculo (relações, atributos e características dos conceitos diretamente relacionados ao conceito-alvo). Entretanto, as soluções propostas agem de forma global, calculando assim a importância de todos os conceitos do esquema. Com isso, conceitos que não possuem um bom nível de proximidade entre si podem ser escolhidos como igualmente importantes. Desde que este trabalho objetiva seguir uma filosofia incremental, a ideia é que se obtenha subontologias com conceitos fortemente relacionados uns aos outros, ou seja, com um bom nível de proximidade.

Neste contexto, Olivier [38] sugere uma medida para calcular proximidade entre conceitos. A ideia é considerar o esquema como sendo um grafo onde os nós são conceitos e as arestas relações. Dado um conjunto de conceitos de entrada, usa-se então a definição de distância entre nós no calculo da proximidade entre os conceitos do conjunto-entrada e os demais conceitos do esquema. Essa forma de trabalhar parece ser bem promissora, uma vez que esquemas podem ser incrementalmente contemplados para a execução da operação de *matching*.

2.4.2 Interatividade com o usuário

A maioria das soluções de *matching* existentes calculam o conjunto total de correspondências entre esquemas em uma única iteração (*single-shot*). Devido a esta característica, o resultado final desse processo pode incluir um número considerável de correspondências incertas, principalmente em se tratando de grandes esquemas e bases de dados. Isso motivou a comunidade científica a estudar alternativas com o intuito de aumentar o grau de precisão e revocação dos resultados das soluções de *matchings*.

Uma das alternativas é atribuir o envolvimento do usuário no processo de *matching.* Euzenat e Shvaiko [21] classificam tal envolvimento como sendo de 3 tipos:

1. Prover um alinhamento inicial para os *matchers* usados
2. Combinar *matchers* manualmente
3. *Feedback* de relevância (*Revelance Feedback*)

O provimento de alinhamentos iniciais (1) por parte do usuário restringe a produção do *matcher* fazendo com que as correspondências produzidas sejam de comum acordo com as providas inicialmente. Essa técnica permite ao usuário ter certo controle sobre a saída do algoritmo de *matcher.*

A combinação de *matchers* (2) também pode ser aplicada por usuários. Dessa forma, o usuário pode executar *matchers* com características diferentes de acordo com suas necessidades.

O *feedback* de relevância (3) caracteriza-se, por exemplo, pelo provimento de parâmetros globais dentro da abordagem, como um limiar ou parâmetros de agregação (por exemplo, definir os pesos de uma função de agregação).

Atualmente, algumas abordagens têm dado a oportunidade ao usuário de rejeitar, adicionar e alterar correspondências contidas nos resultados [10, 13, 46, 36]. A esse envolvimento mais intrínseco em que o usuário edita diretamente o resultado, dá-se o nome de *user feedback* (*feedback* do usuário) [15]. O *feedback* do usuário pode ser definido como sendo uma etapa intermediária que acontece durante a execução do sistema, na qual o usuário interage diretamente com o resultado obtido em um determinado momento. Após essa interação, o sistema prossegue com o seu funcionamento automático.

Nesse contexto, [10] introduz o conceito de *matching* incremental entre

esquemas, propondo uma ferramenta na qual as correspondências entre os esquemas são construídas incrementalmente em várias interações, em vez de uma só. Dados dois esquemas como entrada, tal solução consiste de alinhar tais esquemas elemento por elemento, em vez de alinhá-los de uma vez só. Tal abordagem propicia uma economia de tempo e processamento, indo de encontro com a filosofia *pay-as-you-go* [2]. Apesar disso, o fato do alinhamento ser gerado elemento por elemento pode tornar o processo lento do ponto de vista do usuário.

Para que essa construção incremental aconteça de forma desejada, [15] propõe que, a cada interação, haja um *feedback* por parte do usuário acerca do conjunto-saída de correspondências. A justificativa é que o usuário supostamente possui um conhecimento prévio acerca do domínio dos esquemas envolvidos no processo de *matching*. Através desse conhecimento, o usuário vai moldando o conjunto de correspondências candidatas geradas em cada interação, com o intuito de (1) minimizar o número de correspondências falsas; (2) adicionar correspondências que não foram anteriormente identificadas e (3) gerar um alinhamento de entrada para a próxima interação mais condizente com as suas necessidades. Segundo [15], o *feedback* do usuário pode ser classificado de três formas:

- **Confirmação das correspondências corretas:** Dado um elemento de um esquema-fonte e suas correspondências candidatas para elementos de um esquema-alvo, este tipo de *feedback* define qual das correspondências candidatas se relaciona de fato com o elemento em questão, eliminando assim todas as outras candidatas à correspondência. Isso acarreta diretamente na diminuição do nível de incerteza do alinhamento entre os esquemas. Alguns algoritmos também aproveitam esse tipo de *feedback* para recalcular os pesos de correspondências em outros elementos. Considere o exemplo da Figura **14**; suponha uma classe *Acadêmico* que contém as subclasses *Prof. Adjunto* e *Prof. Substituto*. Tal classe possui correspondências candidatas para duas classes em outro esquema: *Acadêmico* e *Funcionário*. Se o usuário confirmar a correspondência candidata 2 para a classe *Funcionário*, um algoritmo de *matching* pode recalcular os pesos das correspondências entre as subclasses das classes em questão provavelmente incrementando tais pesos. Caso o usuário confirme a correspondência candidata 1, o algoritmo provavelmente diminuirá o peso das correspondências entre as subclasses de *Acadêmico* e *Funcionário*.

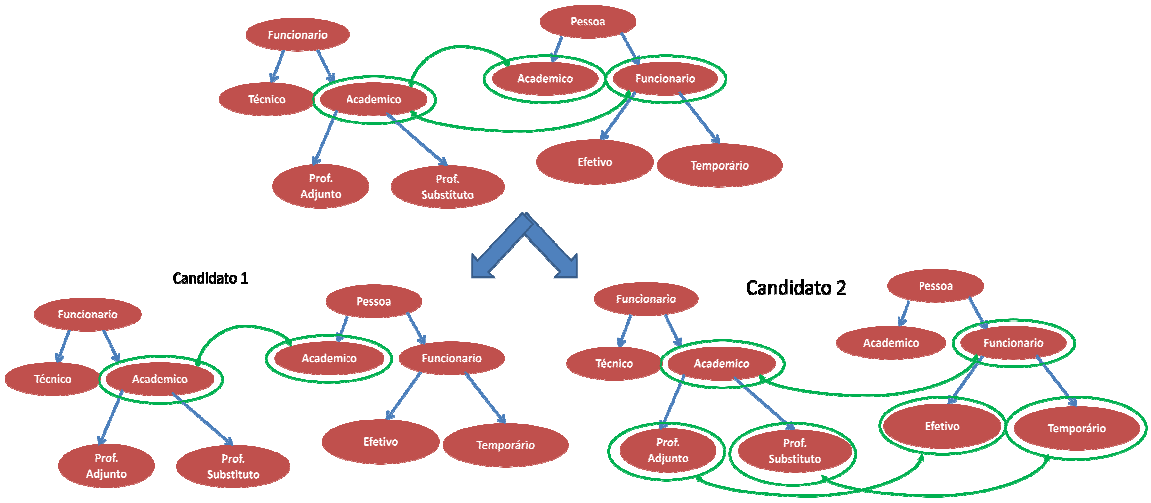

Figura 14. Exemplo de refinamento incremental de correspondências a partir do *feedback* do usuário.

- **Rejeição ou "poda" de falsas correspondências:** A poda de correspondências incorretas diminui a incerteza do alinhamento entre os esquemas. Isso acontece porque uma vez que a informação de uma correspondência foi "podada", o algoritmo de *matching* não fará mais uso da informação em sua próxima computação.

- **Informações extras:** Este tipo de *feedback* é o menos utilizado pelos algoritmos e abordagens existentes. Tal *feedback* consiste, por exemplo, de declarações acerca dos relacionamentos internos entre os elementos de um dos esquemas. Alguns algoritmos aproveitam esse tipo de *feedback* para podar falsas correspondências e aumentar a confiabilidade de outras correspondências candidatas. Outro tipo de informação usado por algumas abordagens são as baseadas em instância. As instâncias podem ser uma grande aliada no processo de alinhamento entre esquemas. Algoritmos podem fazer uso de valores de instâncias, padrões existentes entre instâncias e técnicas de *machine learning* no intuito de gerar novas correspondências ou recalcular correspondências existentes.

A literatura apresenta algumas abordagens de *matching* que fazem uso do *feedback* do usuário. Em [13] é proposta uma ferramenta chamada *Ontrapro*. Tal ferramenta trabalha iterativamente utilizando o algoritmo *similarity flooding* [20] para o cálculo das correspondências candidatas de cada iteração. Ao final da execução do

similarity flooding, o usuário tem a opção de podar as correspondências indesejadas bem como criar novas correspondências manualmente, através de uma interface gráfica amigável. As correspondências que não foram podadas são consideradas automaticamente como confirmadas, evitando assim o esforço do usuário de ter de confirmar cada correspondência. As correspondências podadas são então armazenadas em um banco de dados relacional e são levadas em consideração nas iterações posteriores, de modo a não serem apresentadas novamente em futuras iterações.

Em [1], o *feedback* do usuário é usado no intuito de selecionar o conjunto de correspondências e mapeamentos candidatos entre bases de dados. Dados dois esquemas (fonte e alvo), o sistema recebe um conjunto de *matchings* candidatos. *Mappings* candidatos são criados a partir desses *matchings* candidatos. Uma vez que o usuário interage com o sistema (através de uma consulta, por exemplo), é requisitado que o usuário analise os resultados da interação e julgue os resultados em falso positivos ou tuplas verdadeiras, além de poder informar falsos negativos. Baseado neste *feedback*, o sistema irá eliminando correspondências e mapeamentos candidatos, privilegiando os que geram tuplas verdadeiras. O usuário não é obrigado a analisar todo o conjunto de resultados, mas a ideia é que, quanto maior for a quantidade de resultados analisados, mais precisa será a seleção de correspondências e mapeamentos candidatos (*pay-as-you-go*).

O trabalho descrito em [22] propõe uma abordagem de *machine learning* para o problema do alinhamento entre ontologias. Dadas duas ontologias *O* e *O'*, o usuário especifica pares de conceitos correspondentes iniciais. Tal especificação é armazenada em um arquivo chamado *gold standard matching*. A abordagem então faz uso de métodos léxicos e estruturais para geração de correspondências candidatas bem como o uso de várias métricas para avaliar a similaridade entre os conceitos. A abordagem, então, faz uma comparação entre as correspondências candidatas geradas e as contidas no *gold standard matching* (assume-se que as correspondências contidas nesse arquivo são todas verdadeiras). Essa comparação permite identificar quais métricas de similaridades são mais compatíveis com as necessidades do usuário. Essas métricas são, então, usadas para gerar novas correspondências candidatas. Vale destacar novamente a característica *pay-as-you-go* da abordagem, uma vez que, quanto maior o número de *matchings* informadas no *gold standard matching*, maior será a precisão no calculo de compatibilidade entre as métricas de similaridade. Alguns trabalhos usam o *feedback* do usuário com objetivos que vão além da correção de erros. Como exemplo, cita-se o

trabalho descrito em [7] que aborda, entre outras motivações, a dificuldade da escolha de um limiar adequado para decidir quais correspondências são válidas. Tal trabalho propõe um algoritmo iterativo que define o valor de limiar a partir do *feedback* cedido pelo usuário. Apesar de não gerar bons resultados para todos os cenários, tal algoritmo se prova eficaz na maioria dos casos.

De um modo geral, os sistemas que utilizam interatividade com o usuário possuem uma etapa intermediária em que tal usuário pode manipular o resultado, parâmetros e outros dados críticos com o intuito de melhorar a qualidade do resultado final do processo de *matching*. Esse é considerado um importante mecanismo que auxilia o *matching* incremental entre esquemas, sendo cada vez mais usado em trabalhos e abordagens atuais.

Capítulo 3

I3M – *Matching* Interativo, Iterativo e Incremental

Neste capítulo, apresentamos a abordagem I3M (*Incremental, Interative, Iterative Matching*) para a realização do *matching* entre ontologias. Inicialmente, apresentamos algumas definições que são relevantes para a compreensão da proposta. Em seguida, é apresentada a ideia principal bem como os algoritmos envolvidos. Para validar a abordagem, foi desenvolvido um protótipo utilizando tecnologias relevantes, largamente utilizadas pela comunidade acadêmica. Os detalhes de implementação também são discutidos neste capítulo, que está organizado como segue: a Seção 3.1 aborda as definições usadas no entendimento do trabalho; a Seção 3.2 introduz a abordagem proposta; a Seção 3.3 exibe os trabalhos relacionados; a Seção 3.4 apresenta o protótipo desenvolvido a partir da ideia concebida na Seção 3.2; e a Seção 3.5 apresenta um exemplo ilustrativo do uso da abordagem e ferramenta propostas.

3.1 Definições

A necessidade de se definir o conjunto de correspondências (ou alinhamento) entre duas ontologias surge em diversas aplicações. Uma correspondência pode ser definida como uma relação entre dois conceitos de duas ontologias distintas. Um conceito pode ser visto como um elemento pertencente a uma ontologia, como uma classe, propriedade ou instância. Trabalhos existentes na área de *matching* representam uma correspondência em forma de tupla, em que o formato da mesma varia de abordagem para abordagem. Para este trabalho, serão usadas as seguintes definições para ontologia (adaptada de [21]) e correspondência, respectivamente [21]:

Definição 1 (*Ontologia*): Uma ontologia é uma tupla $O = <C, I, R, P, T, V, E, \leq, \perp, \mu, \beta, \alpha, \equiv>$ onde:

C é o conjunto de classes (ou conceitos) contidas na ontologia ***O***;

I é o conjunto de indivíduos contidos na ontologia ***O***;

R é o conjunto de relações contidas na ontologia ***O***;

P é o conjunto de atributos contidos na ontologia ***O***;

T é o conjunto de tipos de dados contidos na ontologia ***O***;

V é o conjunto de valores de dados (sendo ***C, I, R, P, T, V*** disjuntos entre si) contidos na ontologia ***O***;

E é o conjunto de restrições contidas na ontologia ***O***;

$\leq$ é a relação sobre (C x C) U (R x R) U (P x P) U (T x T) chamada de *especialização*;

$\perp$ é a relação sobre (C x C) U (R x R) U (P x P) U (T x T) chamada de *exclusão;*

$\boldsymbol{\mu}$ é a relação sobre (I x C) chamada *instanciação;*

$\boldsymbol{\beta}$ é a relação sobre (C x P) U (C x R x C) U (P x T), chamada *atribuição;*

$\boldsymbol{\alpha}$ é a relação sobre (R x E) U (P x E) chamada *restrição;*

$\equiv$ é a relação sobre (I x R x I) U (I x P x V) chamada *nomeação;*

Definição 2 (Correspondência): Uma correspondência é uma quíntupla <***id, e, e', r, n***> em que:

- ***id***: Identificador único da correspondência;
- ***e***: conceito pertencente a uma ontologia fonte *O*;
- ***e'***: conceito pertencente a uma ontologia alvo *O'*;
- ***r***: tipo específico de relação (p.ex.: equivalência, subsunção);
- ***n***: número real compreendido entre o intervalo [0,1] que define o grau de confiança da correspondência. Os valores 0 e 1 determinam respectivamente a similaridade mínima e máxima entre os conceitos. Em uma correspondência com grau de confiança 0,7, pode-se dizer que a correspondência em questão é 70% confiável de ser verdadeira.

