Cloves Langendorf Barcellos Junior

Busca Semântica para a Extração e Indexação do Conhecimento Aplicada a Informações Médicas

Florianópolis, Santa Catarina

30 de novembro de 2008

Cloves Langendorf Barcellos Junior

Busca Semântica para a Extração e Indexação do Conhecimento Aplicada a Informações Médicas

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação

Orientador:
Aldo Von Wangenheim

Co-orientador:
Rafael Andrade

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Florianópolis, Santa Catarina

30 de novembro de 2008

Abstract

The constant search for better results in text searches has motivated new researches focused on applying new methods and algorithms to obtain better understanding of the semantic meaning by the hardware. This document presents an application proposal specialized in semantic search in medical findings, using PostgreSQL databases indexed by Ispell lexemes.

The result is a unified, sharable structured representation of medical findings integrating the representations of findings in SNOMED CT, MeSH and DeCS. Techniques of indexation and storage of large volumes of documents of literal content, as well as integration of data will be approached.

Keywords: Semantic Search, Medical Informatics, SNOMED CT, MeSH e DeCS.

Resumo

A constante procura por melhores resultados em buscas textuais, tem motivado novas pesquisas focadas em aplicar novos métodos e algortimos tentando obter uma melhor "compreensão" do significado semântico pelo *hardware*. Este trabalho apresentará um modelo de aplicação especializada em efetuar busca semântica em laudos médicos.

Utilizando banco de dados PostgreSQL indexadas com lexemas provenientes de dicionários Ispell por meio do índice invertido GIN. Isso resulta em laudos mais claros, com informações mais detalhadas, buscas com resultados mais relevantes que os de uma pesquisa por palavras-chave. Além disso, cria-se a possibilidade de interagir entre bases de dados que tratam de informações de mesma natureza. São abordadas técnicas de indexação e armazenamento de grandes volumes de documentos de conteúdo textual, como também integração de dados provenientes das ontologias SNOMED CT, MeSH e DeCS.

Palavras-chave: Busca Semântica, Informática Médica, SNOMED CT, MeSH e DeCS.

Sumário

Lista de Figuras

Lista de Abreviaturas e Acrônimos

BIREME - Biblioteca Virtual em Saúde

CDR - Cover Density Ranking

DeCS - Descritores em Ciências da Saúde

GIN - Generalized Inverted Index

GiST - The GiST Indexing Project

Ispell - spelling checker

MeSH - Medical Subject Headings

NHS - UK National Health Service

NLM - U.S. National Library of Medicine

PHP - Scripting Language

PostgreSQL - Global Development Group

W3C - World Wide Web Consortium

SNOMED - Systematized Nomenclature of Medicine

SNOMED CT - Systematized Nomenclature of Medicine Clinical Terms

Tsearch2 - PostgresSQL Full Text Search Extension

VSM - Vector Space Model

1 Introdução

Estamos vivenciando uma era da informação que vem sendo impulsionada pela necessidade imposta pelo cotidiano de um mundo globalizado, e principalmente pelo avanço tecnológico atingido até o momento. Dentre esses avanços tecnológicos, são notáveis o impacto causado pela Informática e pelas Telecomunicações. Estes vêm gerando um aumento exponencial do volume da informação que é potencializado pelo acesso interativo *on-line* das informações presentes no mundo todo.

Em uma recente reportagem do jornal espanhol *El Pais* datada de sete de março de 2007, esse aumento do volume de informação digital no mundo é tratado um trecho se destaca: "Há 900 milhões de computadores, 550 milhões de reprodutores de música digital, 600 milhões de telefones celulares com câmera, 400 milhões de câmeras... A informação digital que todos estes dispositivos contêm a soma de 161 bilhões de gigabytes, o equivalente a três milhões de vezes a informação contida em todos os livros já escritos até hoje ou a 12 pilhas de livros que cobririam, cada uma delas, a distância entre a Terra e o Sol. E em três anos, essa cifra se multiplicará por seis, até os 988 bilhões de gigabytes".

O desenvolvimento e a utilização de padrões para dados produzidos na área de saúde são guiados pela necessidade de representar de forma coerente e precisa a informação, garantir o armazenamento, a recuperação efetiva desses dados e principalmente, permitir a interoperabilidade entre diferentes sistemas de informação. A padronização da linguagem utilizada é um grande desafio da informática em medicina, devido à complexidade inerente à linguagem e às informações envolvidas. Enquanto grandes esforços têm sido aplicados há décadas para o desenvolvimento de terminologias controladas (ontologias), a adaptação, a recuperação e a indexação ainda não podem ser utilizadas como única fonte de informação. Embora clinicamente ricos e praticamente globais, o poder expressivo destas ontologias pode ser uma limitação, uma vez que oferece mais do que uma forma de representar um determinado conceito (JUNIOR et al., 2008).

Neste contexto, nem sempre é possível obter um resultado de busca isento de conteúdos

não pertinentes, podendo o conteúdo indesejado torna-se grande a ponto de dissolver o objetivo incialmente esperado. Surge então a necessidade e mecanismos eficientes que organize a forma como disponibilizar a informação nesse imenso acervo. Com isso é possivel obter a eficácia na recuperação da informação. Além disso, existe o problema de falta de interoperabilidade entre as diversas ontologias, mesmo quando tratamos de ontologias de um conteúdo de mesma natureza.

A apresentação desse trabalho está estruturada da seguinte forma: no capitulo 1 descreve a introdução do trabalho, no capítulo 2 serão apresentados os objetivos, no capítulo 3 está descrita a justificativa dessa abordagem. Já no capítulo 4 será abordado um resumo sobre toda revisão bibliográfica que foi utilizada durante o processo de desenvolvimento da metodologia. No capítulo 5 é descrito passo a passo todo o processo de implementação desse trabalho, desde a coleta de dados até a aplicação final.

Os resultados obtidos na aplicação das técnicas e ferramentas descritas no capítulos 4 e 5, serão apresentados no capítulo 6. Conclusões e trabalhos futuros da abordagem descrita são apresentados no capítulo 7.

2 *Objetivos*

2.1 Objetivo Geral

Conceber uma metodologia adequada e capaz de efetuar busca semântica inteligente com suporte textual completo e indexação em informações clínicas, que contenha os termos das ontologias SNOMED CT, MeSH e DeCS.

2.2 Objetivos Específicos

Os objetivos específicos são:

- integrar as ontologias SNOMED CT, MeSH e DeCS.
- convergir diferentes ferramentas de tratamento de informações textuais.
- personalizar os dicionários Ispell adicionando as *stop words*.
- desenvolver algoritmos de busca semântica para extração do conhecimento em laudos médicos.

3 *Justificativa*

A possibilidade de um médico ter acesso às informações de seus pacientes em qualquer setor do hospital permite a ele uma maior mobilidade e facilidade na execução de suas tarefas. Um usuário médico, quando da necessidade de informações para diagnóstico ou atendimento ao paciente poderá utilizar um navegador de Internet para acessar bases de dados que contém informações clínicas sobre sintomas, diagnósticos e procedimentos.

As particularidades encontradas na redação de um laudo médico podem dificultar a compreensão por outro profissional das informações nele contidas. Para evitar a discrepância entre as interpretações de um laudo médico, pode-se então criar índices e indexar as informações contidas nesses laudos, e posteriormente armazená-los em um banco de dados indexado pelo dicionário do idioma em que esse laudo foi redigido. Ao selecionar sentenças (trechos) desse laudo e utilizá-las para inquirir sobre as ontologias, parte destas particularidades podem ser isoladas, e um algoritmo semântico pode se concentrar somente nos termos reconhecidos pela base indexada, retornando resultados mais relevantes.

Essa abordagem ainda pode permitir a interação entre diferentes laudos, possibilitando ao usuário médico consultar no banco de dados por um determinado conceito ou descritor de uma das ontologias, um resultado pode ser os laudos que foram classificados por esse conceito. Isso permite uma comparação entre as redações dos laudos, ou ainda uma consulta prévia por diferentes diagnósticos antes do usuário médico redigir o seu parecer em um laudo.

