Uma abordagem estrutural para calcular similaridade entre conceitos de ontologias

Jairo Francisco de Souza[1]

Rubens Nascimento Melo[2]

Jonice Oliveira[3]

Jano Moreira de Souza[4]

Resumo: O problema da compatibilização entre ontologias ainda é um problema aberto. Para tratar esse problema, apresentamos o algoritmo implementado no sistema GNoSIS. Nossa solução faz uso tanto de técnicas sintáticas quanto semânticas numa abordagem estrutural, facilitando o mapeamento ou alinhamento entre ontologias de diferentes domínios. Esse algoritmo se utiliza de diferentes funções de similaridade e calcula o grau de similaridade entre os conceitos de forma recursiva, calculando o resultado da função de similaridade entre dois conceitos com base no grau de similaridade total entre os conceitos que possuem parentesco próximo. Uma validação da abordagem é apresentada neste artigo.

1 Departamento de Ciência da Computação – Universidade Federal de Juiz de Fora {jairofsouza@gmail.com}

2 Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro {rubens@inf.puc-rio.br}

3 Departamento de Ciência da Computação, IM, Universidade Federal do Rio de Janeiro {jonice@cos.ufrj.br}

4 Programa de Engenharia de Sistemas e Computação, COPPE, UFRJ {jano@cos.ufrj.br}

Abstract: The ontology compatibilization issue it's an unsolved problem. To address this problem, we presented the algorithm developed in GNoSIS system. This is solution uses both syntactic and semantic techniques in a structural approach. The algorithm uses distinct resemblance functions and calculates the similarity degree between concepts recursively, calculating the resemblance function result between concepts based on total similarity degree between concepts with closed relationship. A validation of this approach is presented in this article.

1 Introdução

Mesmo utilizando-se de uma estrutura de conhecimento para permitir a interoperabilidade entre sistemas - seja na comunicação entre agentes, na integração de banco de dados ou ainda em outros cenários - ainda assim a interoperabilidade é comprometida quando duas estruturas de conhecimento diferentes são utilizadas.

As divergências ontológicas podem ser divididas em (1) divergências no nível da linguagem (diferenças causadas pelo uso de diferentes formalismos) e (2) divergências no nível da conceituação (diferenças quanto à estruturação dos conceitos na ontologia) [8].

Divergências no nível da linguagem são resolvidas trocando o formalismo de uma ou das duas ontologias. A troca de formalismo também gera novos problemas, como os causados pela diferença de expressividade de um formalismo em relação a outro, mas ainda assim é a solução mais adequada para resolver esse tipo de divergência. Neste trabalho, adotamos a linguagem OWL como padrão de descrição de ontologias e, assim, não tratamos problemas de divergências desse nível.

Divergências no nível da conceituação ocorrem, entre outros casos, pela diferença de codificação, uso de sinônimos, uso de ontologias genéricas distintas, diferença de granularidade entre as ontologias, etc. Esses casos exigem uma comparação da estrutura dos conceitos e do contexto, ou seja, uma comparação semântica. Comparações sintáticas podem adicionar bons resultados às comparações semânticas encontrando relações semânticas entre termos, como acontece em muitos algoritmos que mineram corpo de textos [4, 6].

Para tratar o problema da compatibilidade de ontologias, propomos um algoritmo que faz uso tanto de técnicas sintáticas quanto semânticas numa abordagem estrutural, segundo classificação proposta em [11]. Esse algoritmo se utiliza de diferentes funções de similaridade e calcula o grau de similaridade entre os conceitos de forma recursiva, calculando o grau de similaridade entre dois conceitos com base no grau de similaridade total entre os conceitos que possui parentesco próximo.

Este algoritmo foi implementado como parte do GNoSIS [9, 13, 14], criado para que grupos de especialistas de domínios possam negociar significados de conceitos descritos em ontologias e produzir os mapeamentos entre as ontologias. O GNoSIS recomenda mapeamentos com base no cálculo de similaridade entre os conceitos e, em seguida, os especialistas de domínio negociam a validade do mapeamento e diferenças de conceituação.

Este trabalho apresenta nossa abordagem na seção 2. A seção 3 apresenta o processo de validação da nossa abordagem. Por fim, a seção 4 apresenta os trabalhos correlatos e a seção 5 nossas conclusões, as limitações do trabalho e trabalhos futuros.

2 Algoritmo para cálculo de similaridade

Esta seção apresenta a abordagem desenvolvida no GNoSIS para calcular similaridade de conceitos de ontologias. O módulo implementado recebe pares de ontologias como entrada e retorna uma tabela com os conceitos mais similares. Para preencher essa tabela, o módulo analisa as ontologias de forma sintática (nomes de conceitos e propriedades) e semântica (hierarquia, relacionamentos e *property constraints*). O valor da similaridade total entre dois conceitos é calculado recursivamente com base na hierarquia de conceitos e seus relacionamentos.

O cálculo de similaridade é feito utilizando-se funções de semelhança F(a,b), onde $F_i(a,b)$, i=1,k; F: A x B $\rightarrow$ [0,1], A é o conjunto de elementos da primeira ontologia e B é o conjunto de elementos da segunda ontologia. Por exemplo, $F_1(a,b)$ pode ser uma função como "a quantidade de propriedades com o mesmo nome de um conceito", onde pode retornar o valor 1 caso as ontologias possuam o mesmo número de propriedades com o mesmo nome, 0 caso não possuam nenhuma propriedade idêntica. Valores intermediários representam o grau de semelhança entre os conceitos, quanto maior o valor de F(a,b), maior a semelhança entre os conceitos comparados.

Para cada elemento ontológico (classe, propriedade, restrição, etc), um conjunto de funções de semelhança é aplicado. Em princípio, todos os construtores usados para criar uma ontologia podem ser úteis no cálculo de similaridade e, assim, ter uma ou mais funções de semelhança associadas. Nesta abordagem, utilizamos funções de similaridade aplicadas a nomes, propriedades e relacionamentos de conceitos.

Cada propriedade de um conceito da ontologia possui funções de semelhança que analisam seu nome e tipo. Em OWL, as propriedades podem ser de dado ou objetos, representadas pelos construtores owl:DatatypeProperty e owl:ObjectProperty, respectivamente. Propriedades de dados são aquelas que são intrínsecas do conceito, isto é, que não se relacionam com outros conceitos. Por exemplo, podem ser propriedades de dados para o conceito "Pessoa": nome, idade, tamanho, etc. Por sua vez, propriedades de objetos são aquelas que são extrínsecas ao conceito, i.e., que se relacionam com outros objetos. Para o mesmo conceito "Pessoa", são exemplos de propriedades de objetos: "trabalha em" ou "é filho de".