Um exemplo simples de correspondência é descrito abaixo:

<1, reviewedarticle, article =, 0,63>

Na qual o conceito ***reviewedarticle*** da ontologia fonte é equivalente ao conceito ***article*** da ontologia alvo, com grau de confiança 0,63. Um conjunto de correspondências entre duas ontologias caracteriza um alinhamento entre as mesmas.

As duas definições seguintes são de suma importância para o entendimento da abordagem, uma vez que a filtragem de esquemas é feita baseada nelas:

Definição 3 (*Proximidade*): Dados os conceitos ***e1*** e ***e2*** de uma ontologia ***O***, a função de proximidade entre ***e1*** e ***e2***, denotada por *prox*(***e1***,***e2***), pode ser definida como o número mínimo de relações entre conceitos que interligam estes conceitos. Esta definição é semelhante à definição de distância entre nós em teoria dos grafos. Por

exemplo, suponha que a Figura **15** ilustra uma ontologia, em que os nós são conceitos e arcos são relacionamentos. O valor de proximidade entre conceitos *A* e *B* é 1 (*prox*(*A*,*B*) = 1), pois existe um relacionamento direto entre *A* e *B*. No entanto, a proximidade entre conceitos *C* e *G* é 4, porque o caminho mínimo entre esses dois nós consiste de quatro relações (Arcos).

Definição 4 ***(Partição de ontologia):*** Uma partição de uma ontologia *O* pode ser definida como uma subontologia *S*, derivada de *O*. Os conceitos destacados, bem como as relações que os conectam, apresentados na Figura **15**, são um exemplo de uma partição de ontologia.

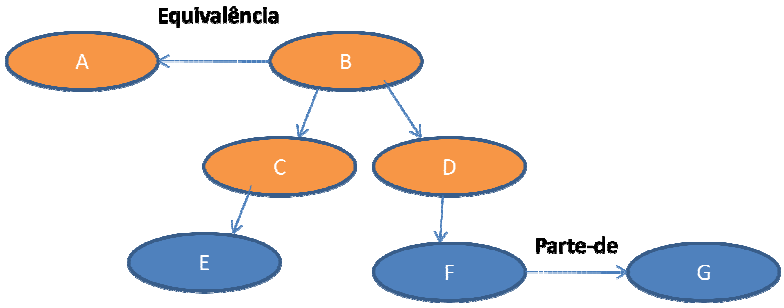

Figura 15. Exemplo ilustrativo de uma ontologia com conceitos destacados em laranja

3.2 Abordagem Proposta

A ideia deste trabalho é estender o processo tradicional de *matching* (*single-shot*), tornando-o um processo com as seguintes propriedades:

- Incremental: Em cada iteração, tal processo trabalha com um subconjunto da ontologia-alvo ao invés de toda a ontologia, podendo ampliar o alcance das correspondências de acordo com os interesses do usuário
- Interativo (*feedback* do usuário): Diferentemente das abordagens de *matching single-shot*, há etapas intermediárias em que o usuário pode interagir com o sistema, rejeitando correspondências candidatas e criando correspondências ausentes, aumentando a precisão e a revocação dos resultados.
- Iterativo: A abordagem é baseada em dois tipos de iteração. O primeiro tipo é relacionado ao tamanho do problema que o usuário decide tratar (iteração quantitativa). Já o segundo tipo está relacionado ao esforço do usuário em refinar determinada parte do problema (iteração qualitativa). Os resultados

computados em uma iteração são armazenados, de forma a serem reaproveitados em iterações seguintes.

No quadro abaixo é apresentado o processo proposto sob forma de um algoritmo. Dessa forma, uma descrição intuitiva do processo pode ser feita, na qual se destaca um laço mais externo como sendo o quantitativo (controla o tamanho das sub-ontologias a serem alinhadas) e o mais interno como sendo o qualitativo (controla o esforço do usuário em alinhar a subontologia em questão).

- Enquanto usuário julgar necessário **(incremental/quantitativo)**
  - Usuário provê informações baseado nas ontologias de entrada;
  - As ontologias são particionadas em função das entradas recebidas;
  - Enquanto o usuário julgar necessário, faça: **(incremental/qualitativo)**
    - Recupere o conjunto de correspondências válidas *A* e rejeitadas *R*, envolvendo os conceitos.
    - Escolha um matcher *M* que recebe *A* e *R* como entradas e executa o matching entre as subontologias, gerando *A'*
    - O usuário provê feedback sobre *A'*
    - Armazene o resultado do *matching*
    - Verificar continuação da iteração.
- Mostrar resultado final ao usuário

A Figura **16** ilustra o processo proposto para geração incremental de correspondências, o qual é descrito a seguir. Intuitivamente, o processo proposto para alinhamento de ontologias de forma incremental e interativa segue os seguintes passos:

1. ***Filtrar Ontologias fonte e alvo***: dada uma ontologia fonte e uma ontologia alvo a serem alinhadas, o usuário inicialmente seleciona um conceito de cada uma das ontologias fonte e alvo, o qual será usado para executar o recorte nas ontologias. O sistema irá recortar ambas as ontologias de acordo com os conceitos iniciais especificados;

2. ***Realizar Matching entre recortes***: Os recortes criados no passo anterior serão utilizados como entrada para a realização de *matching* entre as ontologias. O processo de *matching* incluirá diversos algoritmos que poderão ser selecionados pelo usuário e contará com o suporte de um banco de dados com informações sobre correspondências já descobertas de possíveis execuções anteriores, dado que este processo pode possuir diversas interações. O resultado deste passo é um novo conjunto de correspondências;

3. ***Receber Feedback do Usuário***: O resultado do *matching* é apresentado ao usuário para sua aceitação ou rejeição. Esses feedbacks são armazenados para serem usados nas próximas interações;

Os passos 2 e 3 são repetidos até que o usuário decida parar de interagir com o sistema. Em uma próxima interação, novos conjuntos-entrada serão submetidos, novas sub-ontologias formadas e novos alinhamentos gerados e refinados.

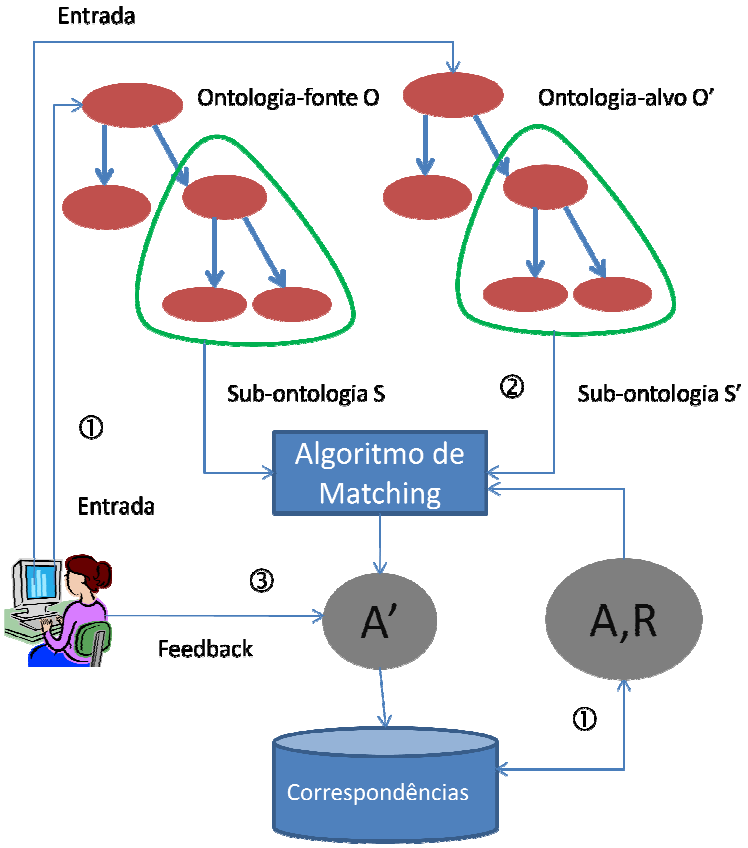

Figura 16. Visão geral do processo de geração/refinamento de correspondências

Como uma visão alternativa do processo, a **Figura 17** ilustra por meio de um *workflow* a sequência de atividades executadas e artefatos obtidos. A seguir, a descrição detalhada, passo-a-passo do processo.

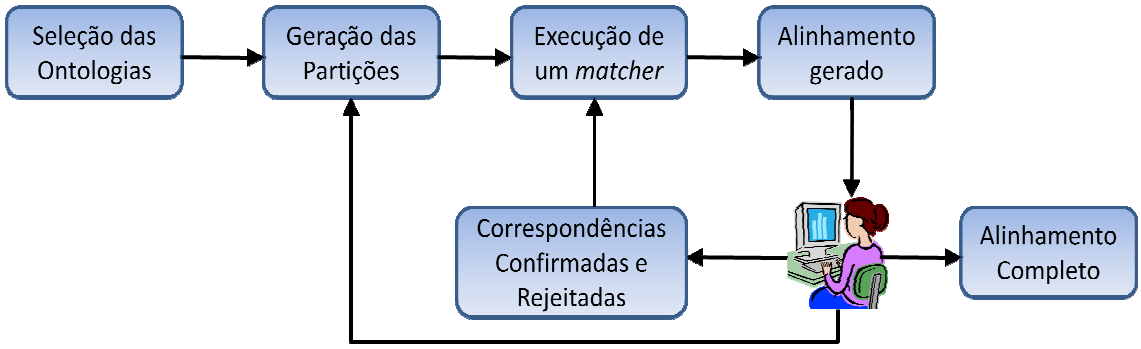

Figura 17. Visão alternativa do processo

Passo 1 - Filtragem das Ontologias fonte e alvo - Figura 16 **(1)**

Inicialmente, o usuário gera um conjunto-entrada para cada uma das duas ontologias que deseja filtrar. Cada conjunto-entrada consiste de um nome de um conceito pertencente a ontologia em questão, seguido de um inteiro positivo *k*. A abordagem então fará um "recorte" em cada uma das ontologias, selecionando assim um subconjunto destas ontologias na qual o *matching* será efetuado. Cada subconjunto será construído com base no conceito contido nos conjuntos-entrada, bem como nos conceitos e propriedades relacionadas a estes termos, de acordo com o algoritmo apresentado abaixo:

Algoritmo: SelecionaConceitos
Entrada: Ontologia **O**, conceito **c** ∈ **O(C)**, inteiro **k**.
Saída: Fila de conceitos **F**

Criar duas Filas de conceitos **F** e **FAUX**;
Criar duas Filas de inteiros **D** and **DAUX**;
Inteiro **dist** = 0;

Inserir **c** em **F** e **Faux**;
Inserir **dist** em **D** e **Daux**;

Enquanto (**dist** < **k** e **FAUX** não está vazia)
  Conceito **c'** = Remover e retornar elemento de **FAUX**;
  Inteiro **dist'** = Remover e retornar elemento de **DAUX**;

```
  Para cada conceito c'' ∈ O(C) tal que prox(c',c'') = 1
    Se F não contém c'' e dist' < k
       Inserir c'' em F e Faux;
       Inserir dist' em D e Daux;
       dist'++;
    Fim Se
    dist = dist';
  Fim Para
Fim Enquanto
Retornar F;
```

A principal ideia do algoritmo *SelecionaConceitos* é usar filas para controlar os conceitos que serão selecionados. As entradas do algoritmo são: uma ontologia *O* (a ser filtrada), um conceito *c* pertencente a *O,* e um inteiro *k* (que delimita o número de conceitos capturados), passados pelo usuário. A Fila *F* de conceitos é a saída final do algoritmo e *FAUX* é uma fila auxiliar para armazenar e fornecer os conceitos que serão colocados em *F*. As filas *D* e *DAUX* são semelhantes às filas *F* e *FAUX*, porém, *D* e *DAUX* armazenam as distâncias (número de relações que separam os conceitos em *F* e *FAUX*, respectivamente, do conceito *c* passado como entrada).

Em primeiro lugar, o conceito de entrada *c* é colocado em *F* e *FAUX*. A distância inicial zero é enfileirada em *D* e *DAUX*. Assim, o algoritmo preenche a fila *F* com todos os conceitos adjacentes (desenfileirando-os de *FAUX*) até que se atinja o número *k* de conceitos estabelecido, ou não haja mais conceitos a serem verificados. A tabela abaixo mostra as filas *F* e *D* após a execução, tendo como entradas a ontologia da **Figura 15**, conceito B e *k* = 4, como entrada.

Queue F	B	A	C	D
Queue D	0	1	2	3

Uma questão que surge na abordagem proposta para a seleção dos conceitos de uma ontologia é o "empate" no critério de captura destes conceitos. Não é adequado utilizar apenas a definição de proximidade entre conceitos, pois pode haver o caso em que dois conceitos tenham a mesma proximidade de um conceito inicial e apenas um destes deve ser escolhido. Uma vez que em uma ontologia, há diversos tipos de relação entre conceitos (equivalência, subsunção, etc.), estabeleceu-se uma seqüência

de prioridade. Tal sequência constitui-se de: (i) relações de equivalência (ii) subsunção (iii) outras (por exemplo: parte-de). Como exemplo, seja o conceito F e *k=2* como entrada. Para este caso, e sem estabelecer uma sequência de prioridades para os relacionamentos, há uma incerteza de qual conceito será capturado (D ou G). Com a sequência estabelecida, o conceito D seria selecionado, pois o relacionamento de subsunção possui maior prioridade que o de meronímia (parte-de).

Apesar da sequência estabelecida, ainda há casos de incerteza ao selecionar um conceito. Suponha por exemplo, uma entrada de $k = 3$ e o conceito B. Neste caso, não se saberia determinar quais dos conceitos (C ou D) seriam armazenados na fila de saída, pois ambos os conceitos são subclasses de B e apenas um deles deve ser selecionado. Sabe-se que ontologias são serializadas em arquivos (por ex: arquivos *owl*). A solução utilizada foi capturar os conceitos em ordem de declaração no arquivo.