Além disso, os métodos tradicionais de busca em ontologias médicas nem sempre permitem obter resultados isentos de conteúdos não pertinentes. Surge então a necessidade de desenvolvimento de novas metodologias utilizando as diversas tecnologias disponíveis, a fim de organizar, pesquisar, qualificar e cruzar as informações contidas nos mais variados acervos (CIMINO; MIN; PERL, 2003).

4 Revisão Bibliográfica

4.1 Web Semântica

Um dos problemas atuais na organização de alguns nichos informação na Internet é que apesar dos dados serem facilmente compreensivos por seres humanos, os computadores encontram dificuldades em compreendê-los devido ao seu formato. Após um certo tempo pesquisando na Internet o ser humano se adapta para compreender a informação exibida no navegador construindo em si uma rede semântica que interliga significado a palavra através da construção hierárquica de classes e objetos. O fato do computador não notar o significado da informação compromete a qualidade da busca pela informação, já com informações devidamente organizadas e respeitando padrões se torna viável o desenvolvimento de novos sistemas e mecanismos de busca mais eficientes.

Tim Berners-Lee criador da linguagem HTML e líder na criação do consórcio mundial W3C (Word Wide Web Consortium) no Massachusetts Institute of Technology (MIT), foi o idealizador da Web Semântica, e a descreve como: "A Web Semântica é uma extensão da web atual, onde a informação possui um significado claro e bem definido, possibilitando uma melhor interação entre computadores e pessoas" (BERNERS-LEE; HENDLER; LASSILA, 2001).

4.1.1 Arquitetura de Camadas

O W3C propoê uma arquitetura em camadas, que indica os passos a serem tomados para concretizar a implementação da Web Semântica. Uma camada é construída sobre uma inferior e não depende da camada superior, cada camada tende a ser mais especializada e complexa do que as que estão dispostas em níveis inferiores. Existe também a possibilidade de desenvolve-las separadamente e integrá-las posteriormente. A figura 4.1, apresenta as camadas da aquitetura de um modelo proposto para *web semântica*.

No nível mais baixo, composto por **Unicode e URI**, temos respectivamente um padrão internacional de caracteres (Unicode) e um modo de identificar unicamente um recurso URI

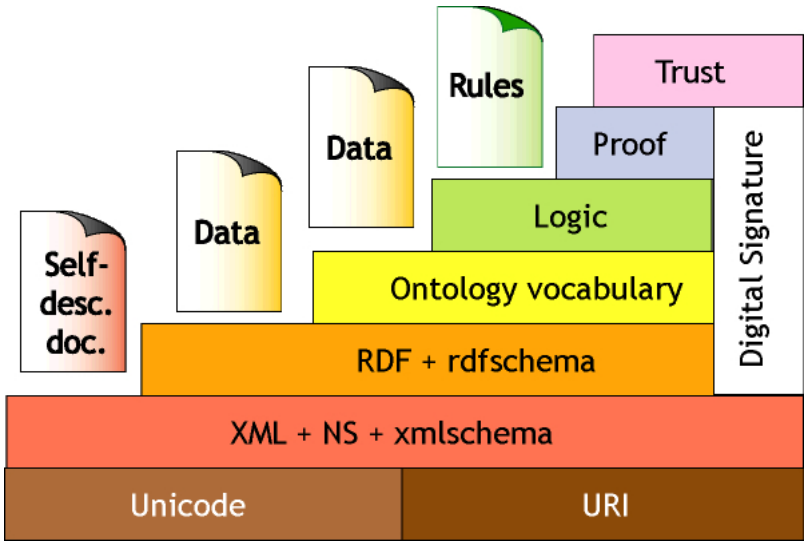

Figura 4.1: arquitetura de camadas da Web Semântica

(Uniform Resource Identifier).

Na camada XML, composta por **Namespaces (NS)** e **xmlschema**, temos estruturação dos dados com um vocabulário definido pelo usuário. Na camada **RDF + rdfschema**, é criada uma estrutura de metadados, que são triplas (sujeito, predicado, objeto). Neste vocabulário torna-se possível associar uma propriedade a um recurso através da sua URI.

Este vocabulário é enriquecido e pode ser expandido pela camada **Ontology vocabulary**, que estende o repertorio de conceitos e relações semânticas envolvidas.

Nas camadas **Logic**, **Proof** e **Trust** estão em desenvolvimento ainda. A camada lógica expressa as regras de tratamento de informações definidas nos níveis inferiores, permitindo ao agente inferir sobre as estruturas de dados. A camada Proof verifica a consistência da informação acessível na Web Semântica, utilizando uma lógica previamente definida em níveis inferiores, aplicando essas regras para gerar novos conhecimentos. O grau de confiança da inferência é definido na camada Trust. Já a camada **Digital Signature** é introduzida nas outras camadas para garantir segurança da informação via criptografia e assinatura digital (RAMALHO; VIDOTTI; FUJITA, 2006).

4.2 Ontologias

Um dos conceitos mais importantes ligados a Web Semântica é o de "ontologia". Basicamente uma ontologia é uma especificação de um conceito, isto é, uma descrição de conceitos e relações que existem em um determinado domínio, ou como Gruber e Guarino a definem: "uma

especificação explícita de uma conceitualização" (GRUBER et al., 1993). Estruturalmente, uma ontologia consiste desses conceitos e relações, suas definições, propriedades e restrições, descritas na forma de axiomas.

No contexto histórico a palavra ontologia tem origem no idioma grego, sendo composta por *ontos (ser)* e *logos (palavra)* (WELTY; GUARINO, 2001).

Na Ciência da Computação, as ontologias são utilizadas com o intuito de classificar um conjunto de informações, descrevendo suas naturezas e co-relacionando entre essas informações, com conceitos e termos formados por um vocabulário específico que descreve um certo domínio.

No ambiente da Web Semântica, a utilização de ontologias favorece o compartilhamento da mesma estrutura de informações entre pessoas e softwares agentes, permitindo inclusive o reuso do conhecimento do domínio, pois torna possível associar uma ontologia a uma página Web, definindo o significado de cada uma das informações existentes e possibilitando a integração e reutilização de ontologias entre diversos domínios. Isso cria a possibilidade, de uma página ser relacionada automaticamente com outras, através de regras de inferência, o que capacita o sistema Web a criar e gerir uma extensa rede de conhecimento, e possivelmente via algoritmos semânticos inferir novos conhecimentos (RAMALHO; VIDOTTI; FUJITA, 2006).

Inúmeros são os benefícios apresentados na literatura para o uso de ontologias, entre eles:

- compartilhamento - permite compreensão comum sobre um domínio de conhecimento.
- reuso - o uso de definições explícitas e formais facilita a manutenção do conhecimento, permitindo o fácil entendimento por parte dos usuários e facilitando a reutilização da ontologia, ou de parte dela.
- estruturação da informação - permite a captura da semântica dos dados e seu processamento automático gerando conhecimento para os humanos.
- interoperabilidade - permite que diferentes sistemas computacionais possam compartilhar dados e informações.
- confiabilidade - uma representação formal torna possível uma automatização consistente e mais confiável.

4.2.1 Tecnologias Agregadas as Ontologias

Diversas linguagens foram desenvolvidas para representação de ontologias. Alguns exemplos são XOL, SHOE, DAML, RDF/RDF(S), OIL e OWL. Tecnologias como *Banco de Dados*,

Data Warehouse e *Mineração de Dados (Data Mining)* têm auxiliado, com o intuito de facilitar a geração de conhecimento que possa ser utilizado para que administradores de sistemas tomem decisões a respeito de problemas com segurança da informação.

- banco de dados - são conjuntos de dados com uma estrutura regular que organizam dados. Um banco de dados normalmente agrupa informações utilizadas para um mesmo fim. Um banco de dados é usualmente mantido e acessado por meio de um software conhecido como Sistema Gerenciador de Banco de Dados (SGBD). Normalmente um SGBD adota um modelo de dados, de forma pura, reduzida ou extendida. Muitas vezes o termo banco de dados é usado como sinônimo de SGDB. O modelo de dados mais adotado hoje em dia ó o modelo relacional, onde as estruturas têm a forma de tabelas, compostas por linhas e colunas.