Propriedades de dados possuem tipos de dados primitivos e propriedades de objetos possuem tipo de dados complexos, ou seja, outros conceitos da ontologia. Assim, por exemplo, a propriedade "nome", citada anteriormente, permite valores como cadeia de caracteres (ou, em OWL, o tipo xsd:string) pois é uma propriedade de dado. Da mesma forma, as propriedades "é filho de" e "trabalha em" permitem valores como instâncias do tipo "Pessoa" e "Local de Trabalho", respectivamente; ambos os tipos são, obrigatoriamente, conceitos descritos na ontologia.

As primeiras das técnicas que usamos como funções de semelhança é a *distância de edição* ou *distância de Levenshtein* [10]. Esta distância recebe duas cadeias de caracteres como entrada e computa a distância entre as cadeias, que é dado pelo número mínimo de inserções, eliminações ou substituições de caracteres que são necessárias para transformar uma cadeia de caracteres em outra. A distância de edição é normalizada. Quanto maior a distância de edição, menor a semelhança entre as cadeias de caracteres. Assim, declaramos uma função de semelhança como:

$$F_{edit}\ a,b = 1 - \frac{Levenshtein\ \ a,b}{max\ \ tamanho\ \ a\ \ ,tamanho\ \ b}$$

Equação 1 – Função de semelhança com distância de edição normalizada

A distância de edição é utilizada para comparação de nomes de conceitos e nomes de propriedades. Também calculamos a semelhança entre os tipos das propriedades (F_T). Por exemplo, definimos o algoritmo F_{NC} abaixo como o algoritmo que calcula a similaridade sintática entre todos os nomes de conceitos das duas ontologias.

Algoritmo 1: Determina a similaridade entre nome de conceitos (NC).

C_A := Conceitos da ontologia A;
C_B := Conceitos da ontologia B;
T := Ø;
para cada $c \in C_A$ **faça**
  para cada $d \in C_B$ **faça**
    $T := T \cup \left[c,d,F_{edit}\ c,d\ \right]$
  fim para
fim para

Similarmente, aplicamos a função F_{edit} para determinar a similaridade entre nome de propriedades de dois conceitos dados, conforme mostra o algoritmo F_{NP} abaixo.

Algoritmo 2: Determina a similaridade entre nomes de propriedades (NP).

P_A := Propriedades do conceito C_A;
P_B := Propriedades do conceito C_B;
T := Ø;

para cada $c \in P_A$ **faça**
  para cada $d \in P_B$ **faça**
    $T := T \cup [c,d,F_{edit}\ c,d\ ]$
  fim para
fim para

A função de semelhança aplicada a nomes de propriedades (F_{NP}) é idêntica à aplicada a propriedades de tipo de objeto e de tipos de dados (vide Equação 1), porém a função de semelhança aplicada aos tipos de propriedades é diferente para as duas. Caso as propriedades possuam como valor tipos de dados primitivos (inteiros, pontos flutuantes, etc) idênticos, então elas receberão valor de semelhança 1. Caso contrário, receberão valor de semelhança 0. Caso as propriedades sejam relacionamentos entre conceitos (propriedade de tipo de objeto), então comparamos o grau de semelhança do tipo da propriedade calculando a distância de edição entre o domínio dessas propriedades. Dessa forma, temos que, se duas propriedades se relacionam com conceitos que são semelhantes, então essas propriedades possuem certo grau de semelhança e, por conseqüência, os dois conceitos que possuem essas propriedades também possuem certo grau de semelhança.

Definimos abaixo o algoritmo F_T abaixo como o algoritmo que calcula a similaridade sintática entre os tipos de propriedades de dois conceitos.

Algoritmo 3: Determina a similaridade entre tipos de propriedades (T).

P_A := Propriedades do conceito C_A;
P_B := Propriedades do conceito C_B;
$T := \emptyset$;
Sim := 0;
para cada $c \in P_A$ **faça**
  para cada $d \in P_B$ **faça**
    se (*c* **ou** *d* são propriedades de tipo de dado primitivo) **e** (*c* é definido com mesmo tipo de dado primitivo de *d*) **então**
      Sim := 1;
    fim se;
    senão
      Sim := $F_{edit}(c.Tipo, d.Tipo)$;
    fim se;
    $T := T \cup [c,d,\mathrm{Sim}]$
  fim para
fim para

Exemplificando as duas funções de semelhança acima, considere os conceitos "Veículo" e "Carro" de ontologias distintas. O conceito "Veículo" possui uma propriedade de tipo de dado chamada "Ano de fabricação" que recebe valores inteiros e uma propriedade

de tipo de objeto chamada “possui” que se relaciona com o conceito “Roda”. Por sua vez, o conceito “Carro”, possui uma propriedade de tipo de dado chamada “Data de fabricação” que recebe valores do tipo data e uma propriedade de tipo de objeto chamada “contém” que também se relaciona com um conceito chamado “Roda”, conforme ilustra a Figura 1. Podemos calcular a função de semelhança de nome de propriedades analisando “Ano de fabricação” e “Data de fabricação”. Ao aplicarmos a distância de edição no nome dessas propriedades, encontraremos o valor (1 – 4 / 18) = 0,78; vide Equação 1. Aplicando a função de semelhança de tipo de propriedades também em “Ano de fabricação” e “Data de fabricação”, encontraremos o valor zero (tipos de dados diferentes). A mesma função de semelhança aplicada às propriedades de tipo de objeto “possui” e “contém” retornará o valor 1 por ser o retorno da distância de edição entre os dois conceitos.

