2
Fundamentos

Esse capítulo descreve os principais conceitos envolvidos na elaboração do *Matchmaking*. Apesar de, com algumas alterações, a infraestrutura funcionar para qualquer tecnologia de banco de dados, usaremos esquemas OWL. Esses esquemas são arquivos XML com uma especificação bem definida que garante a existência de todos os elementos de um esquema de banco de dados como conhecemos.

A seção 2.1 apresenta a noção de esquema e de sua representação em OWL. A seção 2.2 descreve características gerais de alinhamentos de esquemas e apresenta a técnica de alinhamento que usaremos como exemplo para o *Matchmaking*.

2.1.
RDF/OWL

Klyne et al. [15] definem como *recurso* tudo aquilo que possui alguma identidade, podendo esta ser digital (como um documento eletrônico, uma imagem ou um serviço), física (como um livro) ou uma coleção de outros recursos. Um *Uniform Resource Identifier* (URI) é uma cadeia de caracteres utilizada para identificar um recurso na Web. Uma *URIref* apresenta um dos usos mais comuns para URI, acrescentando um fragmento de identificação opcional precedido pelo caractere "#".

RDF apresenta uma linguagem de descrição que pode ser utilizada de forma padronizada e assim aplicações compartilham e trocam de informações, sem que haja perda de significado. Essas informações podem ser sobre recursos da Web ou objetos que são identificados na Web mas não podem ser recuperados por ela (como pessoas, veículos, etc.). Uma *afirmação* em RDF (ou simplesmente *afirmação*, ou *statement*) é uma tripla (*S*,*P*,*O*), onde

- *S* é um recurso chamado de *sujeito* da afirmação

- *P* é um recurso chamado de *propriedade* (as vezes chamado de *predicado*) da afirmação, ele denota uma relação binária
- *O* pode ser tanto um recurso ou um literal, chamado de objeto de uma assertiva; se *O* é um literal então O também é chamado de *valor* da propriedade *P*

Apesar de RDF possuir uma boa flexibilidade e permitir que consigamos descrever recursos com relativa precisão, apenas a propriedade *rdf:type* possui uma semântica pré-estabelecida. Essa restrição impede que estruturas complexas como classes, hierarquia de classes e propriedades possam ser descritas. Para isso existem as extensões de RDF, como RDF *Schema* [8], que formam um vocabulário reservado utilizado para descrever esses itens de uma aplicação.

Uma *classe*, no RDF Schema, é um recurso que possui a propriedade *rdf:type* com o valor *rdfs:Class*. Por exemplo:

```
Escola rdf:type rdfs:Class
```

Para definir uma classe como *subclasse* de outra, deve-se utilizar a propriedade *rdfs:subClassOf* para relacionar as duas classes. Subclasses são transitivas. Por exemplo:

```
Carro rdfs:subClassOf Veículo
CarroTransporte rdfs:subClassOf Carro
```

e, por transitividade:

```
CarroTransporte rdfs:subClassOf Veículo
```

Uma *propriedade* é qualquer instância da classe *rdfs:Property*. *A* propriedade *rdfs:domain* é usada para indicar que uma propriedade em particular se aplica a uma determinada classe, e a propriedade *rdfs:range* é usada para indicar que os valores de uma propriedade em particular são instâncias de uma determinada classe ou, alternativamente, são instâncias (literais) de um tipo de dados do *XML Schema* (*XML Schema datatype*). Por exemplo:

```
peso rdf:type rdf:Property
peso rdfs:range xsd:double
peso rdfs:domain Veículo
```

A *especialização* de uma propriedade é descrita através de *rdfs:subPropertyOf*. Uma propriedade RDF pode ter zero ou mais sub propriedades; todas as propriedades *rdfs:range* e *rdfs:domain* do esquema RDF que se aplicam a uma propriedade RDF também são aplicadas em cada uma das

suas subpropriedades (subpropriedades têm o mesmo conceito de subclasses. Por exemplo, se a propriedade *motorFlex* for subpropriedade da *motorCombustão*, isso significa que a instância que possui *motorFlex* como propriedade também possui *motorCombustão*).

Uma *instância* de uma classe é um recurso que possui o valor da propriedade *rdf:type* com o valor do URIref da classe correspondente. Um recurso pode ser uma instância de mais de uma classe. Para definir o valor de uma propriedade de uma classe na instância, basta criar uma afirmação como no exemplo:

I. KombiXSR0001 rdf:type CarroTransporte

II. KombiXSR0001 peso 1205,00

No caso, em (I) a instância KombiXSR0001 é criada como sendo da classe *CarroTransporte*. Já em (II) o peso de 1205,00 é atribuído.

OWL[1] é uma linguagem de ontologia Web criada pelo *W3C Web Ontology Working Group* que estende o RDF Schema definindo um vocabulário mais abrangente, fornecendo meios de se tratar relacionamentos, cardinalidades e outras definições complexas. A OWL tem três dialetos: *OWL Lite*, *OWL DL* e *OWL Full*, cada uma com mais expressividade que a anterior. A *OWL Lite*, por exemplo, permite somente cardinalidades de valor 0 ou 1, enquanto a *OWL Full* não faz esse tipo de restrição.

Na OWL, uma classe é qualquer recurso que possua o valor *owl:Class* na propriedade *rdf:type*, sendo que *owl:Class* é uma subclasse de *rdf:Class*.