- data warehouse - ou depósito de dados é um sistema de computação utilizado para armazenar informações relativas às atividades de uma organização em bancos de dados, de forma consolidada. O desenho da base de dados favorece os relatórios, a análise de grandes volumes de dados e a obtenção de informações estratégicas que podem facilitar a tomada de decisão. O data warehouse possibilita a análise de grandes volumes de dados, coletados dos sistemas transacionais (OLTP). São as chamadas séries históricas que possibilitam uma melhor análise de eventos passados, oferecendo suporte às tomadas de decisões presentes e a previsão de eventos futuros. Por definição, os dados em um data warehouse não são voláteis, ou seja, eles não mudam, salvo quando é necessário fazer correções de dados previamente carregados.

- mineração de dados - ou Data Mining é um conjunto de técnicas que buscam a aquisição de novos conhecimentos através da análise de grandes bases de dados. Utilizam diversos algoritmos computacionais tais como Segmentação, Classificação e Previsão. As ferramentas de Data Mining analisam os dados, descobrem problemas ou oportunidades escondidas nos relacionamentos dos dados, e então diagnosticam o comportamento dos negócios, requerendo a mínima intervenção do usuário, assim ele se dedicará somente a ir em busca do conhecimento e produzir mais vantagens competitivas.

4.3 SNOMED CT

Formada em 1993, a SNOMED International em especial o SNOMED CT ou *Systematized Nomenclature of Medicine Clinical Terms* é uma completa nomenclatura multiaxial, desenvol-

vida com o intuito de indexar um conjunto de informações clínicas e atualmente disponível em três idiomas (inglês, alemão e espanhol) (SNOMED CT, 2006).

O SNOMED CT foi concebido, através da união, expansão e reestruturação do SNOMED RT *Reference Terminology* com o NHS *UK National Health Service - Clinical Terms Version 3* (NHS, 2006), também conhecido como *Read Codes*. Esta união, agrega as bem definidas terminologias e procedimentos do SNOMED RT e as terminologias de práticas gerais do CTV3 *Clinical Terms Version 3*, com isso formando um dos mais compreensivos vocabulários clínicos disponíveis atualmente totalizando aproximadamente 984.000 termos no ano de 2005, organizados segundo tipos semânticos e hierárquicos.

São 19 os eixos hierárquicos do SNOMED CT, com várias subclassificações conforme exibido na figura 4.2. Um conceito é classificado de acordo com a classe semântica a que pertence. Esta classificação, está dividida em conceitos, descrições, relacionamentos, hierarquias e subhierarquias que resultam em aproximadamente 1,4 milhões de relacionamentos entre conceitos. Os conceitos do SNOMED CT especificam sinais, sintomas, diagnósticos e procedimentos.

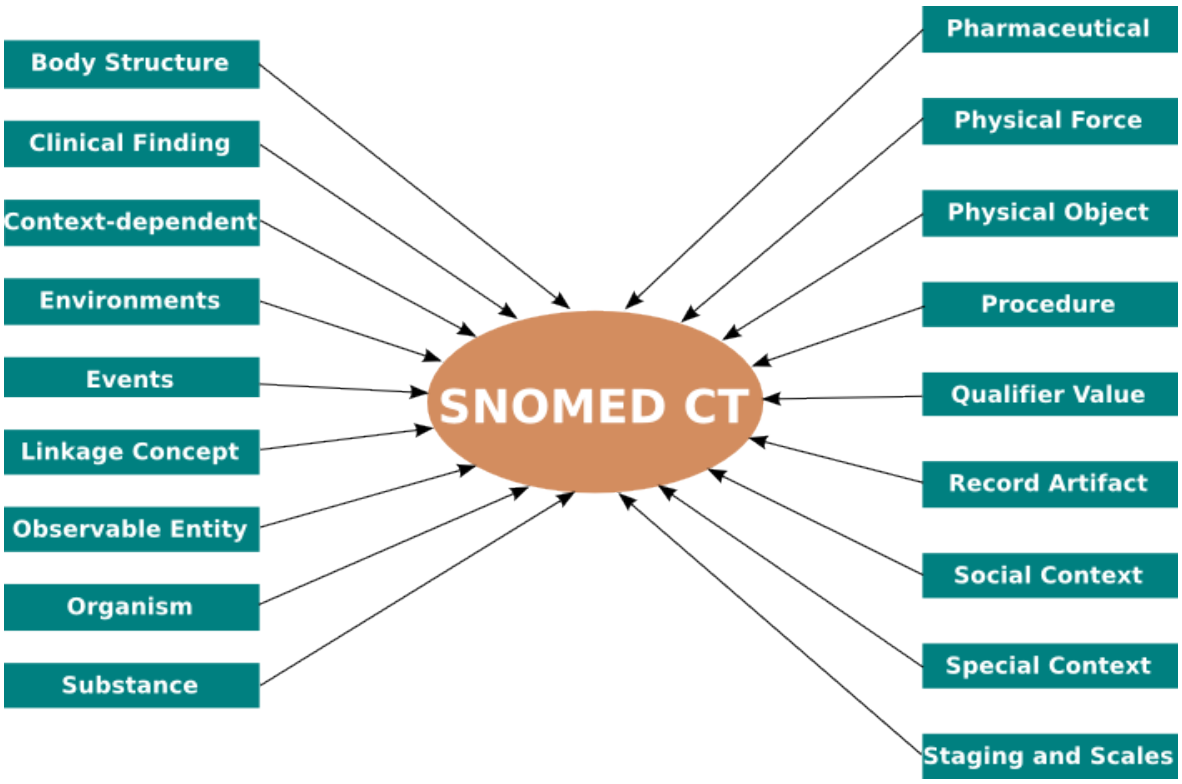

Figura 4.2: eixos da arquitetura do SNOMED CT (versão 2006)

4.4 MeSH

O MeSH ou *Medical Subject Headings* (MESH, 2008) é um outro indexador comumente utilizado em ambiente clínico que teve o seu início em 1963. É controlado pela *U.S. National Library of Medicine* (NLM, 2008), e consiste em uma estrutura hierárquica de termos de diversos níveis de especialidade, totalizando 24.767 descritores na versão 2008. Além destes descritores, existem mais de 172.000 tópicos denominados *Supplementary Concept Records* em um vocabulário separado, também existem 97.000 termos para auxiliar a busca pelo descritor apropriado.

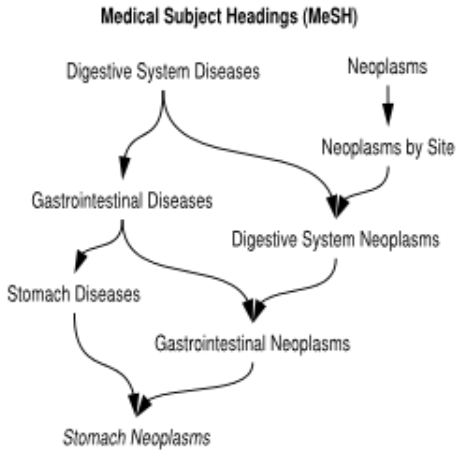

Figura 4.3: exemplo de hierarquia de relacionamentos MeSH

Tem como eixos principais:

- A - Termos Anatômicos.
- B - Organismos.
- C - Doenças.
- D - Medicamentos.
- E - Técnicas e equipamentos Analíticos, Diagnósticos ou Terapêuticos.
- F - Psiquiatria e Psicologia.
- G - Ciências Biológicas.
- H - Ciências Físicas.

- I - Antropologia, Educação, Sociologia, e Fenômenos Sociais.
- J - Tecnologia, Indústria, Agricultura e Alimentos.
- K - Humanidades.
- L - Ciência da Informação e Comunicação.
- M - Grupos de Pessoas.
- N - Saúde.
- V - Características de Publicações.
- Z - Geografia e dados Geográficos.

4.5 DeCS

Concebido pela Biblioteca Virtual em Saúde (BIREME) (BIREME, 2008) em 1986, Descritores em Ciências da Saúde (DeCS) foi criado a partir do MeSH com o objetivo de permitir o uso de terminologia comum para pesquisa em três idiomas (português, inglês e espanhol), proporcionando um meio consistente e único para a recuperação da informação independentemente do idioma.