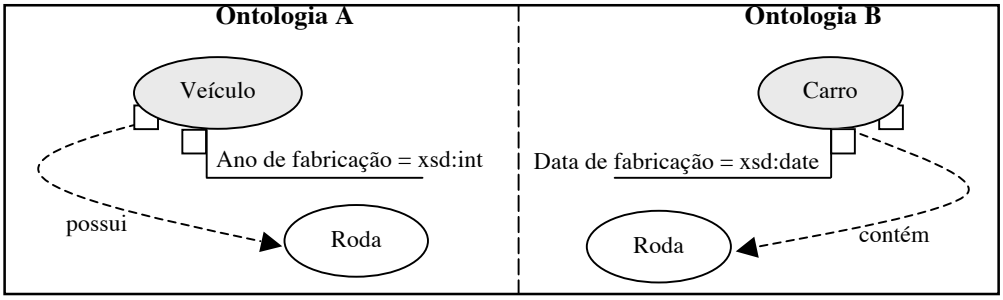

Figura 1 – Trecho de ontologias contendo um conceito e suas propriedades

A hierarquia dos conceitos é utilizada ao comparamos seus filhos e pais, i.e., dois conceitos não-folhas são estruturalmente semelhantes se o conjunto dos seus filhos imediatos é altamente semelhante. A mesma idéia é também usada para os pais imediatos dos conceitos. Essa função de semelhança de hierarquia (F_H) analisa o contexto do conceito, ou seja, para calcular o grau de semelhança de dois conceitos A e B, é necessário calcular o grau de semelhança de seus parentes imediatos. Assim, F_H é calculado levando em consideração a função de semelhança aplicada a superclasses (F_{HP}) e a função de semelhança aplicada a subclasses (F_{HF}). Com estas funções, conceitos raízes, ou seja, conceitos que não possuem superclasses, são computados com grau de semelhança 1. Da mesma forma, conceitos existentes nas folhas, ou seja, conceitos que não possuem filhos (subclasses), são computados com grau de semelhança 1.

Seja β_A e β_B o conjunto das superclasses dos conceitos A e B, respectivamente, e ρ(A,B) o resultado médio da aplicação da função de similaridade aplicada às superclasses de A e B (a função ρ será detalhada posteriormente na Equação 4). A equação abaixo define a função de similaridade F_{HP}. A função de similaridade F_{HF} é definida de forma semelhante, considerando-se β_A e β_B como o conjunto das subclasses de A e B.

$$F_{HP}\ A,B = \left\{ \begin{array}{l} 1, \text{se } \beta_A \cup \beta_B \subset \varnothing \\ o\ A,B\ , \text{caso contrário} \end{array} \right\}$$

Equação 2 – Função de similaridade de superclasses

O grau total de similaridade de dois conceitos (F_X) é calculado através a equação abaixo, proposta por [15]. A equação é usada para avaliar a função de semelhança F_x como o somatório das m funções de semelhança F_q aplicadas aos pares de elementos ontológicos *a* e *b* com seus pesos associados W_q.

$$F_x(a,b) = \sum_{q=1}^{m} F_q(a,b) \times W_q$$

Equação 3 – Somatório das funções de semelhança [13]

A Figura 2 exibe as funções de semelhança (dentro das caixas) e os pesos padrão usados para a comparação entre os conceitos. Os pesos são usados para ajustar o algoritmo e são normalizados para somarem 1. As funções de semelhança retornam 1 para uma semelhança perfeita e um valor positivo menor para pares com menores semelhanças. A equação acima é usada para agregar a informação sobre as sub-semelhanças de baixo pra cima na árvore da figura abaixo, até a raiz. Ou seja, a semelhança final entre dois conceitos de ontologias distintas é dada pela soma das suas funções de semelhança. Os pesos W descritos na figura serão discutidos na seção 3.

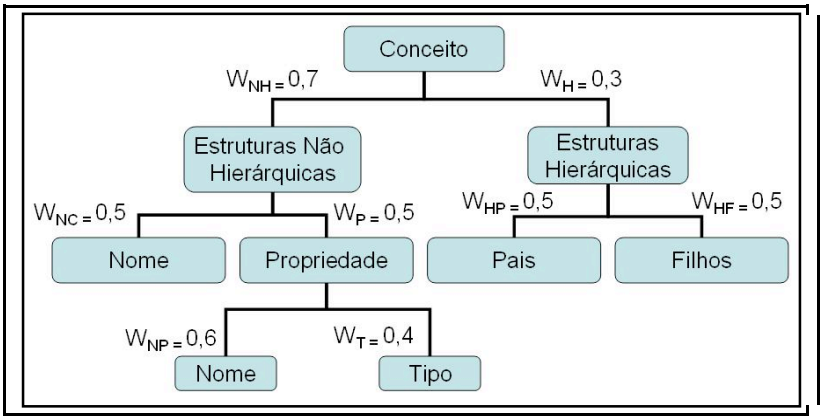

Figura 2 – Funções de semelhança e seus respectivos pesos [13]

Os pesos foram denotados para diferenciá-los quanto ao resultado da função de semelhança a que ele se aplica, seguindo a nomenclatura:

- W_{NP}: Peso aplicado à função de similaridade F_{NP} (nome da propriedade)
- W_T: Peso aplicado à função de similaridade F_T (tipo da propriedade)
- W_P: Peso aplicado à função de similaridade F_P (propriedade)
- W_{NC}: Peso aplicado à função de similaridade F_{NC} (nome do conceito)
- W_{NH}: Peso aplicado à função de similaridade F_{NH} (estruturas não hierárquicas, dada pelo somatório das funções F_{NC} e F_P)
- W_{HP}: Peso aplicado à função de similaridade F_{HP} (superclasses)
- W_{HF}: Peso aplicado à função de similaridade F_{HF} (subclasses)
- W_H: Peso aplicado à função de similaridade F_H (estruturas hierárquicas)

Suponha que dois conceitos sendo comparados possuam uma única propriedade e as funções de semelhança aplicadas ao nome da propriedade (F_{NP}) e ao seu tipo (F_T) retornaram 0,5 e 1 respectivamente. Esta informação contribuirá para o resultado da função de semelhança entre propriedades (F_P) com os pesos 0,6 e 0,4 respectivamente. Dessa forma, F_P será calculada como: 0,5*0,6 + 1 * 0,4 = 0,7.