Após a seleção dos conceitos, outras propriedades e relações que os envolvem são capturadas para serem adicionados na partição resultante. O algoritmo a seguir mostra como fazer isso:

Algoritmo: CriarSubontologia
Entrada: Ontologia **O**, Fila de conceito **F**
Saída: Sub-ontologia **S**

Criar uma ontologia vazia **S**;
Para cada conceito **c** em **F**
  Inserir **c** em **S**;
  Inserir em **S** todas as instâncias **i** de **c** ∈ **O(I)** e axiomas de instanciação a partir de **O(μ)**;
  Inserir em **S** todas as propriedades de dados **p** ∈ **O(P)** e respectivos axiomas, tal que **p** envolve **c** a partir de **O(β)**.
  Inserir em **S** todas as restrições **e** ∈ **O(E)** e respectivos axiomas, tal que **e** envolve **p,** a partir de **O(α)**
  Inserir em **S** todos os tipos de dados **t** ∈ **O(T)** e respectivos axiomas, tal que **t** envolve **p** a partir de **O(β)**

  Para cada conceito **c'** em **F**
    Inserir em **S** todos os axiomas que envolvam **c** e **c'** a partir de **O(≤), O(⊥)**
    Inserir em **S** todas as relações **r** ∈ **O(R)** e respectivos axiomas, tal que **r** envolve **c** e **c'** a partir de **O(β).**
    Inserir em **S** todos os axiomas que envolvem **i** ∈ **O(I)** tal que **i** é instância de **c** em **O(μ)**, **p** ∈ **O(P)** tal que **p** envolve **c** a partir de **O(β)** e **v** ∈ **O(V),** a partir de **O(≡).**
    Inserir em **S** todos os axiomas que envolvem **i** e **i'** ∈ **O(I)** tal que **i** é instância de **c**

e **i'** é instância de **c'** em **O(μ)**, e **r** ∈ **O(R),** a partir de **O(≡).**
Fim Para
Fim Para
Retorne **S**;

O algoritmo acima é responsável por capturar todos os axiomas, propriedades, restrições e instâncias relacionadas aos conceitos selecionados previamente, contidos na fila de conceitos F. Por exemplo, o comando "Inserir em **S** todas as propriedades de dados **p** ∈ **O(P)** e respectivos axiomas, tal que **p** envolve **c** a partir de **O(β)**" diz que deve-se inserir na subontologia todas as propriedades de dados e axiomas relativos a esta, contidas na ontologia *O* tal que essa propriedade envolve (está relacionada a) um conceito *c*, ou seja, tem um conceito *c* como domínio. Em resumo, todas as propriedades, restrições e axiomas que possuem relação com um conceito *c* contido em *F*, farão parte da subontologia *S*.

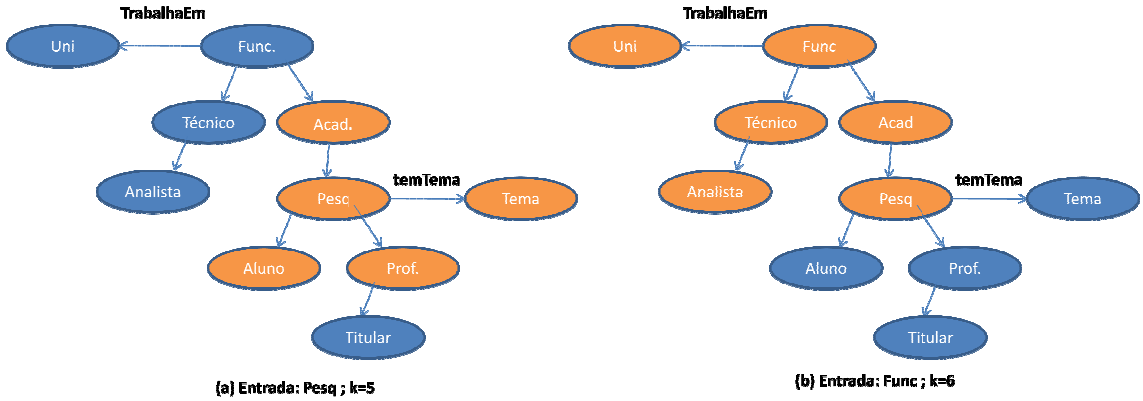

Figura 18. Exemplo da seleção de conceitos

Paralelo a etapa de recorte em cada ontologia de entrada, a abordagem recupera automaticamente as correspondências relacionadas aos termos do conjunto-entrada, contidas em um banco de dados. São recuperados dois alinhamentos: o primeiro deles contém as correspondências rejeitadas (*R*), ou seja, aquelas nas quais o usuário indicou serem correspondências incorretas. O segundo alinhamento contém as correspondências confirmadas (*A*), definidas de iterações anteriores.

Passo 2 – Realizar o *Matching* sobre recortes

As correspondências contidas em ambos os alinhamentos (*A* e *R*) serão passadas ao *matcher*. O alinhamento *R* servirá de base para evitar repetição de sugestões de

correspondência em futuras iterações, evitando assim redundância de rejeições. As correspondências confirmadas farão parte do alinhamento inicial *A*, que irá ser passado ao *matcher* usado. No caso de não haver correspondências relacionadas com os termos de entrada, ou de ser a primeira vez a se trabalhar com as ontologias em questão, o resultado das consultas, bem como os alinhamentos iniciais *A* e *R* serão vazios.

O usuário escolhe o *matcher M* entre os disponíveis pela abordagem para se trabalhar naquela iteração qualitativa. O *matcher M* receberá como entrada os alinhamentos *A* e *R*, e os subconjuntos *S* e *S'* oriundos de *O* e *O'* respectivamente. A saída do processamento será um novo alinhamento *A'* tal que:

$$A' = ref\,(A, \text{resultado}(M) - R)$$

Em que *ref* é uma função de agregação que retornará um conjunto de correspondências em que os pesos estão devidamente ajustados, de acordo com uma função matemática $f : R \times R \rightarrow R$ onde *R* é o conjunto dos números reais. O algoritmo abaixo aborda como é feito esse processo de refinamento.

Algoritmo de Refinamento
Entrada: Alinhamento M (gerado pelo *matcher*), Alinhamentos A (confirmados) e R (rejeitados) relacionados.
Saída: Novo Alinhamento A'

Para cada correspondência **m** em M
  Se R possui uma correspondência **r** tal que **r**.e == **m**.e e **r**.e' == **m**.e' e **r**.r == **m**.r, então
    M = M – **m**
  Fim Se
Fim Para

Criar alinhamento vazio A'
Para cada correspondência **m** contida em M, faça:
  Se A possui uma correspondência **a** tal que **a**.e == **m**.e e **a**.e' == **m**.e', então
    Criar correspondência **a'** tal que:
    a'.id = criar_id();
    a'.e = **a**.e
    a'.e' = **a**.e'
    a'.p = (**a**.p + **m**.p)/2
    Se **a**.r E **m**.r são relações de equivalência , então
      a'.r será uma relação de equivalência
    Senão
      a'.r será uma relação de subsunção
    Fim Se

```
    A' = A' U a'
  Senão
    A' = A' U m
  Fim Se
Fim Para
Retorne A'
```

Nesta etapa, o alinhamento *M* sofre exclusões, que consistem em eliminar todas as correspondências de *M* que também estão em *R*. Após isso, ocorre o processo de refinamento, em que há uma agregação do alinhamento inicial A com o alinhamento gerado por *M*. Nessa agregação, a função ***cria_id()*** é utilizada para gerar identificadores únicos para as novas correspondências. Essa geração envolve consultas por *id*'s já existentes, no intuito de evitar choques de identificadores.

A função utilizada para agregar pesos foi a *AVG* devido à simplicidade de implementação e a igualdade de pesos e prioridades para cada matriz de similaridade (ver seção 2.3.2), mas outra poderia ter sido escolhida. Ocorre também a agregação entre os tipos de relacionamento das correspondências, em que o operador de subsunção, tem maior precedência em relação ao operador de equivalência. Além disso, o operador de equivalência pode ser definido em função do operador de subsunção (sejam dois conceitos *x* e *y*. Se *x* equivale a *y*, isso quer dizer que *x* subsume *y* e *y* subsume *x*). A agregação é feita para todas as correspondências comuns de *M* e *A*. Vale lembrar que, neste trabalho, só é levado em consideração dois tipos de relacionamento em correspondências: equivalência e subsunção. As demais correspondências de *M*, que não são comuns com *A*, entram no alinhamento de saída *A'* sem sofrerem nenhuma alteração.

Passo 3 – Receber Feedback do Usuário

Neste passo, o alinhamento *A'* é exibido ao usuário. O usuário então poderá rejeitar correspondências indesejadas bem como criar correspondências não geradas por *M* ou não encontradas em *A'*, a fim de aumentar a qualidade dos resultados. Todas as correspondências assim criadas terão peso 1.0, bem como as rejeitadas terão seus pesos ajustados para 0. As correspondências assim rejeitadas são adicionadas à *R*, e as criadas são adicionadas a *A'*, que por sua vez são armazenadas em um banco de dados. O algoritmo abaixo é responsável por filtrar as correspondências existentes em um banco de dados, de acordo com as subontologias vigentes em determinada interação. O procedimento é executado tanto para a tabela de confirmados como para a tabela de

rejeitados.

Algoritmo SelecionaCorrespondencias
Entradas: Subontologias *O* e *O'*, Alinhamento Completo *A* da Tabela *T*
Saída: Alinhamento filtrado *F*, oriundo de *T*

Para cada correspondência c ∈ *A*
  Se c.e está contido em *O* e c.e' está contido em *O'*
    Insira *c* em *F*
  Fim Se
Fim Para
Retorne *F*

3.3 Trabalhos relacionados

A literatura apresenta diversos trabalhos que abordam questões envolvidas nesta proposta. O trabalho descrito em [10] introduz o conceito de *matching* incremental, tratando-se do BizTalk Mapper, um protótipo que recebe dois esquemas XML, dispõe os mesmos e envolve o usuário em um processo de *matching* em que o mesmo é feito elemento por elemento. Ou seja, para cada elemento de um esquema, a ferramenta utiliza um algoritmo de *matching* para sugerir correspondências candidatas para o mesmo. O usuário então é apto a confirmar (ou criar) correspondências para cada elemento. Apesar de toda a preocupação dos autores com a interface e usabilidade da ferramenta, o fato de dispor todo o esquema e de se trabalhar em cada elemento por vez pode tornar o processo lento e desgastante para o usuário, especialmente em se tratando de grandes esquemas.

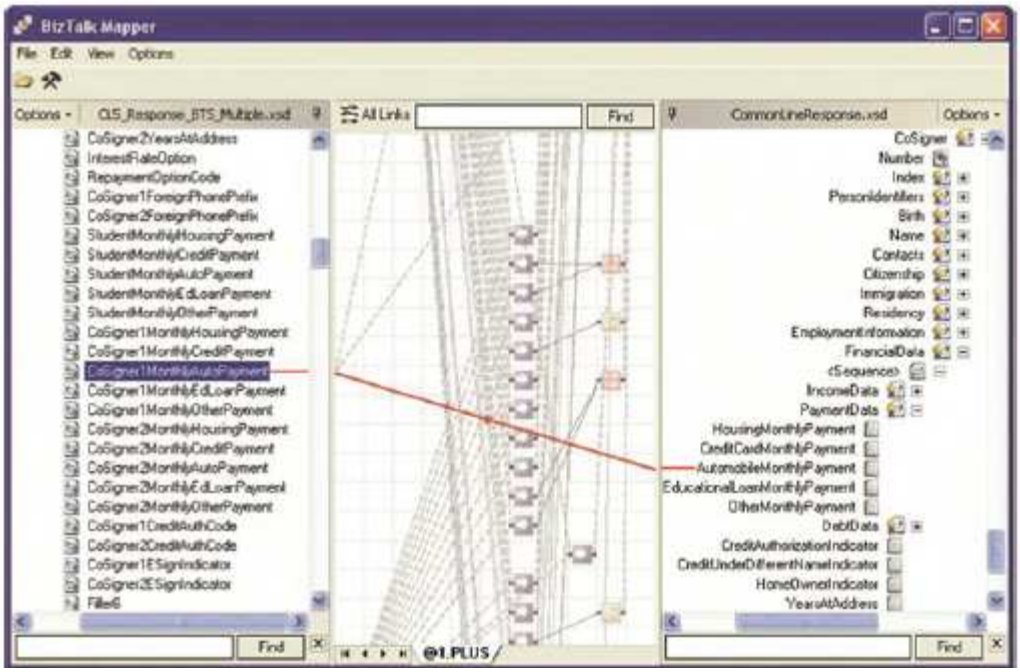

Figura 19. Interface do BizTalk Mapper [10]

Danny Chen et al [13] propõem uma ferramenta de *matching* chamada *OntraPro*, que possui uma interface simples e de fácil entendimento. Tal ferramenta utiliza o algoritmo iterativo *Similarity Flooding* [20] para calcular as correspondências entre ontologias. Após o cálculo das correspondências, o usuário está apto a podar as correspondências rejeitadas e a criar correspondências não identificadas anteriormente. Todos os resultados são armazenados em um BDR. Porém a abordagem também não trata a questão da filtragem de informações e possui problemas caso o usuário não efetue a correta eliminação das falsas correspondências.

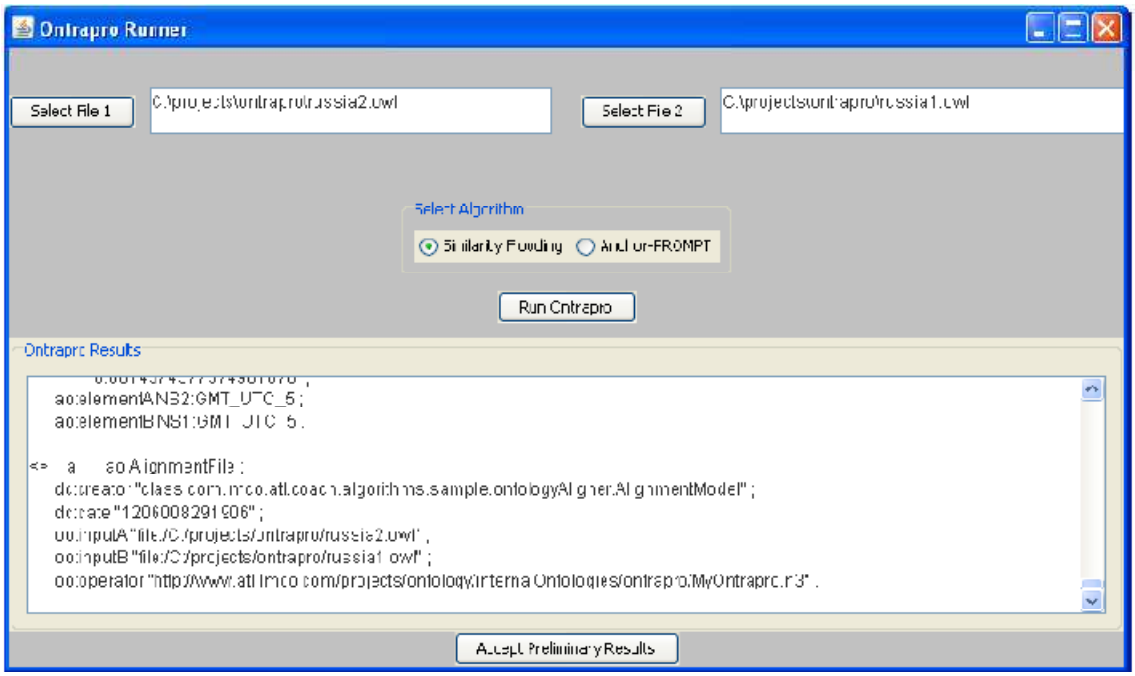

Figura 20. Interface de entrada do OntraPro [13]

O PROMPT [46] é uma ferramenta semi-automática que provê suporte para as tarefas de *matching* e *merge* entre ontologias. O PROMPT possui um *plugin* opcional chamado COGZ, que auxilia o usuário cognitivamente ao lidar com grandes ontologias. Além de recursos visuais, COGZ possui um mecanismo de filtragem para termos de pesquisa, de forma a destacar os termos relevantes da ontologia e ocultar os irrelevantes. Esse mecanismo aumenta o foco do usuário em determinadas partes das ontologias, reduzindo as ocorrências de erro. No entanto, o mecanismo filtra as ontologias apenas visualmente, para fins de usabilidade. Algoritmos automáticos disponíveis no PROMPT não consideram a filtragem estrutural, gerando resultados de menor qualidade.