O DeCS é um vocabulário dinâmico, que está organizado em uma estrutura hierárquica permitindo uma pesquisa mais ampla ou específica ou até, uma busca por todos os termos que pertençam a uma mesma estrutura hierárquica, sua estrutura é baseada em uma divisão do conhecimento em classes e subclasses, que leva em consideração as ligações conceituais e semânticas. Possui aproximadamente 29.490 descritores, sendo destes 24.767 do MeSH, 218 de Ciência e Saúde, 1.950 de Homeopatia, 3.487 de Saúde Pública e 830 de Vigilância Sanitária, sendo que um descritor pode ocorrer mais de uma vez na hierarquia. Por ser dinâmico, registra o processo constante de crescimento e mutação, segundo (DECS, 2008) são registradas a cada ano no mínimo 1000 interações em sua base de dados.

Além dos termos médicos contidos no MeSH, o DeCS adiciona 4 novas áreas específicas de Saúde Pública, Homeopatia, Ciência e Saúde e Vigilância Sanitária.

Anualmente os descritores do MeSH contidos no DeCS são atualizados, o que acarreta mudanças nas demais categorias do DeCS, exigindo uma revisão e atualização do sistema de pesquisa, em especial da estrutura e códigos hierárquicos.

A BIREME também participa do projeto de desenvolvimento de uma terminologia única em saúde conhecida como *Unified Medical Language System* (UMLS) da *U.S. National Library of Medicine* tendo a responsabilidade de enviar atualizações e novos termos nos idiomas português e espanhol.

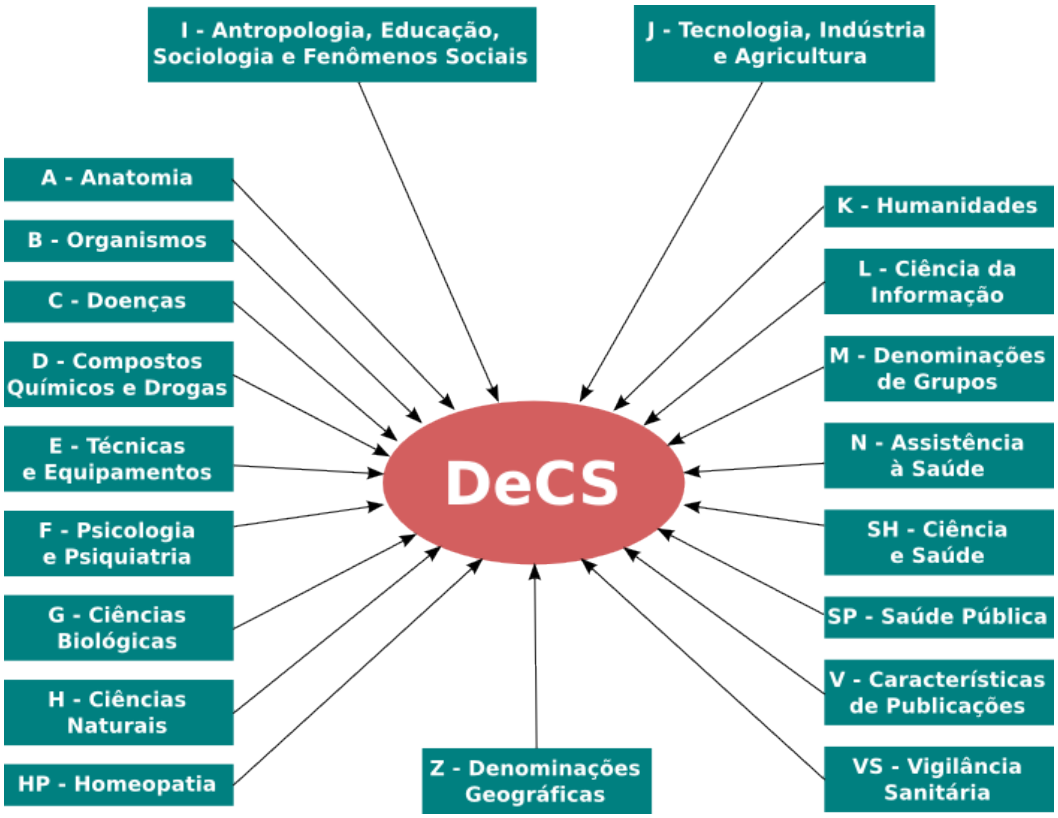

Figura 4.4: eixos da arquitetura do DeCS (versão 2008)

4.6 Ispell

O Ispell é um *spelling checker* para sistemas Unix, que possui suporte para a maioria dos idiomas ocidentais, entre eles português brasileiro, inglês e espanhol. Ele também pode ser interpretado como um dicinário de lexemas, ou seja, um dicionário que contém os lexemas necessários para a formação da maioria das palavras de um determinado idioma.

A história do Ispell é longa, teve o sua primeira versão escrita em 1971 na linguagem Assembly por R.E.Gorin, posteriormente foi portado para a linguagem *C* o popularizando, o que possibilitou a contribuição de vários desenvolvedores. Em pouco tempo o Ispell introduziu a descrição generalizada de morfemas **Affix**, que foi imitada e copiada por muitos outros *spelling checkers* como o MySell e Aspell.

Além disso o Ispell foi o pioneiro em disponibilizar uma interface programática, que foi projetada inicialmente para uso no *emacs*, isso possibilitou a outras aplicações adicionar a opção de *spell-checking* a suas próprias interfaces.

4.6.1 Lexemas

Lexema é um conjunto de palavras de mesma classe morfológica que se distribuem de forma complementar e diferem morfologicamente entre si unicamente por sufixos flexivos (GRAMáTICA DESCRITIVA, 2008). Ou ainda:

> "elemento da língua, a forma básica, que fundamenta as possíveis formas do discurso, bem como todos possíveis significados da palavra (...) de que dispõe a competência do falante/ouvinte" (VILELA, 1979).

Os dicionários utilizam intensivamente o conceito de lexema. Por medida de economia, os dicionários não apresentam entradas para todas as palavras do idioma e se limitam a fornecer uma entrada por lexema. Alguns exemplos:

- as palavras menino, menina, meninos e meninas fazem parte de um mesmo lexema.
- as palavras cantar, cantei, cantamos, cantarias e cante compõem um lexema verbal.
- as palavras lindo, linda, lindos, lindas, lindinha e lindíssima integram o mesmo lexema adjetivo.

4.6.2 Funcionamento

Como em outros *spelling checkers*, o funcionamento do Ispell começa pela leitura da sentença de entrada, essa leitura é efetuada palavra por palavra e só é parada quando uma determinada palavra não é encontrada no seu dicionário de lexemas. Então o Ispell efetua uma tentativa de gerar uma lista de possíveis correções, apresentando a palavra incorreta e alguma sugestão que tenha sido encontrada para o usuário, que por sua vez escolhe entre subistituir a palavra pela sugestão da correção, deixar a palavra como está ou adicionar a palavra não encontrada ao dicionário.

4.7 GIN

O *Generalized Inverted Index* (GIN) é um índice invertido, comumente utilizado em motores de busca textual em documentos de mesma natureza e com um número variado de dados. O termo *"inverted"* ou invertido tem origem na teoria por trás da busca textual completa ou *Full Text Search*. Em um índice padrão, do tipo *"direct"* ou direto, são armazenados pares de identificadores de um determinado documento e parte do texto contido nesse documento. Já em um índice do tipo *"inverted"*, são armazenados pares de palavras de um documento e seus identificadores. Logo em um índice direto, existe um único item (entrada) para cada documento, já em um índice invertido existem tantos itens (entradas) quanto palavras no documento. A estrutura básica do GIN é exibida na figura 4.5.