Para os casos onde existam mais de um elemento relacionado a ser usado nas funções de semelhança F_P, F_{HP} ou F_{HF}, isto é, conceitos que possuem várias propriedades ou conceitos com mais de uma subclasse ou superclasse, o resultado da função de semelhança corresponde à média aritmética do resultado da aplicação da função de semelhança em cada um dos elementos, nos casos onde os dois conceitos possuem o mesmo número de elementos (propriedades, subclasses ou superclasses). Se alguma informação estiver faltando a algum dos conceitos, por exemplo, os conceitos se diferem quanto ao número de propriedades que possui , utilizamos uma técnica para fazer uma média dos pesos. O objetivo é introduzir uma penalidade (equivalente a um peso negativo) que é calculado como segue, onde L_{min} e L_{max} representam o número mínimo e o número máximo de propriedades do conceito (ou filhos ou pais do conceito), respectivamente, e *somaSimilaridades()* é uma função que realiza a soma das similaridades inseridas numa tabela *T* calculadas por uma função de similaridade qualquer.

$$a, b = \frac{somaSimilaridades\ \ T}{L_{min}\ \ Penalidade \times\ \ L_{max} - L_{min}}$$

Equação 4 – Função de semelhança aplicada na hierarquia e propriedades

A penalidade varia de 0 (quando não importa a diferença no número de elementos de um conceito, resultando numa simples média aritmética) até 1 (quando a diferença no número de elementos é importante). Assim, ao quanto mais próximo o número de elementos (uma vez que ρ é aplicado ao cálculo da função de semelhança entre propriedades, superclasses ou subclasses), maior será a similaridade entre os conceitos.

Ao analisarmos o resultado da função aplicada a dois conceitos onde cada conceito possui mais de uma propriedade ou subclasse, caímos no problema de escolher quais relações serão escolhidas como relevantes. Por exemplo, imagine o conceito A e B, com respectivas subclasses [a1, a2] e [b1, b2, b3]. A função de semelhança F_{HF}, por exemplo, será aplicada em toda combinação de subclasses gerando uma matriz como a da **Tabela 1**, no exemplo abaixo ordenada por grau de semelhança:

Tabela 1. Matriz com combinações de semelhança [15]

a1,	b2,	0.8
a2,	b2,	0.7
a1,	b3,	0.6
a2,	b1,	0.4
a1,	b1,	0.3
a2,	b3,	0.3

A cada entrada da tabela acima no formato [*x*, *y*, *z*] lê-se o conceito *x* possui grau de semelhança *z* com o conceito *y*. O resultado da função pode se dar pela escolha da primeira combinação encontrada, sem repetir elementos, onde teríamos [a1, b2, 0.8] e [a2, b1, 0.4] totalizando uma semelhança de 1,20. Um algoritmo que confere todas as combinações para escolher a melhor resposta possível escolheria as combinações [a2, b2, 0.7] e [a1, b3, 0.6] totalizando 1,30. Mesmo em alguns casos retornando um valor inferior, utilizamos a função ***somaSimilaridades(),*** se utiliza da primeira abordagem, pois (1) nossa principal preocupação é encontrar as maiores semelhanças entre os conceitos e encontrar um elemento muito semelhante é mais interessante do que encontrar dois elementos menos semelhantes, mesmo que a soma total das funções seja maior; além disso, (2) encontrar a resposta que retorne o maior valor total é um problema NP completo [15] e pode ser impraticável calcular a resposta quando a quantidade de elementos for muito elevada.

Após o cálculo das semelhanças, os conceitos que satisfazem a equação abaixo são listados ao usuário como altamente similares. Na avaliação descrita na seção 3, consideramos o valor de Ω como 0,65 (ver seção 5 para uma discussão sobre o valor aplicado).

$$\sum_{q=1}^{m} F_q(a,b) \times W_q \geq \Omega$$

Equação 5 – Regra para selecionar conceitos altamente similares

O algoritmo 4 resume o cálculo de similaridade implementado neste trabalho. Ao fim da execução do algoritmo, temos uma tabela de similaridades entre todos os conceitos da ontologia. Vale ressaltar que, embora tenhamos definido diferentes funções de similaridade que são calculadas de forma composta (isto é, para calcular $F_{NH}(c,d)$ é necessário, antes, calcular F_{NC} e F_P. Para este último, por sua vez, é necessário calcular F_T e F_{NP}), pode-se definir diversas outras funções de similaridades e utilizá-las para compor a função de

similaridade total F_x utilizando-se de outras técnicas para analisar demais elementos presentes nas estruturas a serem comparadas, como comparação entre instâncias de conceitos, descrições de conceitos, axiomas, etc.

Algoritmo 4: Gera tabela de similaridade entre dois conceitos

$T := \emptyset$;
$F_X := 0$;
C_A := {Conjunto de conceitos da ontologia A};
C_B := {Conjunto de conceitos da ontologia A};
para cada $c \in C_A$ **faça**
  para cada $d \in C_B$ **faça**
    $F_X = F_{NH}(c,d) * W_{NH} + F_H(c,d) * W_H$
    se F_X **>=** Ω **então** $T := T \cup [c,d,F_x]$;
  fim para
fim para

3 Avaliação

Para avaliarmos essa abordagem, três comparações distintas são realizadas. Na primeira delas, é comparado trecho da ontologia que descreve o domínio de programação orientada a objetos com uma cópia dela que teve os nomes dos conceitos arbitrariamente alterados. Assim, temos o intuito de avaliar que o algoritmo funciona para o caso em que as duas ontologias possuem estruturas idênticas e nomes diferentes. A segunda comparação será feita alterando-se a estrutura da segunda ontologia e com nomes de conceitos quase idênticos. Por fim, a terceira comparação será feita alterando-se novamente os nomes da segunda ontologia, de forma arbitrária, representando assim diferenças grandes de conceituação entre as duas ontologias. Este último é o cenário de pior caso em que o algoritmo poderá trabalhar. As ontologias a serem comparadas estão descritas na Figura 3.

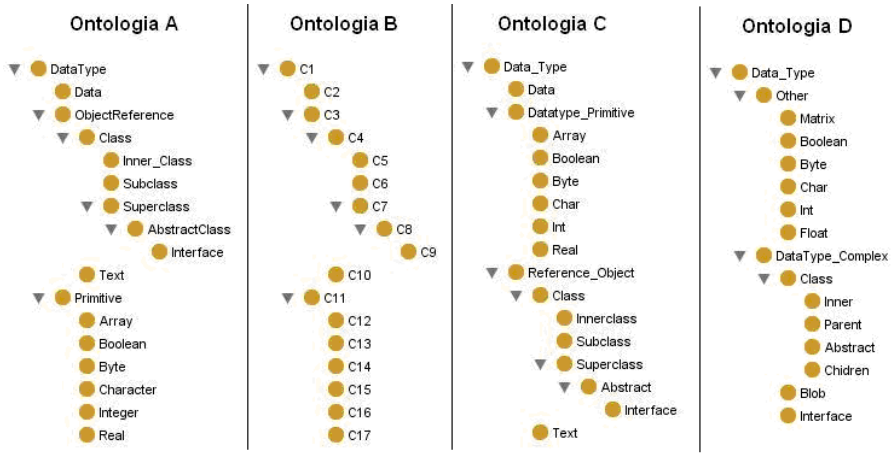

Figura 3 – Ontologias utilizadas na validação do algoritmo

Todos os pesos aplicados às funções de semelhança da Figura 2 foram definidos arbitrariamente. Possíveis soluções para definição dos pesos das funções são discutidas na seção 5.