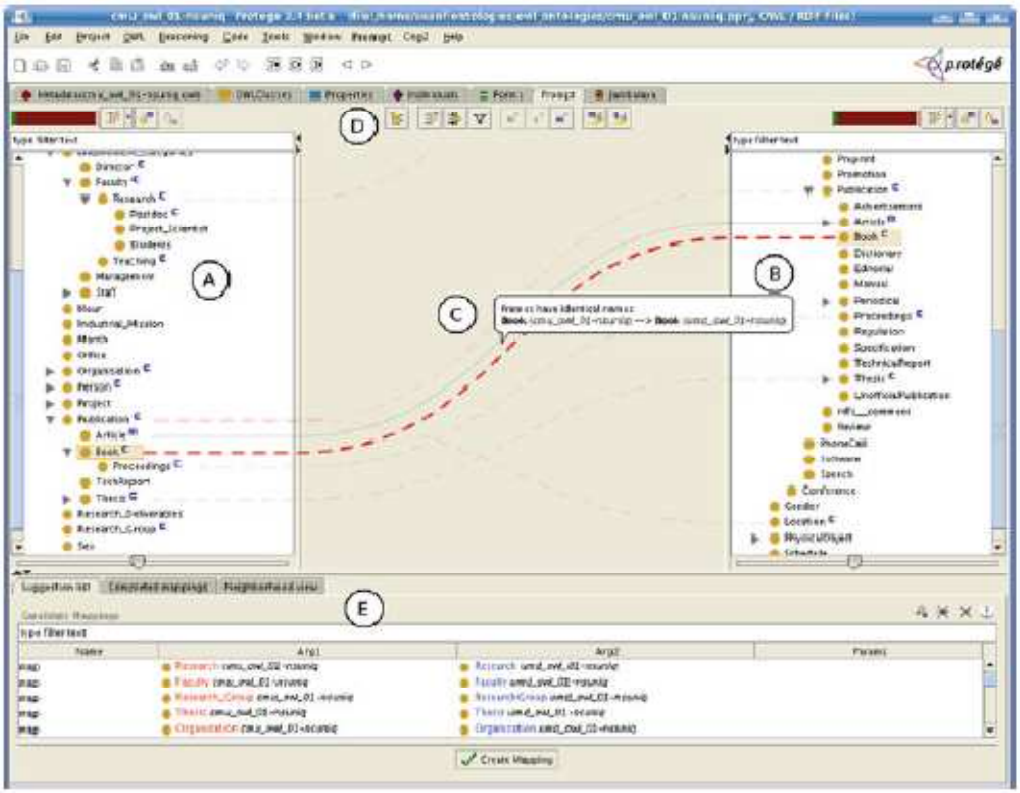

Figura 21. Interface do PROMPT+COGZ [46]

Além do PROMPT, o COMA++ [36] tornou-se uma ferramenta semi-automática bastante utilizada para a geração de correspondências entre esquemas. O COMA++ contém vários *matchers*, no objetivo de sugerir correspondências candidatas ao usuário, que deve prover seu *feedback* e gerar o alinhamento resultante. Como vantagens do COMA++, citam-se a capacidade de lidar com várias tecnologias de modelagem de dados como XML, OWL, SQL, etc., além da interface gráfica interativa, incluindo a funcionalidade de desenhar correspondências. O COMA++ não trabalha com partições, o que dificulta a execução da tarefa de *matching* para grandes esquemas.

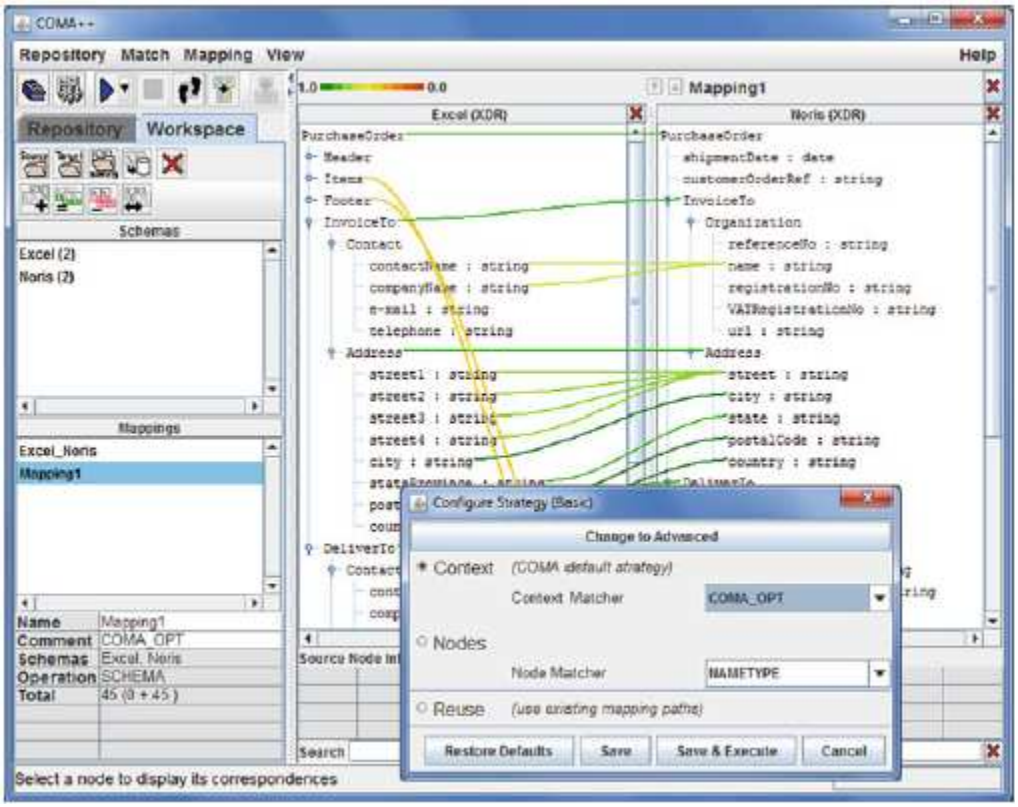

Figura 22. Interface do COMA++ [36]

O trabalho desenvolvido por Burgess em [25] propõe uma abordagem para criar interligações entre bases de dados contidas no *Linked Open Data Project* na qual o usuário é a entidade mais fundamental e importante dentro do processo. Como conseqüência disso, questões como interface e usabilidade são o foco principal do trabalho. As experiências são feitas considerando diferentes perfis de usuário e consistem não somente de métricas e avaliações convencionais, mas também de um questionário cognitivo e de gráficos comparativos que envolvem a experiência e desempenho de cada usuário. Infelizmente, além da abordagem contemplar apenas a definição de correspondências em nível de instância, o usuário tem que definir as correspondências de forma manual, sem o auxílio de nenhum algoritmo auxiliar de *matchings*, o que compromete o desempenho do processo em especial para grandes bases de dados.

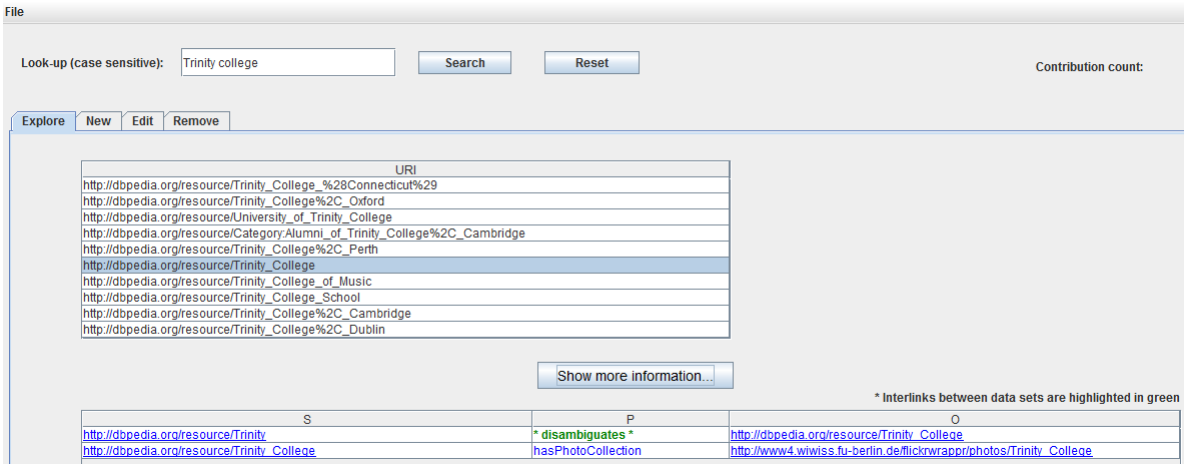

Figura 23. Interface do protótipo desenvolvido em [25]

3.4 Protótipo Desenvolvido

Nesta seção apresentamos um protótipo que implementa a abordagem I3M, que implementa o processo de *matching* incremental, iterativo e interativo descrito na seção 3.2. O protótipo foi implementado em Java, utilizando OWLAPI para manipulação das ontologias de entrada, sendo aceitas ontologias descritas em OWL. Além disso, utiliza-se a *Alignment* API [23] para manipulação de alinhamentos e provimento de um ambiente para implementação de aplicações voltadas à resolução do problema de *matching* entre ontologias.

Um dos requisitos da abordagem proposta é o armazenamento de correspondências confirmadas e rejeitadas pelo usuário, a fim de auxiliar o processo. Para esta tarefa, um esquema lógico foi implementado utilizando o banco de dados relacional Oracle. A Figura **24** apresenta a modelagem lógica utilizada no projeto de banco de dados. A tabela "*TblMatching*" registra as URI´s das ontologias passadas como entrada em uma determinada interação. Assim, cada tupla representa uma instância de uma tarefa de *matching*, ou seja, representa a tarefa de obter o conjunto de correspondências entre o par de ontologias passado como entrada. Para cada tarefa de *matching*, correspondências confirmadas e rejeitadas são geradas e armazenadas no banco. As mesmas são armazenadas nas tabelas *TblaliConfirmados* e *TblaliRejeitadas* respectivamente, que possuem um relacionamento (1..n) com *TblMatching*. Além da implementação da definição de correspondência apresentada neste trabalho, cada tupla de correspondência rejeitada e confirmada armazena o valor da chave estrangeira para a tarefa de *matching* cadastrada e o *matcher* que sugeriu a correspondência. A informação

do *matcher* de origem permite ao usuário verificar quais *matchers* mais adequados para determinada tarefa de *matching*.

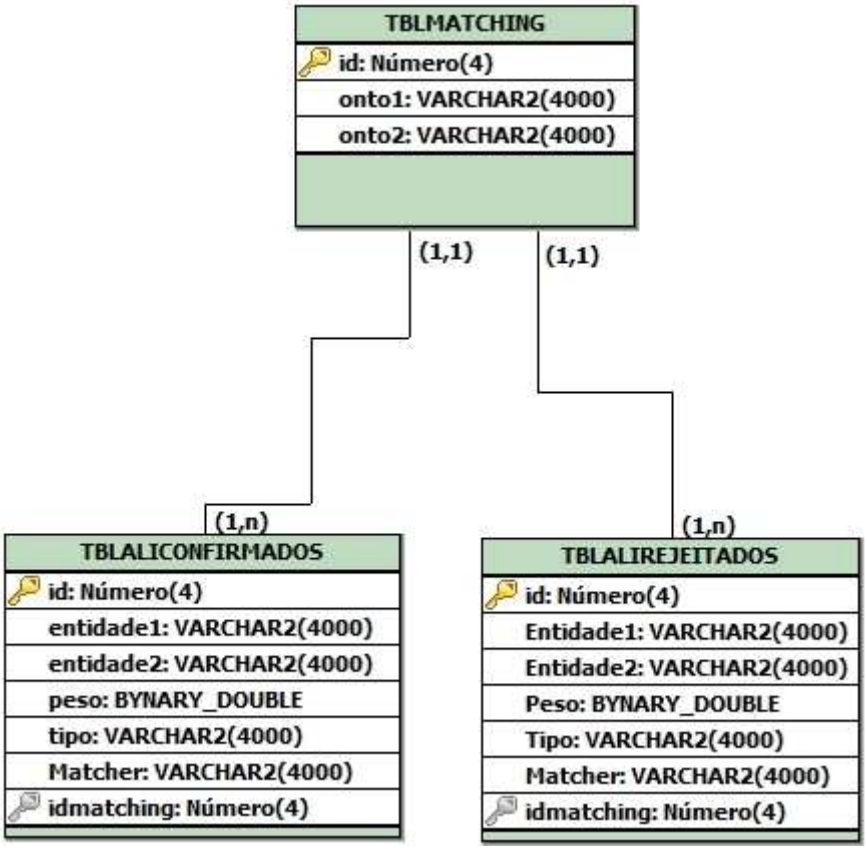

Figura 24. Modelo lógico para armazenas correspondências

O esquema lógico da Figura **24** permite uma fácil elaboração e rápida execução de consultas úteis ao protótipo. Sejam URI1 e URI2 os valores das URI´s de duas ontologias. Para capturar uma determinada tarefa de matching (par de ontologias previamente cadastradas):

```
IDMATCHING: NUMBER;
IDMATCHING = SELECT ID FROM TBLMATCHING
                    WHERE ONTO1 = @URI1 AND ONTO2= @URI2
```

em que ID identifica unicamente cada tarefa de *matching*. De posse dessa informação, pode-se facilmente obter as correspondências confirmadas e rejeitadas referentes àquela determinada tarefa:

```
SELECT * FROM TBLALICONFIRMADOS
WHERE IDMATCHING = @IDMATCHING
```

```
SELECT * FROM TBLALIREJEITADOS
WHERE IDMATCHING = @IDMATCHING
```

A Figura **25** ilustra o diagrama de caso de uso do protótipo, com as principais funcionalidades. As funcionalidades inerentes ao usuário são:

- **Fornecer entrada para geração das subontologias:** O usuário fornece o conjunto entrada para gerar as subontologias. Cada par de subontologias trabalhadas caracteriza uma iteração quantitativa.
- **Escolher e executar um *matcher*:** Para dar início ao processo de geração de correspondências candidatas, o usuário deve escolher e executar um *matcher*, que irá gerar correspondências entre as subontologias, baseado em suas características. As correspondências geradas pelo *matcher* são agregadas às correspondências relevantes recuperadas no BDR, gerando as correspondências candidatas de uma determinada iteração qualitativa.
- **Prover *feedback*:** Outra importante funcionalidade do sistema entra em ação, que é receber o *feedback* do usuário sobre o alinhamento candidato. Tal *feedback* irá moldar o alinhamento candidato, resultando em um alinhamento final. Espera-se que o usuário forneça o *feedback* utilizando conhecimentos acerca do domínio e habilidades cognitivas.
- **Imprimir alinhamento resultante:** Ao final das interações, o usuário pode imprimir o alinhamento resultante (ou final) gerado.