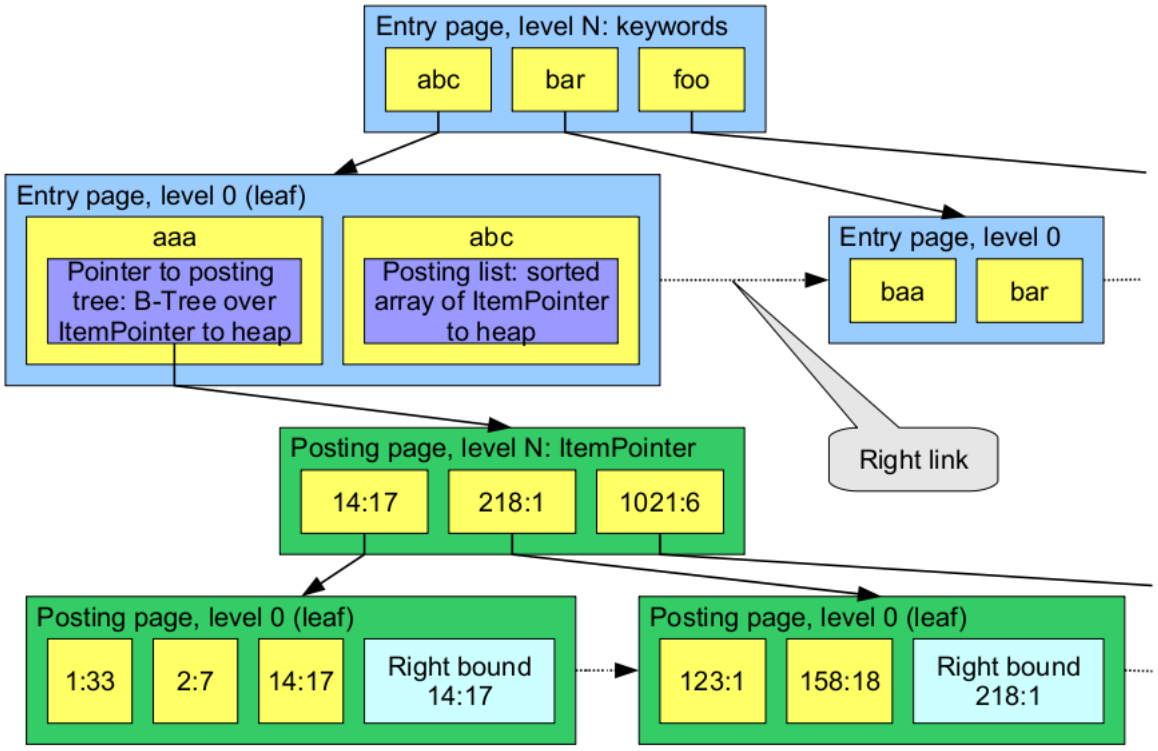

Figura 4.5: estrutura do GIN

Em virtude de melhorias e otimizações, o índice invertido armazena um conjunto composto por uma chave (palavra) e por uma lista de identificadores dos documentos em que a chave está contida normalmente armazenada em ordem, possibilitando uma rápida integração entre os conjuntos quando ocorre uma busca por várias palavras.

O termo *Generalized* ou generalizado, tem significado ligado ao fato de o GIN não saber qual operação acelerar, ele trabalha com estratégias personalizadas, que são definidas de acordo com o tipo de dado armazenado, sendo muito similar ao GiST que é diferenciado pelo uso de

indexação em árvores B ou *B-Trees*, que possue operações de comparação pré-definidas (GIN, 2008).

Algumas características do GIN:

- concorrência/paralelismo - utilizando o algoritmo concorrente de gerência de árvores B de Lehman e Yao.
- WAL - ou *write ahead logging* é um conjunto de técnicas para prover atomicidade e durabilidade para sistemas compostos po banco de dados.
- opclasses - Tsearch2, suporte para array dimensional e a tipo de campo int4[]. Em alguns casos permite ao usuário definir classes de operação para tipos de dados específicos.

Mas nem tudo no GIN são qualidades, ele também possui limitações como:

- não há suporte para índice de multiplas colunas.
- não utiliza chamadas como **scan->kill_prior_tuple** ou **scan->ignore_killed_tuples**.
- efetua busca somente por itens (entradas) totalmente iguais ao termo da busca.
- não permite vasculhar aleatóriamente por índices.
- não indexa valores **NULL**.

Apesar do índice invertido ter sido inventado para busca textual completa, na prática ele pode ser aplicado também para outros propósitos. O índice invertido tem sido muito utilizado em indexação de documentos de uma mesma natureza.

4.8 Cover Density Ranking

Em vez de utilizar algoritmos de calculam a relevância baseados somente na aparência do termo (palavra), como o *Vector Space Model* (VSM, 2002) existem outro algoritmos que obtém maior relevância nos resultados, como o *Cover Density Ranking* ou simplesmente **CDR**, ele avalia a aparência da sentença por completo. O CDR foi desenvolvido para atender melhor as expectativas do usuário quando efetua uma busca textual, um documento que contém a maioria dos termos (palavras) de uma consulta, deve ser classificado com uma pontuação *rank* maior do que outro documento que contém menos termos, independente da frequência em que esses termos ocorrem no documento (CDR, 2002). No CDR, os resultados de uma consulta sobre um documento são classificados seguindo as etapas:

- os documentos que contém um ou mais termos da consulta, são classificados pela posição desses termos no documento. Assim, documentos com um maior número de termos distintos são classificados como mais relevantes. Os documentos são então separados em grupos de acordo com o número de termos distintos que cada um contém.

- posições no documento são classificadas individualmente, e são chamadas *cover set* $\omega = \{(p_1,q_1),(p_2,q_2),\ldots,(p_n,q_n)\}$ e posteriormente são unificas produzindo uma classificação geral.

  Essa classificação é então calculada da seguinte forma:

$$S(\omega) = \sum_{j=1}^{n} I(p_j,q_j) \tag{4.1}$$

  onde:

$$I(p_j,q_j) = \{\frac{\frac{\lambda}{q_j-p_j+1}}{1}\} se(q_j - p_j + 1) > \lambda \tag{4.2}$$

onde (p_j,q_j) é um par ordenado em um documento chamado *cover*. Ele representa o menor intervalo entre 2 termos distintos em um documento, p_j é a posição de um termo e q_j a posição de outro termo, assumindo que q_j é maior que p_j. λ é uma constante referente a tamanho. *Covers* de tamanho λ ou menor, recebem a pontuação 1, já *covers* maiores recebem pontuações menores que 1 em proporção inversa ao seus tamanhos.

Normalmente as pontuações recebidas são normalizadas no intervalo [0,1] para documentos que contém um único termo, e no intervalo [1,2] para documentos com dois ou mais termos. O grande benefício de utilizar o CDR, não está somente no fato de ele considerar o número distinto de termos em um documento, mas também considerar como esses termos estão posicionados nesse documento, por exemplo o quanto perto eles estão uns dos outros.

4.9 PostgreSQL

O PostgreSQL é um gerenciador de banco de dados objeto relacional ou SGBDOR, foi desenvolvido com base nos códigos do projeto Postgres (POSTGRES, 1994), encabeçado e mantido pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley, desde 1985.

Segundo o *The PostgreSQL Global Development Group*, ”O PostgreSQL descende deste código original de Berkeley, possuindo o código fonte aberto. Fornece suporte às linguagens SQL92/SQL99, além de outras funcionalidades modernas. O POSTGRES foi pioneiro em muitos conceitos objeto-relacionais que agora estão se tornando disponíveis em alguns bancos de dados comerciais. Os Sistemas de Gerenciamento de Bancos de Dados Relacionais (SGBDR) tradicionais suportam um modelo de dados composto por uma coleção de relações com nome, contendo atributos de um tipo específico. Nos sistemas comerciais em uso, os tipos possíveis incluem número de ponto flutuante, inteiro, cadeia de caracteres, monetário e data. É amplamente reconhecido que este modelo não é adequado para aplicações futuras de processamento de dados. O modelo relacional substituiu com sucesso os modelos anteriores em parte devido à sua ”simplicidade Espartana”. Entretanto, esta simplicidade tornou a implementação de certas aplicações muito difícil“ (POSTGRESQL, 2008).

4.9.1 Tsearch2 - full text extension for PostgreSQL

Tsearch2 é uma motor de busca textual completa ou *Full Text Search*, totalmente integrado ao core do PostgresSQL desde sua versão 8.3, em versões anteriores era uma extensão. Busca textual completa provê a capacidade de identificar a linguagem natural contida em documentos que satisfaça a consulta, e opcionalmente ordenar os resultados pela sua relevância. O tipo mais comum de busca é a por todos os documentos que contém os termos de uma consulta, onde os resultados são apresentados por ordem de similaridade com a consulta. Os conceitos de consulta (*query*) e similaridade, variam muito de acordo com o tipo de aplicação em que são utilizados, uma busca simples considera uma consulta como um conjunto de palavras e similaridade como a frequência em que essa consulta ocorre no documento.