2.1 Comparando ontologias com termos distintos

No nosso primeiro cenário, iremos comparar as ontologias A e B (Figura 3) que possuem a mesma estrutura (propriedades e hierarquia), contendo somente diferenças entre os nomes dos conceitos que foram arbitrariamente alterados. Como toda representação gráfica de ontologias, existe perdas semânticas nesta visualização, uma vez que não conseguimos visualizar outros aspectos que não sejam os hierárquicos, como, por exemplo, as propriedades de dados e de objetos desses conceitos. Para visualizar todas as informações das ontologias, disponibilizamos os arquivos OWL em [12].

A janela abaixo (Figura 4) exibe a lista de todos os conceitos com maiores graus de similaridade calculado para o conceito ObjectReference, acima do limite mínimo de 0,65 (campo *threshold*) e ordenado por grau de similaridade. O conceito com maior grau de similaridade encontrado foi C3, com valor aproximado de 0,805. Ao analisarmos a Figura 3, verificamos que esse conceito corresponde realmente ao conceito ObjectReference. Os outros conceitos listados na lista dos mais similares a ObjectReference são C1 e C11, com valores aproximados de 0,70 e 0,67, respectivamente. Conforme pode ser verificado na Figura 3, C1 é o correspondente ao conceito DataType (pai de ObjectReference) e C11 ao conceito Primitive (irmão de ObjectReference), ou seja, conceitos próximos na sua estrutura (tanto hierárquicas quanto nas propriedades herdadas).

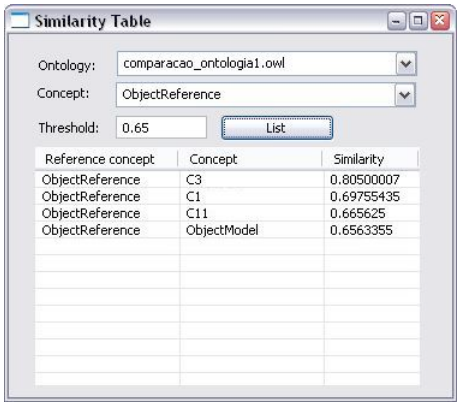

Figura 4 – Conceitos mais similares de ontologiaA#ObjectReference

A **Tabela 2** descreve os resultados do algoritmo com os valores encontrados para cada dupla de conceitos. A tabela foi montada com cada conceito da ontologia A e com o conceito mais similar apontado pelo algoritmo. Nota-se que ao recuperarmos os conceitos mais similares a cada conceito da primeira ontologia, o algoritmo acerta em todos os casos.

Tabela 2. Grau de similaridade entre os conceitos mais similares

Ontologia A	Ontologia B	Grau de similaridade
Datatype	C1	0,85
Data	C2	0,85
ObjectReference	C3	0,805
Class	C4	0,711
Inner_Class	C5	0,869
Subclass	C6	0,823
Superclass	C7	0,864
AbstractClass	C8	0,761
Interface	C9	0,871
Text	C10	0,85
Primitive	C11	0,801
Array	C12	0,85
Boolean	C13	0,85
Byte	C14	0,85
Character	C15	0,85
Integer	C16	0,85
Real	C17	0,85

Na implementação feita do algoritmo, é possível discriminarmos os valores das principais funções de semelhanças aplicadas. A

Tabela 3 mostra os valores calculados para as funções de semelhanças aplicados aos conceitos ObjectReference e C3. Nota-se que a similaridade entre os dois conceitos não foi total (valor 1) por conta da sua diferença entre os termos *(conceptName* = 0, denotando o resultado da função F_{NC}). Tal diferença acaba por influenciar, em certo grau, as funções de

semelhança aplicadas sobre a hierarquia (F_H, abaixo denotada por *HierarchySim),* sobre dados não-hierárquicos (F_{NH}, abaixo denotada por *ContextFreeVariableSim)* e, é claro, o grau de similaridade total (F_X, abaixo denotada por *ConceptSim*). Por outro lado, estes valores de semelhança possuem valores superiores ao considerado aceitável (limite inferior de 0,65) por causa da alta semelhança nas suas estruturas, o que pode ser notado de forma mais clara no valor 1 retornado pela função de semelhança aplicada às propriedades dos conceitos (F_P, abaixo denotada por *PropertySim*).

Tabela 3. Similaridades de [ObjectReference, C3] separadas por função de semelhança

```
Command: h
-> Analise: proccess sintatic and semantic analysis.
-> Exit: terminate the analysis.
-> Help: show help.
-> List: list the concepts most similars to a chosen concept.
-> Load: load ontologies for semantic analysis.
-> Quit: terminate the analysis.
-> Repository: show ontologies saved in repository.
-> ShowAllCellSims: show all similarities from two chosen concepts.
-> ShowConceptSims: show all concepts similarities.
-> ShowCFVSims: show context free variables similarities.
-> ShowPropertySims: show properties similarities.
-> ShowStructure: show tree structure of a chosen ontology.
-> LaunchTable: shows similarity table


Command: ShowAllCellSims
First concept name: ObjectReference
Second concept name: C3

+++Showing similarities
PropertySim: 1.0
ConceptName: 0.0
ContextFreeVariableSim: 0.8
HierarchySim: 0.81000006
ConceptSim: 0.80500007

Command:
```

Este caso de diferença clara entre os nomes das duas ontologias pode ser resolvido ao alterarmos o valor dos pesos aplicados pelo algoritmo nas funções de semelhança. Ao aplicarmos o valor 0 (zero) para as funções de semelhança aplicadas ao nome dos conceitos e ao nome das propriedades, teremos, neste caso, valores de similaridade mais reais. A **Tabela 4** exibe os dados atualizados, recuperados ao aplicarmos tal alteração nos pesos das funções de semelhança.