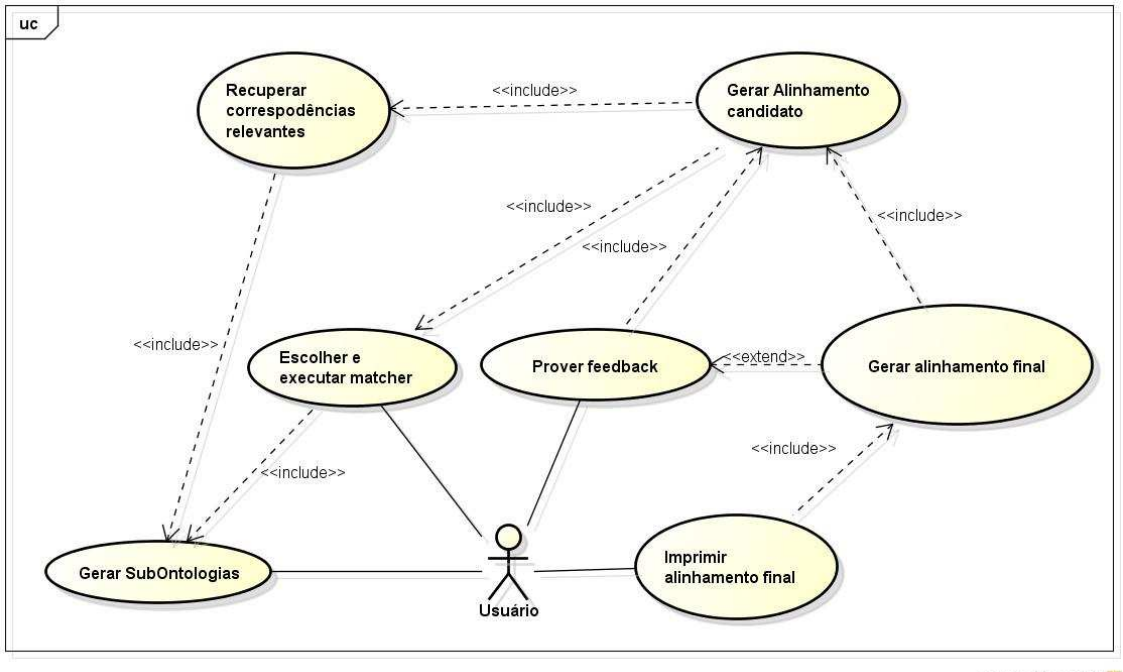

Figura 25. Diagrama de casos de uso do protótipo.

O diagrama de classe do protótipo I3M é ilustrado na Figura **26**. As classes *DBOracle*, *UtilAlignmentAPI* e *UtilOWLAPI* são responsáveis por implementar as operações envolvendo respectivamente banco de dados ORACLE (inserir, remover, atualizar tuplas), manipulação de alinhamentos (criar, deletar, alterar correspondências, métodos para agregação de correspondência) e manipulação das ontologias (criação das subontologias). A classe *MatrixSimilarity* implementa o conceito de matriz de similaridade apresentado na seção 2.3. A *BothDBAlignment* implementa as funcionalidades mistas (que envolvem bancos de dados e manipulação de alinhamento) e a *ClasseMediadora* instancia um objeto de cada classe. A *InterfacePrincipal* é a classe de fronteira do sistema, que implementa a interface de interação entre o usuário e o protótipo.

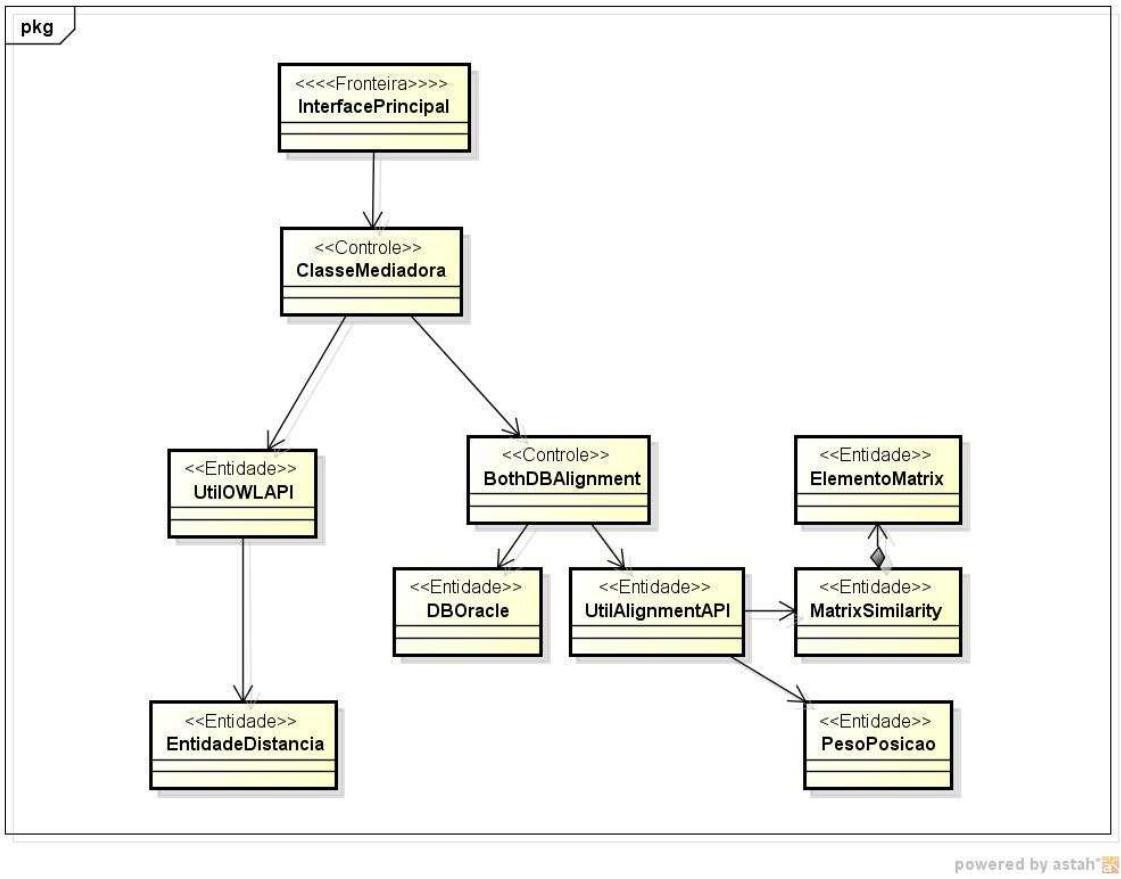

Figura 26. Diagrama de classes

3.5 Exemplo Ilustrativo

Para ilustrar o uso da abordagem e do protótipo I3M, considere o seguinte

cenário: um usuário deseja obter interoperabilidade com a *Web dos dados*, estabelecendo correspondências entre os esquemas de uma determinada ontologia proprietária e a ontologia do DBPedia. A ontologia proprietária é relativamente pequena, contendo um pequeno número de conceitos e refere-se ao domínio biológico, incluindo conceitos como animais, plantas e termos relacionados. A Figura **27** mostra a hierarquia da ontologia proprietária. Tal ontologia foi desenvolvida visando abranger parte do domínio escolhido, apenas para fins didáticos, com o objetivo de facilitar esta exemplificação. A ontologia do DBPedia é uma ontologia abrangendo vários domínios e contém centenas de conceitos.

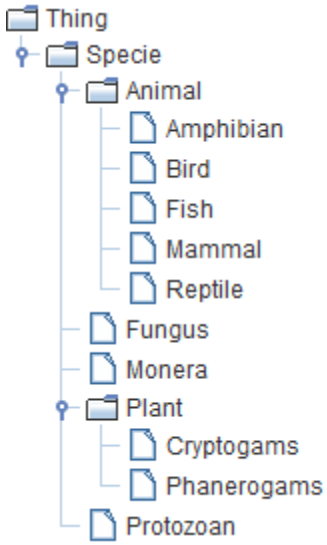

Figura 27. Hierarquia da ontologia proprietária

A Figura **28** mostra a interface inicial do protótipo I3M, desenvolvido para validar a abordagem proposta, em que as entradas devem ser especificadas pelo usuário. O usuário pode escolher entre filtrar as ontologias ou usá-las em sua totalidade.

Considerando o cenário proposto, será executada uma filtragem na ontologia ***DBPedia***, devido ao seu grande número de conceitos. Para isso, o usuário deve especificar um conceito e um número inteiro que servirá como base na delimitação do tamanho da partição. Para facilitar a escolha deste conceito, um matcher de baixa complexidade (i.e, *matcher* sintático) pode ser utilizado para sugerir correspondências iniciais entre as ontologias de entrada. Com base em tais correspondências, o usuário pode escolher o conceito mais adequado a ser utilizado para gerar a partição da ontologia ***DBPedia*** na qual se irá trabalhar nas próximas iterações. A escolha do número inteiro delimitador da partição também fica a critério do usuário. Em geral, é recomendado que, nesta escolha, o usuário leve em consideração o seu conhecimento

prévio acerca do domínio das ontologias, bem como suas habilidades cognitivas e os limites de visualização impostos pela ferramenta. Em grandes ontologias, uma escolha de um número muito grande pode comprometer a qualidade do alinhamento gerado por *matchers* bem como o provimento do *feedback* por parte do usuário.

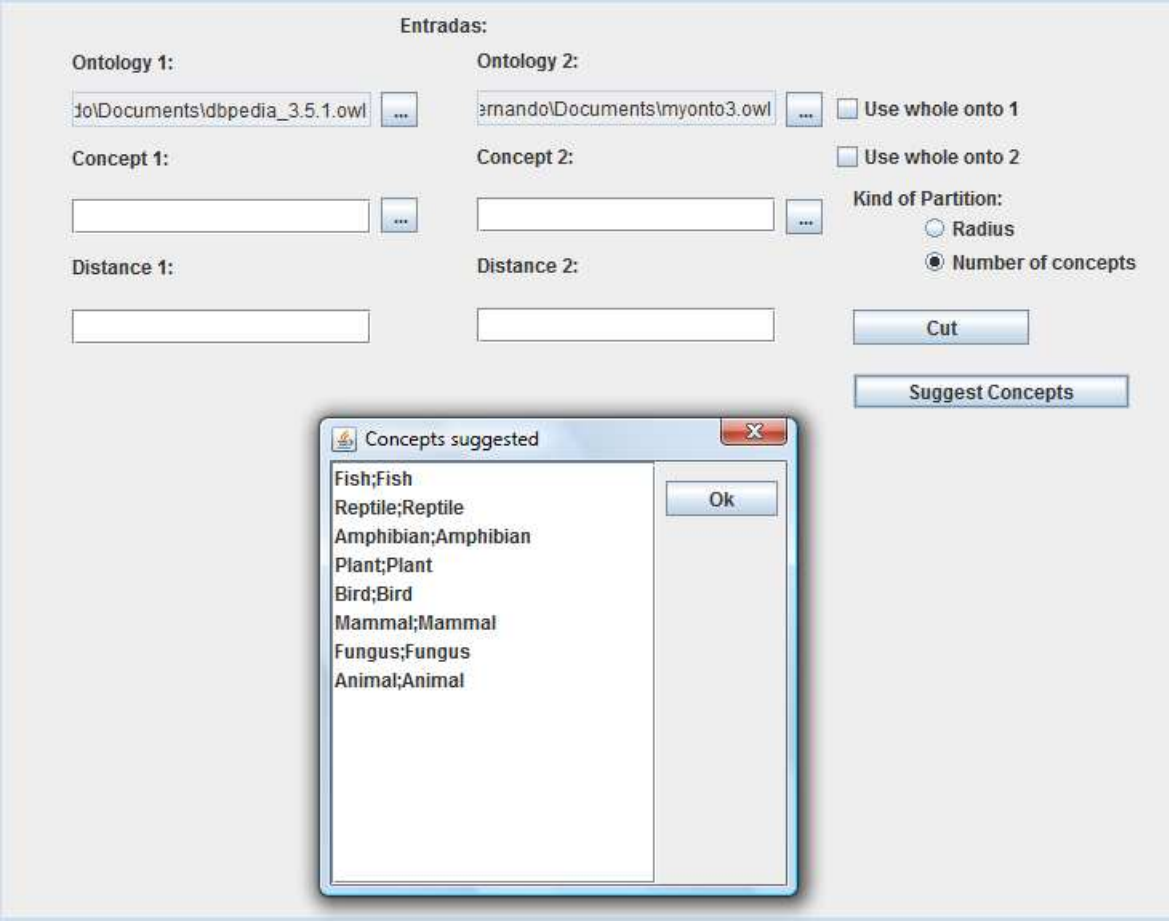

Figura 28. Definindo as partições

Suponha que o usuário escolha "Animal" (última sugestão de conceito na Figura **28**) e o número 15. Assim, a ontologia da ***DBPedia*** é filtrada, gerando uma partição que consiste no conceito de "Animal" com mais 14 conceitos relacionados, além de conter as propriedades que envolvem estes conceitos (Etapa 1). Entende-se como sendo propriedades envolvidas as:

- Propriedades de dados cujo domínio seja constituído de alguma classe selecionada.
- Propriedades de objeto cujo domínio e alvo possuam pelo menos uma classe selecionada cada um.

Como mencionado anteriormente, a ontologia proprietária é relativamente pequena, e não será filtrada. Após a filtragem da ontologia ***DBPedia***, correspondências

confirmadas e rejeitadas que envolvem conceitos das subontologias, definidas em interações anteriores, são retornadas. O usuário então pode selecionar e executar um *matcher* disponível no protótipo. O alinhamento gerado pelo *matcher* é filtrado (correspondências rejeitadas são eliminadas), agregado ao alinhamento inicial retornado pelo banco de dados (Etapa 2). Em seguida, o alinhamento resultante da agregação é exibido (Figura **29**). O usuário pode eliminar correspondências falsas e/ou adicionar novas correspondências que não foram identificadas anteriormente. As alterações serão gravadas no Banco de dados. O usuário pode então selecionar outro *matcher*, visando gerar um novo alinhamento a ser agregado com o alinhamento previamente gerado (Repetição das etapas 2 e 3, até que o usuário decida encerrar o processo).