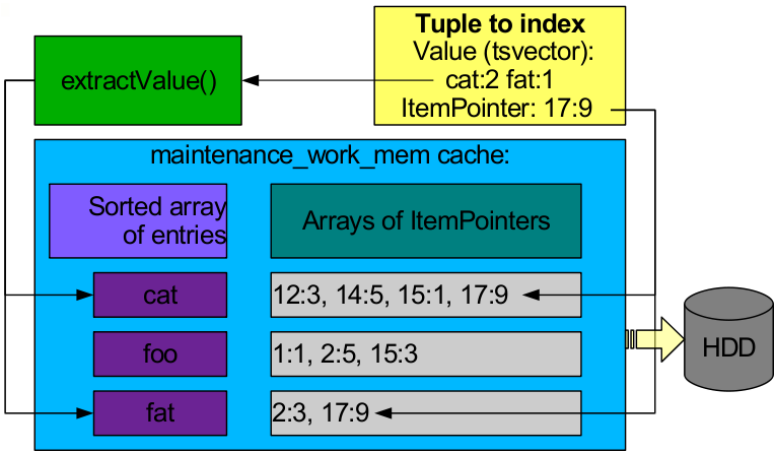

Figura 4.6: fluxo de índices do GIN

O Tsearch2 também é composto por alguns componentes personalizados comuns de bancos de dados, como tabelas, tipos e operadores, ele utiliza campos do tipo **tsvector** para armazenar documentos pré-processados, e o tipo **ts_query** para reprensentar as consultas. Um **tsvector** é composto por lexemas e suas respectivas posições no documento. Nessa aplicação, o Tsearch2 utiliza uma combinação do dicionário Ispell, do GIN e do algoritmo CDR, resultando no fluxo de índice do GIN representado na figura 4.6.

5 *Metodologia*

Atualmente existem mais de cem ontologias médicas que representam informações sobre diversos conceitos (CIMINO; MIN; PERL, 2003). Representar e recuperar estas informações não pode ser considerada uma tarefa simples. Os dados precisam ser re-estruturados e padronizados. Isto implica em considerável esforço computacional para padronização dos dados. Mesmo ao utilizar um indexador padrão (B-tree, por exemplo) para um campo texto, esse índice só teria efeito para consultas que considerassem o início do texto (ex: campo ILIKE palavra%), mas geraria uma árvore de pesquisa grande, aumentando o tamanho da sua base de dados, tornando-a mais lenta.

Apesar dos operadores textuais estarem disponíveis em banco de dados há anos, eles não são completos, faltando muitas propriedades essenciais para aplicações modernas, como:

- não há suporte a idiomas, e expressões regulares não são suficientes, já que elas não conseguem lidar facilmente com palavras derivadas.
- não há suporte a *ranking* ou ordenação, o que torna a busca ineficiente quando inúmeros resultados são encontrados.
- as buscas tendem a ser lentas, já que não há suporte a indexação. Assim todos os documentos devem ser processados a cada busca.

Uma solução para a indexação de textos extensos, é utilizar uma aplicação capaz de converter os termos contidos nos documentos em lexemas indexados pela sua posição em um documento e diminuir o número de ocorrências de um elemento distinto. Dessa forma, a indexação é feita individualmente (palavra por palavra), a partir de um vetor, em um número menor de itens. É nesse ponto que se encontra o escopo desse trabalho: melhorar e criar novas possibilidades de consultas textuais completas e obter resultados com maior conteúdo semântico; efetuar buscas que obtenham resultados mais relevantes do que os de busca por palavras, que é o método mais utilizado hoje, possibilitando a extração do conhecimento em informações clínicas. Para isso, será utilizado um conjunto de técnicas e ferramentas combinadas.

5.1 Ambiente de execução

O sistema operacional escolhido foi o Linux, devido ao fato de que muitas das ferramentas e técnicas detalhadas no capítulo 4 operam especificamente ou tem o seu desempenho melhorado em ambientes Unix. Também foi utilizado o servidor *web Apache* na sua versão 2.2, PostgreSQL versão 8.3 e a linguagem PHP versão 5.

A escolha pela linguagem PHP se deu pela larga gama de recursos atualmente disponíveis para a manipulação de dados, pela sua flexibilidade em lidar com diferentes bancos de dados, e por a versão 5 já disponibilizar um excelente suporte a programação orientada a objetos.

Já a escolha pelo banco de dados PostgreSQL se deve ao excelente suporte a inter-relacionamento entre tabelas, suporte a busca textual completa com indexação prévia através do Tsearch2, além da perfeita integração com a linguagem PHP.

5.2 Extraindo Informações Clínicas das Ontologias

Um primeiro passo se deu em analisar e extrair as informações clínicas do Snomed CT e DeCS. A versão do Snomed CT utilizada nesse trabalho é datada de 31 de janeiro de 2006, foi fornecida em arquivos texto (txt) com definições e conceitos em arquivos separados e no idioma Inglês. Os dados contidos nos arquivos do SNOMED estão listados cada um em uma linha, e seus detalhes são separados em cada coluna como na figura 5.1.

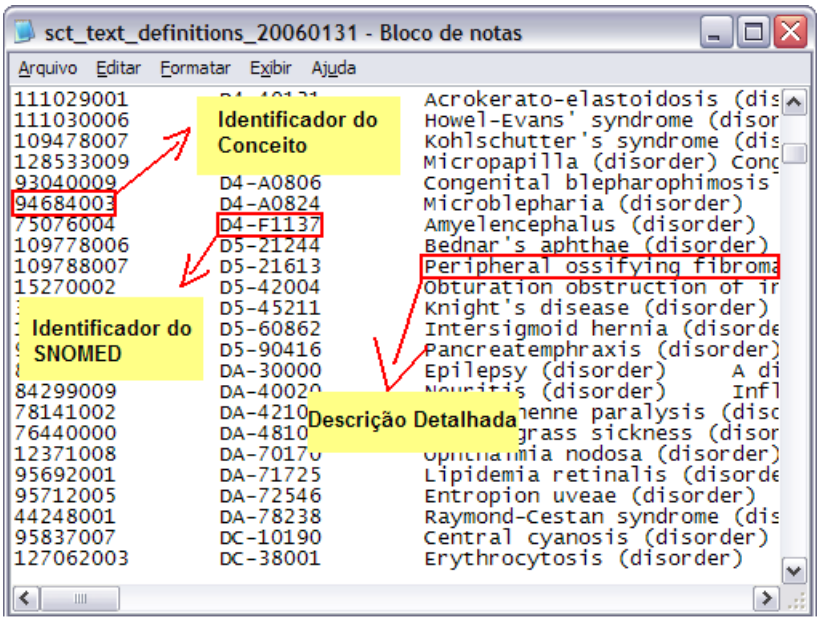

Figura 5.1: arquivo texto de definições do Snomed

A figura 5.1 exibe claramente três colunas, que nesse exemplo de arquivo representam:

- concept id - é o identificador do conceito;
- snomed id - é o código principal de identificação do SNOMED CT;
- description - descrição detalhada do que está sendo referenciado pelo Snomed Id.

O **Concept Id** é um exemplo do conceito de inter-relacionamentos proposto pelo design dessa ontologia, esse identificador também é referenciado em outros arquivos, o que demonstra a conectividade entre os *concepts* (conceitos), tornando esse identificador uma variável essencial na pesquisa realizada pelo sistema.