Tabela 4. Grau de similaridade após anular funções de semelhança de nomes

Ontologia A	Ontologia B	Grau de similaridade
Datatype	C1	1,00
Data	C2	1,00
ObjectReference	C3	1,00
Class	C4	1,00
Inner_Class	C5	1,00
Subclass	C6	1,00
Superclass	C7	1,00
AbstractClass	C8	1,00
Interface	C9	1,00
Text	C10	1,00
Primitive	C11	1,00
Array	C12	1,00
Boolean	C13	1,00
Byte	C14	1,00
Character	C15	1,00
Integer	C16	1,00
Real	C17	1,00

2.1 Comparando ontologias com estruturas distintas

No segundo cenário, iremos comparar a ontologia A com a ontologia C (Figura 3). Esta última ontologia possui seus nomes quase idênticos aos da ontologia A e sua estrutura foi levemente alterada ao modificarmos arbitrariamente algumas propriedades.

Para uma visão mais clara das diferenças entre as duas ontologias, analisemos a Figura 5. A figura ilustra as propriedades entre os conceitos "Class" presentes nas duas ontologias. Considerando somente as linhas que ligam o conceito "Class" aos outros conceitos da ontologia, verificamos que existem diferenças consideráveis entre as duas estruturas. Na ontologia A, o conceito "Class" possui três propriedades: *instancia*, *implementa* e *contém*. A propriedade "instancia" se relaciona com o conceito "Object", a propriedade "implementa" se relaciona com o conceito "Inteface" e a propriedade "contém" se relaciona com os conceitos "Comment", "Field", "Method", "Datatype", "Command", "Heritage", "Declaration" e "Package". Na ontologia C, por sua vez, o conceito "Class" possui somente duas propriedades: *implementa* e *contém*. Assim como na ontologia A, a propriedade "implementa" da ontologia C se relaciona com o conceito "Interface". Porém, a propriedade "contém" se relaciona somente com conceitos "Declaration", "Package", "Field" e "Data_Type".

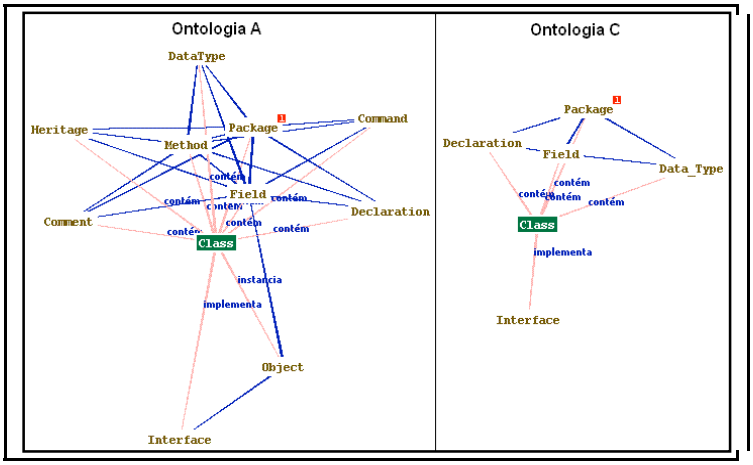

Figura 5 – Diferença nas propriedades do conceito “Class” das duas ontologias

A **Tabela 5** representa os maiores valores obtidos após o cálculo de similaridade para cada dupla de conceitos, considerando-se os pesos-padrão originais (Figura 2). Nota-se que, em algumas duplas, os valores calculados são menores que os da **Tabela 2**, mas, ainda assim, o algoritmo acerta em todos os casos.

Tabela 5. Grau de similaridade entre os conceitos mais similares

Ontologia A	**Ontologia C**	**Grau de similaridade**
Datatype	Data_type	0,782
Data	Data	0,747
ObjectReference	Object_Reference	0,784
Class	Class	0,651
Inner_Class	InnerClass	0,61
Subclass	Subclass	0,664
Superclass	Superclass	0,66
AbstractClass	Abstract	0,602
Interface	Interface	0,631
Text	Text	0,85
Primitive	Datatype_Primitive	0,805
Array	Array	0,773
Boolean	Boolean	0,776
Byte	Byte	0,773
Character	Char	0,63
Integer	Int	0,63
Real	Real	0,773

A diferença entre os nomes dos conceitos fez com que o grau de similaridade calculado fosse menor nestes conceitos do que nos que possuem nomes iguais. Contudo, o

determinante para os baixos graus de similaridade encontrados foram os valores retornados pelas funções de semelhança que levam em consideração a estrutura hierárquica e as propriedades dos conceitos, tendo essa última o pior valor. A **Tabela 6** exibe os valores retornados de algumas funções de semelhança ao comparar os conceitos Class das duas ontologias. Nesta tabela, o valor de semelhança entre as propriedades dos conceitos (*PropertySim*) se mostrou consideravelmente baixo (0,2). Esta função de similaridade leva em consideração, entre outras coisas, o nome e tipo das propriedades dos conceitos comparados e aplica uma penalidade nos casos em que os conceitos se diferem quanto o número de propriedades (Equação 4). Como uma das ontologias teve o nome, tipo e quantidade das propriedades de seus conceitos alterados, a função de similaridade retornou um valor baixo de similaridade quanto as propriedades dos conceitos. A penalidade padrão aplicada neste estudo foi de 0,3. Discussões sobre os pesos e penalidades aplicadas podem ser encontradas na seção 5.

Tabela 6. Similaridades de [Class, Class] separadas por funções de semelhança

```
Command: showAllCellSims
First concept name: Class
Second concept name: Class

+++Showing similarities
PropertySim: 0.2
ConceptName: 1.0
ContextFreeVariableSim: 0.65701
HierarchySim: 0.6284756
ConceptSim: 0.6512378

Command:
```

Sabendo desse impacto causado pela aplicação das funções de semelhança de propriedade aplicadas neste cenário, iremos alterar o peso aplicado a essas funções para zero. Desta forma, a diferença entre as propriedades dos conceitos não será mais determinante para o grau de similaridade total dos conceitos. A **Tabela 7** contém os graus de similaridade encontrados ao alterar o peso desta função de semelhança. Nota-se que os valores encontrados são muito próximos dos contidos na **Tabela 4**, conforme esperado. Tal diferença de valores se dá pela pequena diferença de nomes de conceitos que ainda persiste nas duas ontologias.