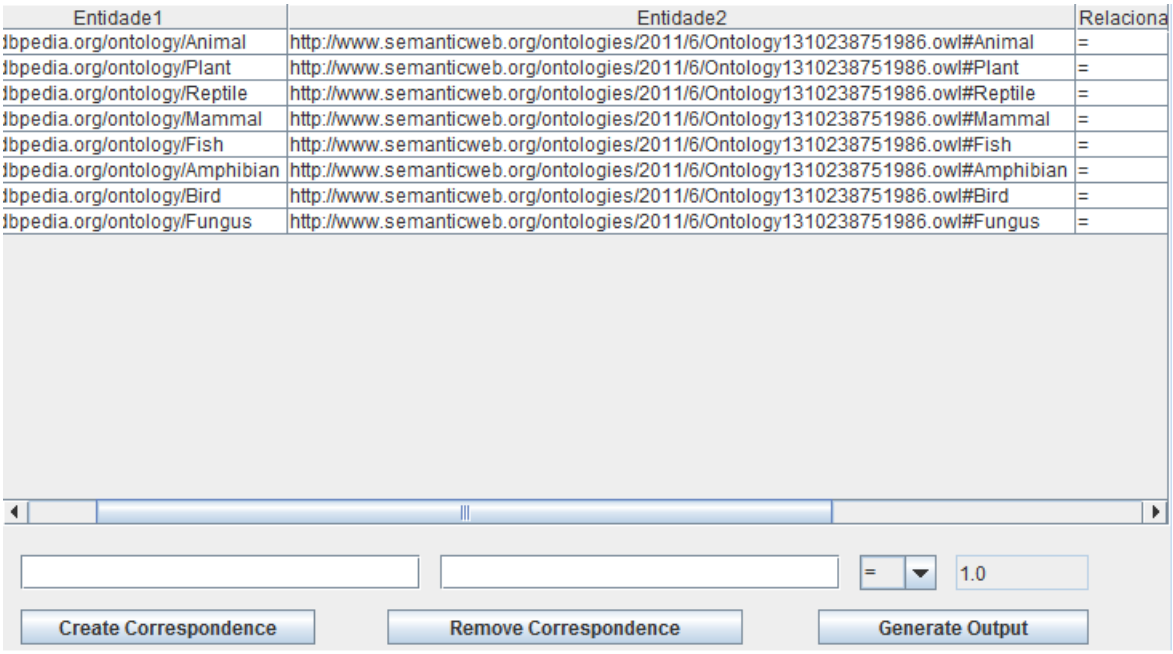

Figura 29. Exibindo os Alinhamentos

Após as interações necessárias para a definição de um alinhamento satisfatório entre as subontologias, o usuário pode alterar o tamanho delas, caso julgue necessário. Por exemplo, suponha que o usuário examine a estrutura da subontologia de tamanho 15 (Figura **30** (b)) e suspeite que existam mais conceitos relacionados ao domínio biológico que devem ser considerados no alinhamento final entre as ontologias. Então, neste caso, o usuário pode, por exemplo, alterar os conceitos ou os inteiros passados como parâmetro inicial na Etapa 1. Suponha que o usuário continue considerando a ontologia proprietária completa, mantenha o conceito inicial “Animal”, mas decida

aumentar o tamanho da partição oriunda da ontologia ***DBPedia***, alterando o inteiro passado de 15 para 25 (interação quantitativa). Como resultado, uma nova subontologia é criada (Figura **30** (a)) e o usuário pode interagir com ela, repetindo as etapas 2 e 3 várias vezes (constituindo iterações qualitativas) para gerar novas correspondências, e refinar as correspondências já existentes, até obter um alinhamento final.

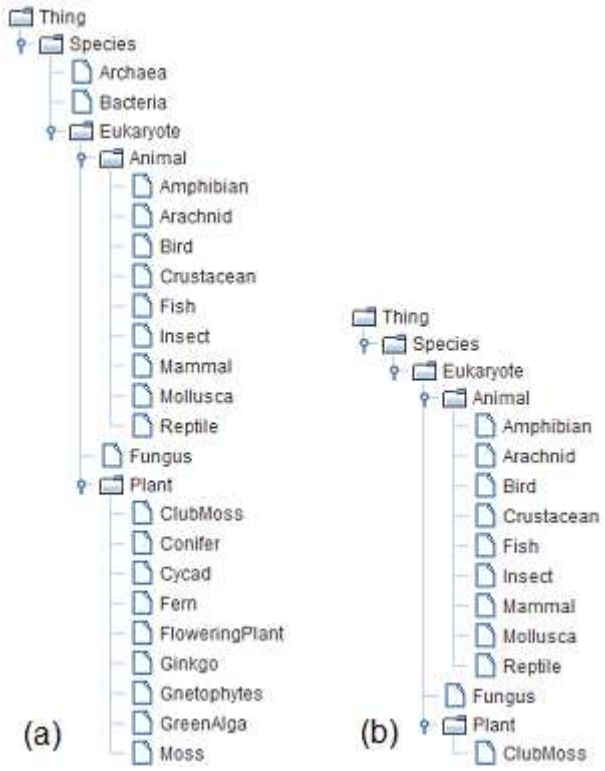

Figura 30. Subontologias oriundas da ontologia DBPedia (tamanhos 25 e 15 respectivamente)

Para mostrar o potencial do recurso de filtragem, presente na abordagem proposta, foi feita uma pequena experiência utilizando a ontologia DBpedia, a ontologia proprietária, descrita na **Figura 27**, e os *matchers* OLA [49] e AROMA [44]. A **Tabela 1** mostra os resultados do OLA e AROMA, recebendo como entrada a ontologia ***DBPedia*** e a ontologia particular, com e sem filtragem de 25 conceitos respectivamente.

Em cada célula da tabela, o número ao lado esquerdo da barra é referente ao resultado do *matching* utilizando filtragem, e ao lado direito, sem filtragem. Por exemplo, no esquema filtrado, o OLA encontrou 13 correspondências, enquanto sem filtragem, foram encontradas 11 correspondências. Das 13 correspondências encontradas no esquema filtrado, 3 são falsas, 10 são corretas e 0 ausentes. Das 11 correspondências encontradas para esquemas sem filtragem, 11 são falsas, 0 corretas e 0 ausentes. O mesmo raciocínio vale para o AROMA. Para determinar quais

correspondências são verdadeiras e falsas, um alinhamento de referência foi criado com a ajuda de um "*expert*" no domínio. Tal alinhamento é tido como ideal e qualquer correspondência não contida nele é considerada como falsa.

Tabela 1. Resultados com/sem filtragem

	OLA Matcher	AROMA Matcher	Referência
Correspondências encontradas	13/11	8/8	10
Correspondências falsas	3/11	0/0	0
Correspondências corretas	10/0	8/8	10
Correspondências ausentes	0/0	2/2	0

Conforme indicado na Tabela **1**, a filtragem pode aumentar significativamente a qualidade dos resultados em determinados *matchers*. De acordo com [8], OLA é um algoritmo fortemente baseada em técnicas estruturais. Isso explica a grande diferença de qualidade nos resultados com e sem filtragem, relativos à saída do *matcher* OLA.

Segundo [7], existe a possibilidade de o usuário cometer erros durante as interações com o sistema, como por exemplo, podar uma correspondência que não deveria ser podada ou confirmar uma correspondência que seja um falso positivo. Logo, o protótipo desenvolvido possui mecanismos de desfazer as ações do usuário. A Figura **31** exibe a tela relativa à tabela de correspondências rejeitadas do protótipo. O botão "Retornar Correspondência" é responsável por desfazer uma rejeição, removendo a correspondência da tabela de rejeitados e inserindo-a na tabela de correspondências confirmadas.

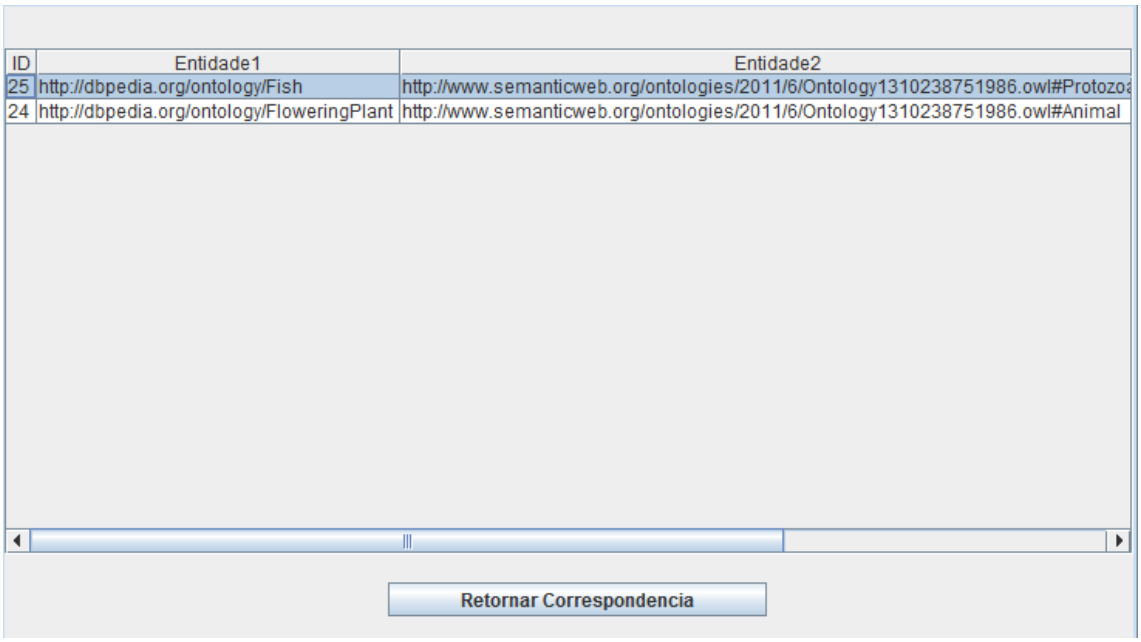

Figura 31. Opção de desfazer correspondências rejeitadas

Outra funcionalidade cognitiva presente no protótipo consiste em exibir ao usuário os conceitos das partições que ainda não possuem correspondência. O caráter incremental da abordagem exalta ainda mais a importância desta funcionalidade, que deve ser dinâmica, de acordo com as alterações feitas pelas interações com o usuário. A Figura 32 mostra a tela responsável por exibir as listas de conceitos sem correspondências.

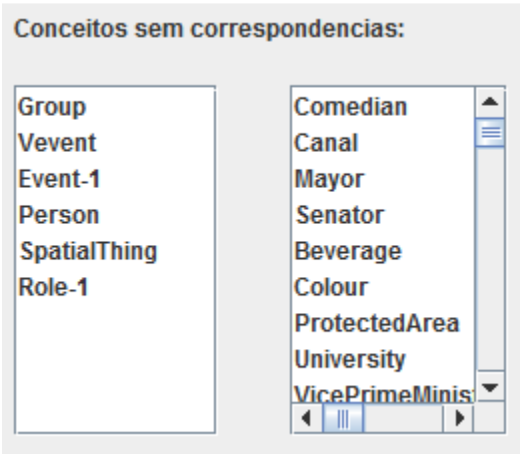

Figura 32. Conceitos sem correspondência

A ferramenta também possui uma opção de gerar uma saída consistindo de um documento contendo o conjunto de correspondências confirmadas expressas em RDF cujo formato é proposto em [21]. Tal formato é descrito como se segue:

Nível que define a expressividade dos *matchings*. Para este trabalho utilizaremos nível 0

Aridade permitida entre os conceitos. Para este trabalho utilizar-se-á a aridade *:* (muitos-para muitos). O trecho de RDF abaixo representa um exemplo de alinhamento no formato de saída proposto.

Referência às ontologias de entrada

Conjunto de Correspondências que expressam as relações entre os conceitos das duas ontologias de entrada