O DeCS utilizado nesse trabalho, foi obtido através da BIREME em arquivos no formato XML como mostra a figura 5.2. Cada arquivo XML obtido representa um único descritor do DeCS, nesse arquivo estão incluídas informações como: definições, descritores, sinônimos, descritores descendentes e antecessores nos idiomas inglês, português e espanhol. Como em todo arquivo XML, essas informações estão dispostas de forma hierárquica.

```
<decsvmx version="1.0" date="20081012 224933">
- <decsws_response service="" tree_id="A15.145.693">
  - <tree>
    - <self>
      - <term_list lang="pt">
          <term tree_id="A15.145.693">Plasma</term>
        </term_list>
      </self>
    - <ancestors>
      - <term_list lang="pt">
          <term tree_id="A">ANATOMIA</term>
          <term tree_id="A12">Líquidos e Secreções</term>
          <term tree_id="A12.207">Líquidos Corporais</term>
          <term tree_id="A12.207.152">Sangue</term>
          <term tree_id="A">ANATOMIA</term>
          <term tree_id="A12">Líquidos e Secreções</term>
          <term tree_id="A12.207">Líquidos Corporais</term>
          <term tree_id="A12.207.270">Líquido Extracelular</term>
          <term tree_id="A">ANATOMIA</term>
          <term tree_id="A15">Sistemas Sanguíneo e Imune</term>
          <term tree_id="A15.145">Sangue</term>
        </term_list>
      </ancestors>
```

Figura 5.2: arquivo XML com as informações de um descritor do DeCS

As informações contidas nesses arquivos, foram extraídas e inseridas em um banco de dados PostgreSQL o qual será descrito na seção 5.4. Para isso foi implementado um algoritmo *parser*, capaz de ler linha por linha de cada arquivo e extrair as informações.

5.3 Aprimoramento dos Dicionários

O Ispell demostrou ser uma excelente escolha como dicionário de lexemas, possui suporte para a maioria dos idiomas ocidentais. Além disso é muito customizável, o que permitiu o seu aprimoramento através do acréscimo de *stop words* que são artigos, proposições, pronomes, palavras curtas e comuns que ao serem considerados em buscas textuais, "poluem" os resultados. Para aprimorar o algoritmo de busca da aplicação, incluímos as seguintes *stop words* para serem ignoradas durante o processo de indexação:

- *stop words* português - de, a, o, que, e, do, da, em, um, para, é, com, não, uma, os, no, se, na, por, mais, as, dos, como, mas, foi, ao, ele, das, tem, à, seu, sua, ou, ser, quando, muito, há, nos, já, está, eu, também, só, pelo, pela, até, isso, ela, entre, era, depois, sem, mesmo, aos, ter, seus, quem, nas, me, esse, eles, estão, você, tinha, foram, essa, num, nem, suas, meu, às, minha, têm, numa, pelos, elas, havia, seja, qual, será, nós, tenho, lhe, deles, essas, esses, pelas, este, fosse, dele, tu, te, vocês, vos, lhes, meus, minhas, teu, tua, teus, tuas, nosso, nossa, nossos, nossas, dela, delas, esta, estes, estas, aquele, aquela, aqueles, aquelas, isto, aquilo, estou, está, estamos, estão, estive, esteve, estivemos, estiveram, estava, estávamos, estavam, estivera, estivéramos, esteja, estejamos, estejam, estivesse, estivéssemos, estivessem, estiver, estivermos, estiverem, hei, há, havemos, hão, houve, houvemos, houveram, houvera, houvéramos, haja, hajamos, hajam, houvesse, houvéssemos, houvessem, houver, houvermos, houverem, houverei, houverá, houveremos, houverão, houveria, houveríamos, houveriam, sou, somos, são, era, éramos, eram, fui, foi, fomos, foram, fora, fôramos, seja, sejamos, sejam, fosse, fôssemos, fossem, for, formos, forem, serei, será, seremos, serão, seria, seríamos, seriam, tenho, tem, temos, tém, tinha, tínhamos, tinham, tive, teve, tivemos, tiveram, tivera, tivéramos, tenha, tenhamos, tenham, tivesse, tivéssemos, tivessem, tiver, tivermos, tiverem, terei, terá, teremos, terão, teria, teríamos, teriam.

- *stop words* inglês - i, me, my, myself, we, us, our, ours, ourselves, you, your, yours, yourself, yourselves, he, him, his, himself, she, her, hers, herself, it, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, will, would, shall, should, can, could, may, might, must, ought, i'm, you're, he's, she's, it's, we're, they're, i've, you've, we've, they've, i'd, you'd, he'd, she'd, we'd, they'd, i'll, you'll, he'll, she'll, we'll, they'll, isn't, aren't, wasn't, weren't, hasn't, haven't, hadn't, doesn't, don't, didn't, won't, wouldn't, shan't, shouldn't, can't, cannot, couldn't, mustn't, let's,

that's, who's, what's, here's, there's, when's, where's, why's, how's, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, very, one, every, least, less, many, now, ever, never, say, says, said, also, get, go, goes, just, made, make, put, see, seen, whether, like, well, back, even, still, way, take, since, another, however, two, three, four, five, first, second, new, old, high, long.

- *stop words* espanhol - de, la, que, el, en, y, a, los, del, se, las, por, un, para, con, no, una, su, al, es, lo, como, más, pero, sus, le, ya, o, fue, este, ha, sí, porque, esta, son, entre, está, cuando, muy, sin, sobre, ser, tiene, también, me, hasta, hay, donde, han, quien, están, estado, desde, todo, nos, durante, estados, todos, uno, les, ni, contra, otros, fueron, ese, eso, había, ante, ellos, e, esto, mí, antes, algunos, qué, unos, yo, otro, otras, otra, él, tanto, esa, estos, mucho, quienes, nada, muchos, cual, sea, poco, ella, estar, haber, estas, estaba, estamos, algunas, algo, nosotros, mi, mis, tú, te, ti, tu, tus, ellas, nosotras, vosotros, vosotras, os, mío, mía, míos, mías, tuyo, tuya, tuyos, tuyas, suyo, suya, suyos, suyas, nuestro, nuestra, nuestros, nuestras, vuestro, vuestra, vuestros, vuestras, esos, esas, estoy, estás, está, estamos, estáis, están, esté, estés, estemos, estéis, estén, estaré, estarás, estará, estaremos, estaréis, estarán, estaría, estarías, estaríamos, estaríais, estarían, estaba, estabas, estábamos, estabais, estaban, estuve, estuviste, estuvo, estuvimos, estuvisteis, estuvieron, estuviera, estuvieras, estuviéramos, estuvierais, estuvieran, estuviese, estuvieses, estuviésemos, estuvieseis, estuviesen, estando, estado, estada, estados, estadas, estad, he, has, ha, hemos, habéis, han, haya, hayas, hayamos, hayáis, hayan, habré, habrás, habrá, habremos, habréis, habrán, habría, habrías, habríamos, habríais, habrían, había, habías, habíamos, habíais, habían, hube, hubiste, hubo, hubimos, hubisteis, hubieron, hubiera, hubieras, hubiéramos, hubierais, hubieran, hubiese, hubieses, hubiésemos, hubieseis, hubiesen, habiendo, habido, habida, habidos, habidas, soy, eres, es, somos, sois, son, sea, seas, seamos, seáis, sean, seré, serás, será, seremos, seréis, serán, sería, serías, seríamos, seríais, serían, era, eras, éramos, erais, eran, fui, fuiste, fue, fuimos, fuisteis, fueron, fuera, fueras, fuéramos, fuerais, fueran, fuese, fueses, fuésemos, fueseis, fuesen, siendo, sido, tengo, tienes, tiene, tenemos, tenéis, tienen, tenga, tengas, tengamos, tengáis, tengan, tendré, tendrás, tendrá, tendremos, tendréis, tendrán, tendría, tendrías, tendríamos, tendríais, tendrían, tenía, tenías, teníamos, teníais, tenían, tuve, tuviste, tuvo, tuvimos, tuvisteis, tuvieron, tuviera, tuvieras, tuviéramos, tuvierais, tuvieran, tuviese, tuvieses, tuviésemos, tuvieseis, tuviesen, teniendo, tenido, tenida, tenidos, tenidas, tened.

5.4 Banco de Dados Integrado

O mapeamento e a integração de ontologias pertencentes a domínios semelhantes, como também a utilização de um método de busca semântica são fundamentais quando queremos agregar mais conhecimento em laudos médicos. Assim, a modelagem do banco de dados se deu logo após a análise dos campos e das informações contidas nos arquivos descritos na seção 5.2. Como resultado foi obtido o banco de dados representado pela figura 5.3.