Tabela 7. Graus de similaridade após anular funções de semelhança de propriedades

Ontologia A	**Ontologia C**	**Grau de similaridade**
Datatype	Data_type	0,982
Data	Data	0,991
ObjectReference	Object_Reference	0,982
Class	Class	0,991
Inner_Class	InnerClass	0,982
Subclass	Subclass	0,991
Superclass	Superclass	0,991
AbstractClass	Abstract	0,963
Interface	Interface	0,977
Text	Text	0,991
Primitive	Datatype_Primitive	0,918

Array	Array	0,977
Boolean	Boolean	0,975
Byte	Byte	0,977
Character	Char	0,946
Integer	Int	0,957
Real	Real	0,977

2.1 Comparando ontologias com nomes e termos distintos

Para o terceiro cenário deste estudo, foi feita uma comparação entre a ontologia A e a ontologia D (Figura 3). A ontologia D foi obtida através de alterações na ontologia C, onde seus conceitos foram alterados sintática (nomes dos conceitos e relações) e estruturalmente (diferentes relacionamentos e hierarquia). Esse é o caso mais difícil em que o algoritmo tem que trabalhar. Além das diferenças de nomes dos conceitos, a ontologia D difere também da ontologia A quanto à sua estrutura (contém diferenças nas propriedades e, desta vez também, na árvore hierárquica). Este cenário representa duas ontologias com alta divergência de conceituação, i.e., diferenças nas suas visões de mundo. Nota-se diferenças nas ontologias quanto ao nome dos conceitos e a diferença na hierarquia (conceito "Interface" não é mais filho de "AbstractClass" como era na ontologia A, etc). A Figura 3 novamente omite as diferenças quanto às propriedades dos conceitos. Contudo, essas diferenças foram aplicadas da mesma forma arbitrária que no segundo cenário.

A **Tabela 8** descreve o grau de similaridade de cada conceito das ontologias A e D. Desta vez, o ajuste dos pesos não é uma tarefa simplória, uma vez que existem divergências tanto sintáticas quanto semânticas. A maioria dos conceitos mais similares não ultrapassou o limite inferior considerado razoável (*threshold* = 0,6). Contudo, reduzindo-se o limite inferior para 0,3, verificamos que o algoritmo acertou na maioria dos casos, embora tenha retornado duplas indesejadas em alguns conceitos. Os graus de similaridade destacados com um número entre parênteses denota qual a posição em que o conceito correto apareceu. Por exemplo, [Inner_Class, Inner, 0,345 (2)] quer dizer que o grau de similaridade entre os conceitos corretos "Inner_Class" e "Inner" foi de 0,345 e que "Inner" é o segundo conceito mais similar a "Inner_Class", ou seja, o algoritmo encontrou erroneamente um conceito mais similar a "Inner_Class" que "Inner".

Tabela 8. Graus de similaridade entre os conceitos mais similares.

Ontologia A	**Ontologia D**	**Grau de similaridade**
Datatype	Data_type	0,924
ObjectReference	DataType_Complex	0,768
Class	Class	0,320
Inner_Class	Inner	0,345 (2)
Subclass	Children	0,569
Superclass	Parent	0,562
AbstractClass	Abstract	0,651
Interface	Interface	0,353 (3)
Text	Blob	0,856 (3)
Primitive	Other	0,879
Array	Matrix	0,472

Boolean	Boolean	0,651
Byte	Byte	0,802
Character	Char	0,683
Integer	Int	0,699 (2)
Real	Float	0,576 (5)

Dos 16 conceitos separados para análise no último cenário, o algoritmo acertou em 11 casos. Nos outros casos deste cenário, o algoritmo colocou o conceito esperado em até quinto lugar na lista dos mais similares, como pode ser visto na entrada [Real, Float, 0,576 (5)]. Os conceitos que não foram devidamente encontrados pelo algoritmo correspondem aos que possuem maior divergência quanto a sua conceituação. Tais conceitos são mesmo muito complexos de identificar e a intervenção humana se torna imprescindível.

4 Trabalhos relacionados

Abordagens para calcular similaridade entre conceitos vem sendo estudada por diversos pesquisadores [5]. Algumas propostas tentam resolver o problema utilizando comparações semânticas [1] ou sintáticas. A nossa abordagem se utiliza de diferentes funções de similaridade em conjunto. A principal contribuição dessa proposta está na separação das funções de semelhança para cada grupo de elementos da ontologia de forma recursiva sobre os conceitos, uma implementação atualmente em uso no sistema GNoSIS e na validação da abordagem.

O mapeamento de ontologias baseado em instâncias é um conjunto promissor de soluções da classe problemas de alinhamento de ontologias. Algumas soluções foram propostas utilizando essa abordagem [3, 7, 16]. A utilização do ABox para calcular similaridade entre conceitos, contudo, apresenta problemas que não foram tratados neste artigo. Um dos problemas em se utilizar instâncias está em descobrir esse conjunto de instâncias, uma vez que o ABox pode ser descrito tanto como instâncias em OWL na própria ontologia quanto como instâncias em outros arquivos OWL, como anotações em página HTML, documentos de texto, etc. Como o sistema GNoSIS, no qual a nossa abordagem de alinhamento foi construída, não armazena instâncias, não foi criado uma função de similaridade correspondente. Todavia, a árvore de funções da Figura 2 pode ser alterada para receber novas funções de semelhança, as quais serão acrescentadas na Equação 3. De fato, a implementação do algoritmo para o sistema GNoSIS [13] prevê que novas funções de semelhança sejam adicionadas implementando-se uma interface comum.

5 Conclusões

Este trabalho apresentou uma abordagem estrutural para alinhamento de ontologias utilizando comparações hierárquicas, léxicas e relacionais entre os conceitos das ontologias. O algoritmo apresentado realiza um cálculo recursivo sobre os conceitos das ontologias onde a função parcial de similaridade entre subclasses ou superclasses de um conceito depende do

cálculo da função total de similaridade entre esses dois conceitos. Apesar do maior grau de complexidade do algoritmo, as avaliações realizadas mostrou que essa abordagem pode ser utilizado de forma eficiente em estruturas ontológicas com divergências conceituais de grau mediano de complexidade.