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<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#'
        xmlns:align='http://knowledgeweb.semanticweb.org/heterogeneity/alignment#'>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type>**</type>
  <onto1>
    <Ontology rdf:about="file:/C:/Users/Fernando/Documents/russia1.owl">
      <location>file:/C:/Users/Fernando/Documents/russia1.owl</location>
    </Ontology>
  </onto1>
  <onto2>
    <Ontology rdf:about="file:/C:/Users/Fernando/Documents/russia2.owl">
      <location>file:/C:/Users/Fernando/Documents/russia2.owl</location>
    </Ontology>
  </onto2>
  <map>
    <Cell>
      <entity1 rdf:resource='http://lonely.org/russia#music'/>
      <entity2 rdf:resource='http://travel.org/russia#music'/>
      <relation>=</relation>
      <measure
rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
    </Cell>
  </map>
</Alignment>
</rdf:RDF>
```

No trecho RDF acima, além dos *namespaces* e *entities* usados ao longo do documento, há a descrição das características do alinhamento como um todo, e logo após, uma correspondência entre os conceitos *http://lonely.org/russia#music* e *http://travel.org/russia#music* cujo tipo de relacionamento é equivalência, e o peso é 1.

Capítulo 4

Experimentos

O objetivo deste capítulo é avaliar a abordagem incremental e a ferramenta desenvolvidas no capítulo anterior. Para isso, foi feito um estudo acerca das principais métricas bem como algumas metodologias utilizadas na avaliação de ferramentas de *matching*. Além disso, um *setup* de experimentos foi construído, a partir de dados provenientes da Web e do auxílio de *experts* nos domínios utilizados. O capítulo está dividido da seguinte forma: a Seção 4.1 introduz uma visão geral acerca das métricas para avaliação de ferramentas de *matching*; a Seção 4.2 detalha a construção e especificação dos artefatos usados nos experimentos; a Seção 4.3 exibe a metodologia utilizada e os dados coletados dos experimentos.

4.1 Visão Geral

O método mais utilizado pela comunidade para avaliar a qualidade dos alinhamentos gerados por uma abordagem é baseado em [11], e utiliza métricas originais do campo da recuperação de informação (*information retrieval)*. Essas métricas são a precisão (*precision*), a revocação (*recall*) e a medida F (*F-Measure*).

A precisão mede o percentual de correspondências válidas dentre as contidas em um alinhamento, ou seja, tem como objetivo calcular o nível de corretude do alinhamento. Assim, quanto maior o número de correspondências falsas (ou falsos positivos) o alinhamento contiver, menor será seu índice de precisão.

A revocação mede o percentual de correspondências válidas contidas em um alinhamento, em comparação ao total que este deveria conter, ou seja, tem como objetivo calcular o nível de completude do alinhamento. Assim, quanto maior o número de correspondências ausentes (ou falsos negativos) em um alinhamento, menor será seu índice de revocação.

As duas medidas citadas por si só não são suficientes para avaliar a qualidade de um alinhamento. O uso isolado das duas medidas pode dar uma falsa sensação de boa qualidade a um determinado alinhamento. Por exemplo, para atingir um alto nível de precisão em detrimento da revocação, basta retornar um pequeno conjunto de correspondências todas corretas. O oposto é facilmente alcançado bastando retornar todas as combinações possíveis de correspondência. Neste contexto, a medida F foi

criada como sendo a média harmônica entre as medidas de precisão e revocação. A Figura **33** mostra as fórmulas utilizadas para cálculo das medidas de precisão, revocação e medida F respectivamente, onde:

|A| = Número de correspondências corretas que não foram encontradas pela abordagem.
|B| = Número de correspondências corretas que foram encontradas pela abordagem.
|C| = Número de correspondências falsas que foram encontradas pela abordagem.

$$\text{Precisão} = \frac{|B|}{|B| + |C|}$$

$$\text{Revocação} = \frac{|B|}{|A| + |B|}$$

$$\text{F-Measure} = \frac{\text{Precisão x Revocação}}{(1 - \alpha) \text{ x Precisão} + \alpha \text{ x Revocação}}$$

Figura 33. Precisão, Revocação e F-Measure

A medida α contida na definição da fórmula do *F-Measure* é um valor arbitrário entre [0,1] que mede a importância (ou peso) de cada medida dentro da avaliação final. Um $\alpha = 0.5$ representa igualdade de peso entre precisão e revocação. Já para um $\alpha > 0.5$, a precisão teria maior influência no resultado final, e um $\alpha < 0.5$ daria maior peso para a medida revocação.

Para possibilitar o cálculo das medidas citadas, é definido um conjunto completo de correspondências considerado como sendo o alinhamento de saída ideal. Tal conjunto recebe o nome de Alinhamento de referência ou "*Gold Standard*" (GS). Após a obtenção de um alinhamento resultante (gerado por alguma abordagem), calcula-se a precisão e a revocação dos mapeamentos, tomando como base o GS seguido da média harmônica (*F-Measure)* entre as medidas.

As medidas da Figura **33** são amplamente usadas não somente para avaliar a qualidade dos resultados das abordagens automáticas de *matchings*, mas também para avaliação de trabalhos na área de *information retrieval* [11]. Porém, H. Do et al [24] considera que tais medidas por si só não são suficientes para validar abordagens de *matching* semi-automáticas, isto é, que envolvem o usuário. Isto se deve à existência de conceitos subjetivos envolvidos no provimento dos *feedbacks*, que devem ser levadas em consideração, e que as métricas existentes não são capazes de mensurar. Como

exemplo de tais conceitos, cita-se:

- Habilidades cognitivas do usuário
- Conhecimento de *background* do usuário sobre os domínios abordados
- Usabilidade das abordagens
- Esforço do usuário (nível de envolvimento)

Dessa forma, a avaliação de abordagens semi-automáticas é considerada uma problemática difícil e que atualmente ainda não possui uma solução concreta. Isto porque ainda não existem métricas adequadas capazes de capturar e quantificar as questões relativas ao usuário [48]. Ainda assim, existem alguns poucos trabalhos na literatura que efetuam testes de ferramentas semi-automáticas com perfis de usuários, descrevendo e avaliando as saídas de cada um. Tais trabalhos são citados em [48]. Neste contexto, experimentos empíricos foram feitos no intuito de verificar e investigar a viabilidade prática da abordagem.

4.2 Setup dos Experimentos

Para a validação da abordagem proposta utilizou-se o protótipo implementado e descrito na seção 3.4. Com o auxílio de um *expert*, foi criada uma ontologia proprietária alvo no domínio biológico, mais especificamente acerca da classificação dos seres vivos. Além da ajuda de um especialista no domínio, a ontologia também foi desenvolvida baseada no conhecimento provido pelo "*Tree Of Life Web Project*"[4] que consiste em um site que armazena informações acerca de diversos seres vivos descobertos por biólogos, incluindo as diversas classificações existentes.

O cenário proposto é de um usuário que possui interesse em inserir essa ontologia proprietária na rede, interligando-a na Web dos Dados. Para isso, a ideia é gerar um alinhamento entre esta ontologia e uma já existente. A ontologia ***DBPedia*** foi escolhida, por ser uma ontologia multi-domínio e bastante popular na Web dos dados. Além disso, também foi criado um alinhamento de referência entre a ontologia proprietária e a da ***DBPedia***. Tal alinhamento foi desenvolvido com o auxílio de um *expert* no domínio e exigiu bastante tempo e atenção, uma vez que foram estabelecidas

[4] http://www.tolweb.org/

manualmente as correspondências entre as ontologias utilizadas nos experimentos. A Tabela **2** exibe informações acerca de tais ontologias:

Ontologia	Classes	Prop. Objeto	Prop. Dados	Instâncias	Total de Axiomas
DBPedia	259	602	674	0	2458
Ontologia Proprietária	50	0	0	0	32

Tabela 2. Informações sobre as ontologias utilizadas

Com relação aos usuários, foram escolhidos dois alunos do curso de graduação em Ciências Biológicas. O primeiro aluno cursa o quarto semestre (metade do curso), enquanto que o segundo aluno está no último semestre. Ambos os estudantes desconhecem conceitos relacionados a ontologias e a tarefa de *matching* entre esquemas. A ideia é testar o envolvimento destes usuários com o protótipo, além de verificar uma possível relação entre a qualidade dos resultados obtidos com o nível de experiência do usuário.

4.3 Avaliação da precisão e revocação dos resultados obtidos

Nesse trabalho, foi executado um procedimento similar ao adotado em [25], o qual estende o processo tradicional de cálculo da qualidade dos *matchings* gerados, com o objetivo de capturar a subjetividade do usuário dentro do processo. Tal extensão consiste dos seguintes passos:

1. Fase de escolha e preparação dos usuários para as tarefas de validação: esta fase consistirá de um minicurso com duração de 10 à 20 minutos sobre os princípios básicos de ontologia, alinhamento, além de como usar corretamente o protótipo.
2. Designação de tarefas de *matching* ao usuário e imposição de um tempo para solucioná-las.
3. Capturar os resultados fornecidos pelos usuários e calcular os níveis de precisão, revocação e a medida F, baseados no alinhamento de referência previamente gerado.

Como diferencial, utilizou-se o tempo de uma hora e 30 minutos, extraindo-se resultados parciais a cada 30 minutos. Dessa forma, a cada 30 minutos, os alinhamentos

confirmados e rejeitados foram extraídos. A ideia é verificar e fazer uma análise (mesmo que empírica) da evolução dos resultados com o passar do tempo.

4.3.1 Experimentos e Avaliação dos Resultados

Alguns fatos interessantes foram constatados durante os experimentos. Ao longo das interações ambos os usuários foram identificando dificuldades e propondo sugestões e melhorias para a ferramenta. Isso mostra a importância de prover mecanismos cognitivos e que facilitem a usabilidade do protótipo.

A Tabela **3** mostra as entradas usadas pelos usuários na geração das partições trabalhadas ao longo das interações.

Usuário 1	Usuário 2
Molluscas 5 Invertebrate 10 M: Edit Distance	Grape 15 Conifers 20
Plant 7 Plant 5 (S) M: OLA	Muscle 30 Animal 30 M: AROMA
Animal 8 Animal 8 (S) M: Edit Distance	Disease 30 Fungus 30 M: AROMA
Fungus 10 Basidiomicets 10	Protein 30 Mollusks 10 M: OLA
Reptile 10 Reptile 10 (S) M: Edit Distance	Reptil 8 Vertebrate 5 M: JWNL
Molluscas 10 Invertebrate 10	Fungus 20 Ascomicets 15 M: Name_And_Property
Animal 20 Animal 35 (S) M: AROMA	Eukaryote 3 Specie 6
	Eukaryote 10 Specie 6

Tabela 3. Entradas utilizadas para geração das partições. Cada entrada é constituída de dois conceitos e dois números inteiros que definem as partições. O símbolo (S) significa que a escolha dos conceitos foi feita utilizando o mecanismo auxiliar cognitivo da Figura 28 e M lista os *matchers* utilizados para aquelas partições.

O usuário 1 sugeriu apenas modificações no layout da tabela de exibição das correspondências (Figura **29**), como troca de posições das colunas e supressão das URI´s dos conceitos de cada correspondência. O usuário 2 foi mais criterioso, efetuando

sugestões como (i) o armazenamento das ações feitas e a criação de uma tela de log (onde fossem exibidas as últimas ações efetuadas); (ii) Permissão da edição manual de correspondências diretamente na tabela.

Além disso, algumas iniciativas cognitivas presentes no protótipo foram elogiadas, como a geração prévia de sugestões para a criação das partições (Figura **28**), a exibição dos conceitos sem correspondências (Figura **32**) e a possibilidade de filtrar a ontologia e trabalhar nas partições (caráter incremental). Ambos os usuários concordaram que o fato de se trabalhar com partições encoraja a geração manual de correspondências. A Tabela **4** exibe os resultados de precisão, revocação e medida F obtida para cada usuário.

	Precisão	Revocação	Medida F
Usuário 1 (30 min)	0,76	0,11	0.20
Usuário 1 (1 hora)	0.73	0.15	0.24
Usuário 1 (1h 30min)	0.79	0.31	0.45
Usuário 2 (30 min)	0,94	0,17	0.28
Usuário 2 (1 hora)	0.98	0.34	0.51
Usuário 2 (1h 30min)	0.91	0.6	0.72

Tabela 4. Resultados das interações.

Analisando a Tabela **4**, observa-se que os níveis de precisão dos resultados são elevados e os de revocação são relativamente baixos, o que leva a acreditar que, a curto prazo, a solução incremental é adequada para casos em que se precisa de um alinhamento com uma alta corretude, abrindo mão do baixo nível de completude. A necessidade desse tipo de solução basicamente dependerá dos requisitos do usuário.

Se o mesmo necessita, por exemplo, tomar uma decisão e exige o retorno de uma amostra do número total de ocorrência, mas que prevaleça um alto nível de certeza/corretude, então a solução incremental será interessante. Porém, se o mesmo necessita de uma amostra contendo muitas ocorrências, sem exigir um rigor de certeza, então a solução incremental já não seria tão eficiente.

Como exemplo mais geral, cita-se o caso de consultas a bases de dados integradas na Web. Em boa parte das consultas, um grande número de resultados é retornado (isto acontece devido à grande massa de dados atualmente existentes, boa parte proveniente dos recursos oriundos da Web 2.0), o que torna inviável uma verificação manual por parte do usuário. Logo, é desejável que o resultado gerado apresente um alto índice de documentos relevantes (alta precisão) nas primeiras ocorrências do que mostre milhares de documentos pouco relevantes [51, 52].

Para o caso da Web dos dados, o mesmo raciocínio pode ser usado. Dada uma consulta feita à Web dos dados, um *Web Crawler* irá percorrer as bases de dados, capturando informações, de acordo com as correspondências existentes entre as mesmas. Utilizar alinhamentos (entre as bases) que possuam uma maior revocação e menor precisão acarreta um maior número de ocorrências com menor índice de relevância e vice-versa.

Com o passar do tempo, o nível de completude vai crescendo e conseqüentemente, a quantidade de informação com a qual o usuário tem de manipular. Dessa forma, um desafio para os usuários constitui em manter o alto nível de precisão, à medida que a revocação cresce.

	Nº de Correspondências Rejeitadas	Nº de Correspondências. Confirmados
Usuário 1 (30 min)	23	30
Usuário 1 (1 hora)	79	41
Usuário 1 (1h 30min)	300	79
Usuário 2 (30 min)	56	36
Usuário 2 (1 hora)	553	70
Usuário 2 (1h 30min)	545	131

Tabela 5. Número de correspondências rejeitadas e confirmadas a cada 30 minutos.

Outro fato interessante foi o grande número de correspondências rejeitadas (Tabela **5**). Ao final dos experimentos, o usuário 1 rejeitou 545 correspondências e o usuário 2, 300 correspondências. Isso reforça a necessidade de melhoria em *matchers*, reforçando a importância de iniciativas como a OAEI, além da intervenção humana na tarefa de *matching*. O OAEI é uma iniciativa criada em 2004 com o objetivo de avaliar soluções e estratégias de *matching*, comparar suas performances, destacar suas fraquezas e virtudes, melhorar as técnicas de avaliação existentes bem como a comunicação entre os desenvolvedores das soluções e, acima de tudo, contribuir para o desenvolvimento da área de *ontology matching*.

Ao final dos experimentos, ambos os usuários concordaram que se houvesse mais tempo de interação com a ferramenta, mais correspondências poderiam ser geradas e refinadas. Este segundo ponto (limite de tempo) pode justificar o baixo nível de revocação obtido nos resultados.

Capítulo 5

Conclusão e Trabalhos Futuros

Neste capítulo são feitas as conclusões finais (Seção 5.1), bem como são discutidas as vantagens e limitações da abordagem apresentada (Seção 5.2), e alguns dos possíveis trabalhos futuros (Seção 5.3).

5.1 Conclusão

Neste trabalho foi feito um estudo acerca do estado-da-arte das soluções em *ontology matching*, bem como os cenários onde o uso de tais soluções se faz necessária. Além das técnicas básicas, técnicas auxiliares como a filtragem de esquemas e o *feedback* do usuário foram visitadas com o objetivo de desenvolver uma estratégia semi-automática de *matching* mais adequada para cenários em que grandes ontologias estão envolvidas.

Após a elaboração da solução proposta, uma ferramenta foi desenvolvida utilizando tecnologias largamente usadas em implementações de ferramentas de *matching*. Também foi feito um estudo acerca dos métodos e métricas existentes na literatura para avaliação da qualidade do resultado das soluções de *matching*. Além disso, experimentos com usuários e especialistas de domínio foram feitos utilizando a ferramenta implementada. Dessa forma, como contribuições deste trabalho, citam-se:

- Estudo acerca do estado-da-arte das soluções de *matching* entre ontologias além de métodos e métricas para avaliação destas soluções;
- Uso do conceito de filtragem no contexto do problema de *matching*;
- Criação de uma abordagem semi-automática incremental para geração de correspondências entre ontologias;
- Implementação da ferramenta baseada na abordagem proposta.

5.2 Vantagens e Limitações

Algumas vantagens e limitações da abordagem proposta são discutidas nesta seção. Em primeiro lugar, o recurso de filtragem traz benefícios, tanto para o usuário

quanto para o algoritmo automático que se está utilizando. Alguns *matchers* geram alinhamentos em menos tempo e fornecendo menos erros utilizando ontologias filtradas, em vez de usar toda a ontologia como entrada, implicando uma drástica diminuição no número de falsos positivos, pois muitas possibilidades de correspondências incorretas são eliminadas.

Isso também significa que o usuário tem menos dificuldade em corrigir os alinhamentos gerados, pois a filtragem é citada em [6] como sendo um forte auxílio cognitivo, que facilita o fornecimento de *feedback* do usuário, uma vez que o mesmo vai lidar com uma quantidade menor de conceitos. Assim, mesmo que um *matcher* automático forneça resultados semelhantes entre ontologias filtradas e não filtradas, as chances de erro por parte do usuário são reduzidas. Outra vantagem é em caso de evolução da ontologia. Uma vez que o usuário sabe que parte da ontologia foi alterada, é possível considerar apenas a partição que foi alterada, em vez de trabalhar com toda a ontologia.

Para obter bons resultados, é desejável que o usuário tenha um bom conhecimento sobre os domínios envolvidos. Além disso, o processo poderá exigir do usuário uma quantidade de tempo considerável. Assim, como limitações, destacam-se que este processo pode ser demorado e exigir do usuário um certo conhecimento acerca do domínio.

5.3 Trabalhos futuros

O envolvimento do usuário vem ganhando cada vez mais importância no processo de *matching*. Essa importância fica clara com o surgimento das atuais abordagens semi-automáticas para a definição de correspondências entre duas ontologias. Belhajjame et al. [40] define genericamente o que é um *feedback* do usuário, além de propor um modelo geral para o mesmo baseado no seu uso entre vários trabalhos de diversas áreas. Além disso, o mesmo trabalho adverte que um determinado *feedback* pode estar inconsistente (em contradição com outro *feedback*) e/ou inválido e mostra regras e casos em que isso acontece. Um futuro aperfeiçoamento será levar as práticas de validade de *feedback* do usuário citadas por [40] em consideração, adotando o que se chama de gerência de *feedback*.

Na abordagem proposta por este trabalho, a escolha do *matcher* a ser executado

em cada interação fica a cargo do usuário. Alguns trabalhos propõem uma automatização na escolha do *matcher* a ser executado. Essa automatização pode ser incorporada a este trabalho e pode ser baseada na qualidade das correspondências geradas que deve ser medida a partir do número de correspondências rejeitadas pelo usuário (*feedback*).

Alguns trabalhos de *matching* atuais que envolvem o usuário dão uma significativa importância ao "*front-end*" de suas aplicações e defendem a construção de uma interface amigável [25, 35, 39, 41]. Assim, existe um conjunto de técnicas que auxiliam no suporte cognitivo do usuário. Visto que esta abordagem exige informações de entrada e certo conhecimento do usuário, é importante que a interface do protótipo proposto torne-se mais elaborada, satisfazendo maiores exigências e facilitando, por exemplo, a navegação entre os esquemas para o usuário.

O fato de se trabalhar com ontologias pode gerar algumas vantagens. Em [16] é proposta uma abordagem que, dado um conjunto de correspondências iniciais, utiliza um mecanismo de inferência para auxiliar o processo de descoberta de novas correspondências. Com o objetivo de tirar vantagem do mecanismo de inferência, um possível trabalho futuro consiste em criar um arquivo e armazenar as correspondências que não foram rejeitadas sob a forma de axiomas na linguagem OWL. Desse modo, um *reasoner* poderá ter acesso a esse arquivo e assim inferir novos axiomas. Tais axiomas serão na verdade novas correspondências descobertas através da inferência lógica. A Figura **34** ilustra um exemplo de inferência sobre correspondências.

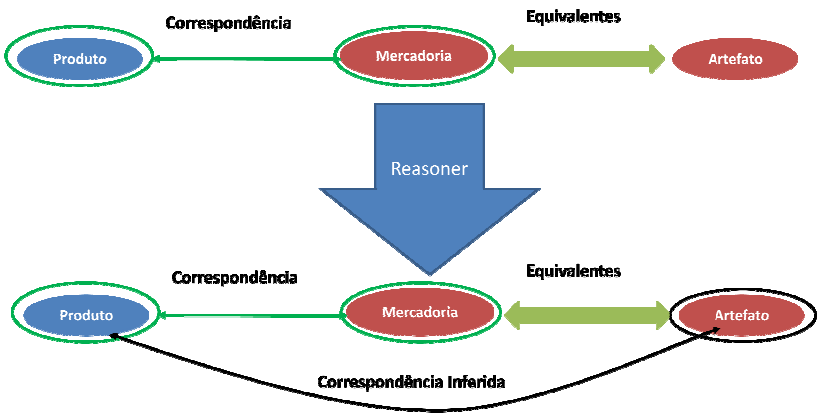

Figura 34. Exemplo de correspondência inferida.

Suponha duas ontologias que em uma delas exista uma classe chamada **Produto**, e na outra exista duas classes chamadas **Mercadoria** e **Artefato**. Suponha ainda que

artefato e **mercadoria** sejam classes equivalentes e que há uma correspondência que liga **produto** à **mercadoria**. Logo, pode-se fazer uso de um *reasoner* para inferir que existe uma correspondência entre **produto** e **artefato**. O mesmo raciocínio pode ser usado para relação de subsunção. Em OWL, os axiomas resultantes da Figura **34** são descritos abaixo:

```
<owl:Class rdf:about="&onto2;#Mercadoria">
        <owl:equivalentClass rdf:resource="&onto2;#Artefato"/>
</owl:Class>

<owl:Class rdf:about="&onto1;#Produto">
        <owl:equivalentClass rdf:resource="&onto2;#Mercadoria"/>
</owl:Class>

<owl:Class rdf:about="&onto1;#Produto">
        <owl:equivalentClass rdf:resource="&onto2;#Artefato"/>
</owl:Class>
```

O trecho destacado em vermelho simboliza o axioma inferido pelo *reasoner*. Os axiomas inferidos são convertidos em quíntuplas (com índice de confiança igual a 1), e em formato RDF a fim de que seja exibido no arquivo de saída. Vale salientar que o processo exemplificado pela Figura **34** é válido para correspondências que envolvam relações de equivalências e de subsunções.

A escolha da linguagem OWL vem do fato de ser a linguagem recomendada pela W3C para expressar correspondências entre termos de ontologias. Ademais, isso facilita a integração de informações, visto que ontologias de entrada e alinhamentos estão expressos em uma mesma linguagem.

Referências

[1] K. Belhajjame, N. W. Paton, S. M. Embury, A. A. A.Fernandes, and C. Hedeler. *Feedback*-based annotation, selection and refinement of schema mappings for dataspaces. *In 13th International Conference on Extending Database Technology*. Lausanne, Switzerland, pp 573–584, 2010.

[2] M. Franklin, A. Y. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Records*: 27–33, 2005.

[3] Rahm E., Bernstein P. A. A survey of approaches to automatic schema *matching*. In *Very Large Databases Journal*: 334-350, 2001.

[4] Ontology alignment evaluation initiative. In *http://oaei.ontologymatching.org/*.

[5] Lutz, M. *Ontology-based Discovery and Composition of Geographic Information Services*. Phd Thesis, Institut für Geoinformatik, Germany, 2006.

[6] Isabel F. Cruz and Huiyong Xiao. The Role of Ontologies in Data Integration. *Journal of Engineering Intelligent Systems*, 13(4): 245–252, 2005.

[7] F. Shi, J. Li, and J. Tang. Actively learning ontology matching via user interaction. In *Proceedings of the 8th International Conference of Semantic Web (ISWC'2009)*, Chantilly, USA, pp. 585-600, 2009.

[8] Calvanese, D., De Giacomo, G., Lenzerini, M., Lembo, D., Poggi, A., Rosati, R.: MASTRO-I: Efficient Integration of Relational Data through DL Ontologies. *In Proceedings of the Description Logic Workshop (DL'07)*, Brixen, Italy pp. 227-234, 2007.

[9] R. Fagin, L. M. Haas, M. Hern´andez, R. J. Miller, L. Popa, and Y. Velegrakis, Clio: Schema Mapping Creation and Data Exchange, In *Conceptual Modeling: Foundations and Applications, Essays in Honor of John Mylopoulos*. Springer, vol. 5600, pp. 198–236, 2009.

[10] Bernstein, P. A., Melnik, S., Churchill, J. E. Incremental schema *matching*, *In Proceedings of the international conference on very large data bases*, Seoul, Korea,pp. 1167-1170, 2006.

[11] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[12] G. A. Miller, R. Beckwith, C. D. Fellbaum, D. Gross, K. Miller. WordNet: An online lexical database. *International Journal of Lexicography*. 3(4), pp. 235–312, 1990.

[13] Chen, D., Lastusky, J., Starz, J., Hookway, S. A User Guided Iterative Alignment Approach for Ontology Mapping. In *Proceedings of the International Conference on Semantic Web and Web Services*. Las Vegas, USA, pp. 51-56, 2008.

[14] Madhavan, J., Bernstein, P.A., Rahm, E. Generic schema matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, Rome, Italy, pp. 49–58, 2001.

[15] Cao, Z., Li, K., Liu, Y. Putting Feedback into Incremental Schema Matching, *World Congress on Software Engineering*, Xiamen, China, pp.332-336, 2009

[16] Jain, P., Hitzler, P., Sheth, A. P., Verma, K., Yeh, P. Z.. Ontology Alignment for Linked Open Data. In *Proceedings of the 9th International Conference of Semantic Web*, Shanghai, China, LNCS 6496, pp. 401-416, 2010.

[17] Bizer, C., Heath, T., Berners-Lee, T. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems, Special Issue on Linked Data*, 5 (3):1-22, 2009.

[18] Heath, T., Motta, E. Revyu: Linking reviews and ratings into the Web of Data. *Journal of Web Semantics*, 6(4):266-273, 2008.

[19] Becker, C., Bizer, C. DBpedia Mobile - A Location-Aware Semantic Web Client. *Semantic Web Challenge at International Semantic Web Conference*, Karlsruhe, Germany, 2008.

[20] Melnik, S., Garcia-Molina, H., Rahm, E. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. *International Conference on Data Engineering*, pp. 117-124. San Jose, EUA: IEEE Computer Society, 2002.

[21] Euzenat, J., Shvaiko, P. *Ontology Matching*. Springer. 2007.

[22] Duan, S., Fokoue, A., Srinivas, K. One Size Does Not Fit All: Customizing Ontology Alignment Using User Feedback. In *Proceedings of 9th International Semantic Web Conference* (1). Shanghai, China, pp 177-192, 2010.

[23] Euzenat, J., Alignment API. In http://alignapi.gforge.inria.fr/, 2011.

[24] Do, H., Melnik, S., Rahm, E. Comparison of schema matching evaluations. In *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems*, Erfurt, Germany, pp. 221-237, 2003.

[25] Burgess, C. *Harnessing User Input to Improve Linked Data Fidelity* – Master Thesis, University of Dublin, Ireland, 2010.

[26] Jarrar, M., Dikaiakos, M. A query formulation language for the data web. *IEEE Transactions on Knowledge and Data Engineering*. IEEE Computer Society, 2010.

[27] Zhong, Q., Li, H., Li, J., Guotong, X., Tang, J., Zhou, L., and Pan, Y. A Gauss Function Based Approach for Unbalanced Ontology Matching. *SIGMOD'09*, Providence, Rhode Island, USA, 2009.

[28] Peukert, E., Massmann, S., König, K. Comparing Similarity Combination Methods for Schema Matching. In *Proceedings of the 40th Annual Conference of the German Computer Society*, Jahrestagung, Germany, 2010.

[29] Mao, M., Peng, Y., Spring, M. A Harmony based Adaptive Ontology Mapping Approach. In *Proceedings of the 2008 International Conference on Semantic Web and Web Services*, Las Vegas, USA, pp. 336-342, 2008.

[30] Ehrig, M., Staab, S. QOM - Quick Ontology Mapping. In *Proceedings of the 3rd International Semantic Web Conference*, Hiroshima, Japan, pp. 683-697, 2004.

[31] Ji, Q., Haase, P., Qi, G. Combination of Similarity Measures in Ontology Matching by OWA Operator, In *Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in KnowledgeBase Systems*, Málaga, Spain, 2008.

[32] Seligman, L., Mork, P., Halevy, A., Smith, K., Carey, M. J., Chen, K., Wolf, C., Madhavan, J., Kannan, A., and Burdick, D. OpenII: An Open Source Information Integration Toolkit. Integration *The Vlsi Journal*, pp. 1057-1059, 2010.

[33] Algergawy, A. *Management of XML Data by Means of Schema Matching*. Master Thesis, Otto-von-Guericke-Universität, Magdeburg, Germany, 2010.

[34] Villegas, A., Olivé, A.: On computing the importance of entity types in large conceptual schemas. In *Heuser, C.A, Pernul, G. (Eds.) ER 2009 Workshops*. LNCS, vol. 5833, pp. 22-32. Springer, Heidelberg, Germany, 2009.

[35] Hu, W., Qu, Y., Cheng, G. Matching large ontologies: A divide-and-conquer-approach. In *Proceedings of Data Knowledge Engineering Journal*. 67(1):140-160, 2008.

[36] Do, H.H., Rahm, E. Matching large schemas: Approaches and evaluation. *Journal of information System*. 32(6): 857-885, 2007.

[37] Rahm, E., Do, H. H., Massmann, S. Matching large XML schemas. In *ACM SIGMOD Record*: 33(4), 26-31, 2004.

[38] Villegas, A., Olivé, A. A Method for Filtering Large Conceptual Schemas. In *Proceedings of the 29th International Conference on Conceptual Modeling*, Vancouver, Canada, pp. 247-260, 2010.

[39] Rahm, E. Towards large-scale schema and ontology matching. In Bellahsene Z, Bonifati A, Rahm E (eds): *Schema Matching and Mapping, Data-Centric Systems and Applications Series*, Springer, Germany, pp. 1–25. 2010.

[40] Belhajjame, K., Paton, N. W., Fernandes, A. A. A., Hedeler, C., Embury, S. M. User Feedback as a First Class Citizen in Information Integration Systems. In *proceedings of the 5th Biennial Conference on Innovative Data Systems Research*, Asilomar, USA, pp. 175-183, 2011.

[41] Granitzer, M., Sabol, V., Onn, K. W., Lukose D., Tochtermann K. Ontology Alignment – A Survey with Focus on Visually Supported Semi-Automatic Techniques. *Journal of the Future Internet*, 2: 238-258, 2010.

[42] Madhavan, J., Bernstein, P., Rahm, E. Generic schema matching with Cupid. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, Roma, Italy, pp. 49–58, 2001.

[43] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S. DBpedia - A crystallization point for the Web of Data. *Journal of Web Semantics* 7(3):154–165, 2009.

[44] David, J., Guillet, F., Briand, H. Matching directories and OWL ontologies with AROMA. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, New York, USA, pp. 830–831, 2006.

[45] Tummarello, G., Oren, E., Delbru, R. Sindice.com: Weaving the open linked data. In *Proceedings of the 6th International and 2nd Asian Semantic Web Conference*, Busan, Korea, pp. 547-560, 2007.

[46] Noy, N. F., Musen, M. A. The PROMPT suite: Interactive tools for ontology merging and mapping. In the *International Journal of Human-Computer Studies*, 59(6), pp. 983-1024, 2003.

[47] Falconer, S., Storey, M. A cognitive support framework for ontology mapping. In *Proceedings of the 6th International and 2nd Asian Semantic Web Conference*, Busan, Korea, pp. 114-127, 2007.

[48] Falconer, S., Noy, N. F. Interactive techniques to support ontology matching. In Z. Bellahsene, A. Bonifati, E. Rahm (Eds.) *Schema Matching and Mapping*, Springer, Germany, pp. 29-49, 2011.

[49] Euzenat, J., Loup, D., Touzani, M., Valtchev, P. Ontology Alignment with OLA. In *Proceedings of the 3rd Evaluation of Ontology-based Tools Workshop and 3rd International Semantic Web Conference*, Hiroshima, Japan, pp. 59-68, 2004.

[50] Hitzler, P., van Harmelen, F. A reasonable Semantic Web. *In Semantic Web Journal*, 1(1): pp. 39-44, 2010.

[51] Heβ, A., Kushmerick, N. Learning to attach semantic metadata to web services. In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, 2003.

[52] Leighton, H. V., Srivastava, J. First 20 precision among World Wide web search services (search engines). *Journal of the American Society for Information Science*, 50(10), 870-881, 1999.

[53] Do, H. H., Rahm, E. COMA - A system for flexible combination of match algorithms. In *Proceedings of the 28th International Conference on Very Large Data Bases* (VLDB). Hong Kong, China, 2002.

[54] T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993.

[55] de Bruijn, J., Martin-Recuerda, F., Manov, D., Ehrig, M. D4.2.1 state-of-the-art-survey on ontology merging and aligning v1. SEKT Project deliverable D4.2.1. 2004.