A OWL separa as propriedades em duas categorias diferentes: *propriedades de objeto* (*object properties*), que relacionam indivíduos a indivíduos, e *propriedades de tipos de dados* (*datatype properties*), que relacionam indivíduos a dados literais. A primeira categoria define os relacionamentos entre as classes. Ambas são subpropriedades de *rdfs:Property*.

Uma propriedade especial, a *owl:sameAs*, define que dois recursos representam o mesmo individuo. Por exemplo, a tripla RDF *(uri1,owl:sameAs,uri2)* define que *uri1* e *uri2* representam o mesmo individuo (ou instância) no banco de dados. A propriedade *owl:equivalentClass* e *owl:equivalentProperty* são análogos à propriedade *owl:sameAs*, mas esses se referem a duas classes e a duas propriedades, respectivamente.

Um banco de dados OWL é um conjunto de triplas no vocabulário da

OWL. Note que o conjunto de triplas pode incluir indistintamente definições relativas ao esquema conceitual do banco de dados e instâncias do banco.

2.2. A técnica para alinhamento de esquemas

No que se segue, assuma que S e T sejam dois esquemas OWL e $\boldsymbol{V_S}$ e $\boldsymbol{V_T}$ seus vocabulários, respectivamente. Assuma também que $\boldsymbol{C_S}$ e $\boldsymbol{C_T}$ sejam os conjuntos de classes e $\boldsymbol{P_S}$ e $\boldsymbol{P_T}$ sejam os conjuntos de propriedades em $\boldsymbol{V_S}$ e $\boldsymbol{V_T}$, respectivamente.

O problema de alinhamento de esquemas é decomposto nos problemas de definir um alinhamento contextualizado de vocabulário (*contextualizated vocabulary matching*) e na definição de mapeamentos de conceitos (*concept mapping*) [20,22].

Um *alinhamento contextualizado de vocabulário entre* S *e* T é um conjunto finito de quádruplas (v_1,e_1,v_2,e_2) de maneira que ou $(v_1,v_2) \in \boldsymbol{C_S}\times\boldsymbol{C_T}$ e $e_1=e_2=$Nulo, ou $(v_1,v_2) \in \mathbf{P_S}\times\mathbf{P_T}$ e e_1 e e_2 são subclasses do domínio, ou os próprios domínios, de v_1 e v_2, respectivamente.

A Tabela 1 ilustra um fragmento de um alinhamento contextualizado de vocabulário. A primeira linha indica que a classe `am:Book` e `eb:Book` são alinhadas, enquanto que a segunda linha indica que as propriedades `am:name` e `eb:publisher` são alinhadas no contexto de `am:Publ` e `eb:Book`, respectivamente.

Para detectar quando duas instâncias denotam o mesmo objeto do mundo real, é preciso ter uma terceira noção. Assuma que $\boldsymbol{U_S}$ e $\boldsymbol{U_T}$ sejam conjuntos de triplas de S e T, respectivamente. Um alinhamento de instâncias de S em T é um conjunto μ_I de quádruplas de modo que, se $(I,C,J,D) \in \mu_I$, então existem triplas $(I,$ rdf:type, $C) \in \mathbf{U_S}$ e $(J,$ rdf:type,$D) \in \mathbf{U_T}$.

Similaridade é um conceito utilizado em várias áreas. Várias funções de similaridade já foram propostas na literatura, como as baseadas em *information content* [31], aquelas baseadas em teoria da informação (*information theory*) [24, 5, 14], *vector model* [11], calculo da distância (*distance measurements*) [16] e *contrast model* [32].

O *contrast model* intuitivamente diz que a similaridade entre dois elementos

aumenta quando a quantidade de valores em comum aumenta, e diminui quando a quantidade de valores comuns diminui.

Já o *vector model* calcula a similaridade entre dois elementos X e Y colocando seus valores dispostos em índices de um vetor. Seja N_X e N_Y como a quantidade de valores que cada um dos elementos possui, teremos dois vetores de N_X e N_Y dimensões. Para calcular a similaridade entre ambos, uma das técnicas é calcular o cosseno do ângulo formado por ambos os vetores.

Neste trabalho usaremos como exemplo o processo de alinhamento em quatro passos definido por Leme [20], definido abaixo:

(1) Gere um alinhamento preliminar de propriedades utilizando funções de similaridade.

(2) Use o conjunto obtido no passo (1) para gerar: (a) o alinhamento de classes; (b) o alinhamento de instâncias.

(3) Use o alinhamento de classes e instâncias obtido no passo (2) para refinar o alinhamento de propriedades.

(4) O alinhamento de vocabulário final é o resultado da união do alinhamento de classes obtido no passo (2) com o alinhamento de propriedades obtido no passo (3), ajustado até que ele se torne estruturalmente correto.

O passo (1) gera um alinhamento preliminar de propriedades baseado na intuição de que “duas propriedades são correspondentes se elas possuem muitos valores em comum e poucos valores não comuns”. Já o passo (2) cria o alinhamento de classes que reflete na intuição de que “duas classes são correspondentes se elas possuem muitas propriedades correspondentes”. Entretanto, para funcionar corretamente, o passo (2) precisa que o passo (1) gere um alinhamento somente para propriedades muito similares.

De modo a testar a ferramenta criada, os passos (1) e (2) foram implementados na infraestrutura. Esses passos englobam o alinhamento de propriedades, de classes e de instâncias.

Tabela 1 - Exemplo de um alinhamento de vocabulário

Amazon		**eBay**	
`am:Book`	`T`	`eb:Book`	`T`
`am:name`	`am:Publ`	`eb:publisher`	`eb:Book`