O algoritmo foi implementado no sistema GNoSIS [9, 14], o qual recomenda mapeamentos com base no cálculo de similaridade entre os conceitos e, em seguida, os especialistas de domínio negociam a validade do mapeamento e diferenças de conceituação.

O estudo apresentado para validação do algoritmo definiu pesos para cada função de similaridade e uma penalidade aplicada a funções de similaridade usadas em comparações hierárquicas e relacionais (propriedades). Os pesos, a penalidade e o *threshold* foram definidos arbitrariamente levando em consideração testes realizados com diferentes ontologias. Embora os valores dos pesos tenha sido definidos de forma empírica, o algoritmo apresenta bons resultados em ontologias que possuem elementos disponíveis para comparação, isto é, ontologias com uma árvore hierárquica definida, propriedades com tipos definidos, etc. Nos casos em que as ontologias se diferem muito em um dos elementos, isto é, uma ontologia com uma árvore hierárquica extensa e outra sem definição de hierarquias, é possível ajustar os pesos para um melhor resultado. A questão da calibragem é um problema que será tratado em estudos futuros dessa abordagem a fim de diminuir o tempo gasto testando diferentes pesos arbitrariamente. Como proposta, tem-se a utilização de uma base histórica de mapeamentos definidos pelos usuários no GNoSIS para retroalimentar os pesos das funções de similaridade.

Quanto ao valor do *threshold* aplicado, esse valor pode ser visto como um intervalo de confiança esperado para o resultado da função de semelhança total e deve ser menor quanto mais distintas forem as ontologias, uma vez que o valor da função de semelhança entre os conceitos tende a ser menor. No cenário em que o GNoSIS trabalha, um especialista deve especificar o valor do intervalo de confiança aceitável para que possa considerar como válidos os resultados do algoritmo e basear suas decisões quanto os alinhamentos que serão implementados levando em consideração o valor calculado pela função de similaridade total. Para automatizar o processo é possível ajustar o *threshold* utilizando a base histórica de mapeamentos do GnoSIS.

Outra limitação do algoritmo está relacionada com o domínio das ontologias a serem comparadas. Uma vez que ontologias de diferentes domínios normalmente possuem uma divergência maior no nível da conceituação, a sua estrutura hierárquica, propriedades e termos de conceitos podem ser tão distintas que inviabilizam os tipos de funções de similaridade utilizadas neste trabalho. Como trabalho futuro, está previsto a inserção de funções de similaridades para comparação entre as instâncias e descrições dos conceitos, bem como um método para reconhecer domínios de ontologias com o objetivo de aumentar a aplicabilidade do algoritmo na descoberta de similaridade entre conceitos de ontologias mais complexas.

Uma vez que agentes de software terão a responsabilidade de lidar com diferentes ontologias na Web Semântica [2], uma implementação do sistema GNoSIS que substitui

usuários humanos por agentes de software está sendo projetada [17] e esse algoritmo será utilizado para negociação de significado entre agentes durante sua comunicação.

6 Referências

[1] Bouquet, P., Serafini, L. and Zanobini, S., Semantic Coordination: A New Approach and an Application. in *International Semantic Web Conference*, (Flórida, EUA, 2003), 130-143.

[2] Breitman, K., Casanova, M.A. and Truszkowski, W. *Semantic Web: Concepts, Technologies and Applications*. Springer-Verlag, London, 2007.

[3] Breitman, K.K., Brauner, D., Casanova, M.A., Milidiú, R., Gazola, A. and Perazolo, M., Instance-Based Ontology Mapping. in *Fifth IEEE Workshop on Engineering of Autonomic and Autonomous Systems*, (Bari, Itália, 2008), 67-74.

[4] Chakrabarti, S. Data mining for hypertext: a tutorial survey. *ACM SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining*, *1* (2), 2000, 1-11.

[5] Euzenat, J. and Shvaiko, P. *Ontology Matching*. Springer-Verlag, Berlin 2007.

[6] Faatz, A. and Steinmetz, R. Ontology Enrichment with Texts from the WWW. *Semantic Web Mining 2nd Workshop at ECML/PKDD-2002*, 2002.

[7] Isaac, A., Meij, L.v.d., Schlobach, S. and Wang, S. An Empirical Study of Instance-Based Ontology Matching. *The Semantic Web*, *4825/2008*, 2008, 253-266.

[8] Klein, M., Combining and Relating Ontologies: An Analysis of Problems and Solutions. in *Workshop on Ontologies and Information Sharing at the 17th International Joint Conference on Artificial Intelligence*, (Seattle, USA, 2001), 53-62.

[9] Oliveira, J., Souza, J.F., Paula, M. and Souza, J.M.d. A Business-Based Negotiation Process for Reaching Consensus of Meanings. in Shen, W., Chao, K.-M., Lin, Z., Barthès, J.-P.A. and James, A. eds. *Computer Supported Cooperative Work in Design III*, Springer, China, 2007, 561-569.

[10] Shvaiko, P., A classification of schema-based matching approaches. in *Meaning Coordination and Negotiation workshop at International Semantic Web Conference (ISWC)*, (Hiroshima, Japão, 2004).

[11] Shvaiko, P. and Euzenat, J. A survey of schema-based matching approaches. *Journal on Data Semantics*, *IV*, 2005, 146-171.

[12] Souza, J.F. GNoSIS. 2009. Available at <www.ufjf.br/jairo_souza/pesquisa/gnosis>, accessed in março, 2010.

[13] Souza, J.F. Negociação de significado para viabilizar interoperabilidade semântica *PESC/COOPE*, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007, 184.

[14] Souza, J.F., Paula, M., Oliveira, J. and Souza, J.M., Meaning Negotiation: Applying Negotiation Models to Reach Semantic Consensus in Multidisciplinary Teams. in *Group Decision and Negotiation*, (Karlsruhe, Alemanha, 2006), Universität Karlsruhe, 297-300.

[15] Souza, J.M. Software Tools for Conceptual Schema Integration, University of East Anglia, 1986.

[16] Todorov, K. and Geibel, P., Ontology Mapping via Structural and Instance-based Similarity Measures. in *3rd International Workshop on Ontology Matching (OM-2008)*, (Karlsruhe, Germany, 2008).

[17] Souza, J. F., Siqueira, S. W. M., Melo, R. N. . Adding Meaning Negotiation Skills in Multiagent Systems. In: *IEEE International Conference on Intelligent Computing and Intelligent Systems*, Shangai., v. 1. p. 663-667, 2009.