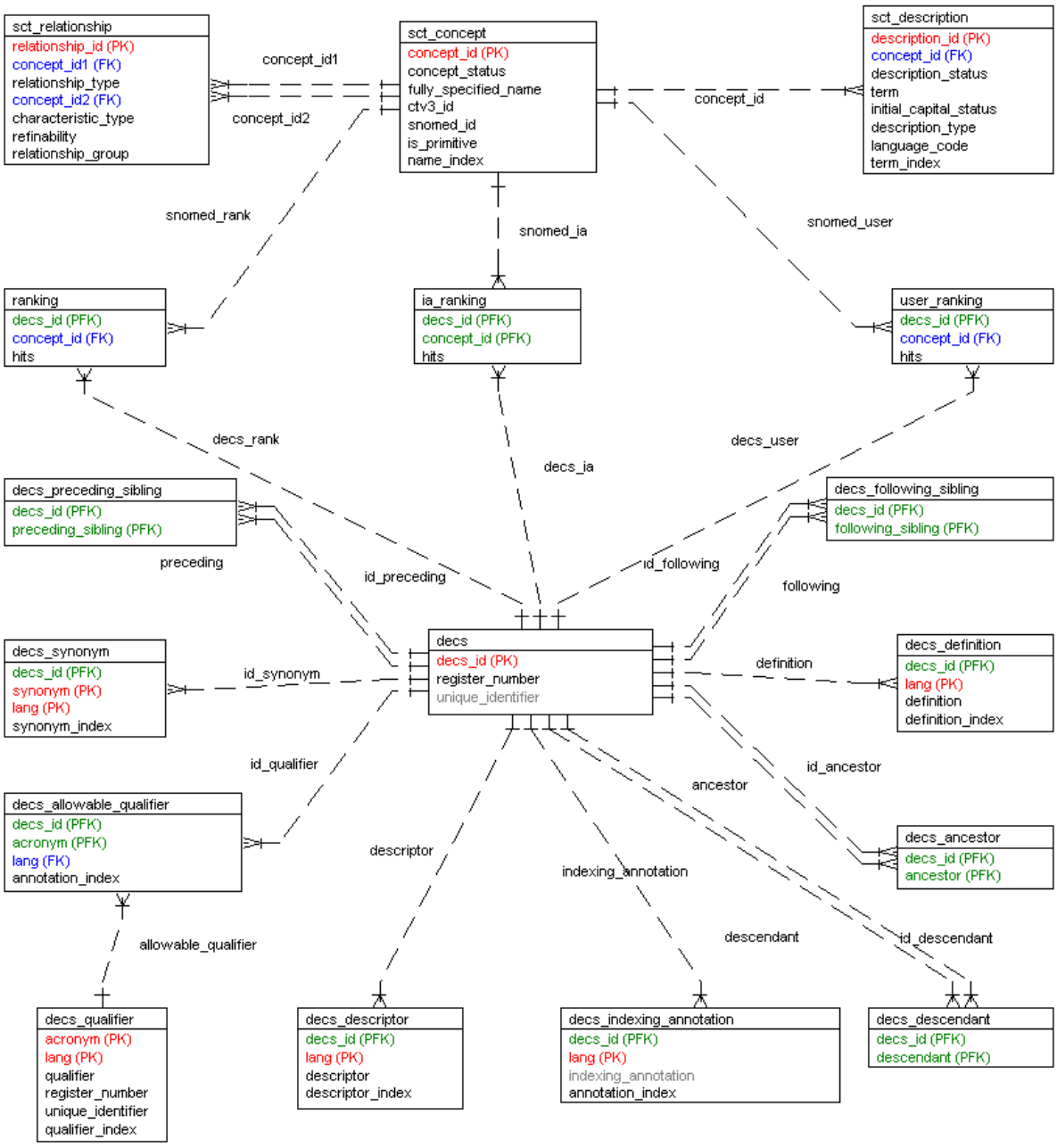

Figura 5.3: diagrama ER do banco de dados resultante

O banco de dados resultante contempla tanto os conceitos do SNOMED CT como os do

DeCS. Isso inclui as tabelas **sct_ concept** (possui o identificador, o seu estado, o nome, o identificador *CTV3 - Clinical Terms Version 3* - e o identificador SNOMED CT), **sct_ description** (onde a descrição do conceito em um determinado idioma é armazenada) e **sct_ relationship** (onde dois conceitos são vinculados, guardando as relações e hierarquia entre estes conceitos) pertencentes ao SNOMED CT.

Já as tabelas **decs** (onde estão o identificador do conceito e o número de registro), **decs_ preceding_ sibling** (conceitos no mesmo nível de hierarquia que estão classificados antes do conceito avaliado), **decs_ following_ sibling** (conceitos no mesmo nível de hierarquia que estão classificados depois do conceito avaliado), **decs_ synonym** (sinônimos de um conceito em um determinado idioma são armazenados), **decs_ definition** (definições de um conceito em cada um dos idiomas), **decs_ ancestor** (conceitos que antecedem o conceito avaliado), **decs_ descendant** (conceitos que descendem do conceito avaliado), **decs_ descriptor** (descritores de um conceito em cada um dos idiomas), **decs_ allowable_ qualifier** (qualificadores do conceito) e **decs_ indexing_ annotation** (indexadores do conceito) pertencentes ao DeCS. Por último a tabela **ranking** (onde os co-relacionamentos entre um conceito SNOMED CT e um conceito DeCS são armazenados e pontuados).

Também foi necessário criar campos do tipo *tsvector*, que são preenchidos com o valor do documento indexado pelo GIN, essa etapa é detalhada na seção 5.5. Entre esses campos temos: **name_ index** (campo fully_ specified_ name da tabela sct_ concept indexado), **term_ index** (campo term da tabela sct_ definition indexado), **definition_ index** (campo definition da tabela decs_ definition), **descriptor_ index** (campo descriptor da tabela decs_ descriptor), **synonym_ index** (campo synonym da tabela decs_ synonym), **annotation_ index** (campo indexing_ annotation da tabela decs_ indexing_ annotation) e **qualifier_ index** (campo qualifier da tabela decs_ qualifier).

5.5 Indexando as Informações

Um dos temas centrais desse trabalho é a indexação eficiente dos documentos contidos no banco de dados descrito na seção 5.4. Para executar essa tarefa, foi utilizada a combinação dos dicionários português, inglês e espanhol do Ispell e a ferramenta GIN, isso gerou campos do tipo *tsvector* indexados utilizando índice invertido, que armazena um conjunto composto por uma chave (palavra) e por uma lista de identificadores dos documentos em que a chave está contida.

A figura 5.4, exibe tanto o campo com o conteúdo textual original (definition) como a sua

versão indexada pelo GIN (definition_ index) e armazenada em um campo do tipo *tsvector*. Um *tsvector* é um campo que representa um documento (texto) reduzido aos seus lexemas.

definition character varying	definition_index tsvector
A layer of the peritoneum which attaches the abdomina	'lay':2 'wall':14 'blood':18 'layer':2 'nerve':21 'attach':7
Capa del peritoneo que une las vísceras abodminales c	'une':5 'vas':17 'capa':1 'nervi':20 'pared':11 'viscer':7
Camada do peritônio que liga as vísceras abdominais à	'liga':5 'vasos':15 'camada':1 'nervos':18 'parede':10 '
The fold of peritoneum by which the COLON is attachec	'fold':2 'wall':15 'colon':8 'attach':10 'attache':10 'attac
Pliegue del peritoneo mediante el cual el COLON se un	'une':10 'colon':8 'pared':13 'pliegue':1 'mediante':4 'a
Prega do peritônio pela qual o COLO liga-se à PAREDE /	'colo':7 'liga':9 'prega':1 'liga-s':8 'parede':12 'peritôni'
A double-layered fold of peritoneum that attaches the	'fold':5 'layer':4 'organ':14 'attach':9 'cavity':18 'double
Pliegue de doble capa del peritoneo que une el ESTÓM/	'une':8 'capa':4 'doble':3 'organ':13 'cavidad':16 'pliegu
Dobra (camada dupla) do peritônio que une (attach) o	'une':7 'dobra':1 'dupla':3 'attach':8 'camada':2 'outro
The space enclosed by the peritoneum. It is divided int	'lie':24 'sac':15,19,30 'two':11,29 'less':18 'bursa':22 'g
Espacio encerrado por el peritoneo. Se divide en dos p	'dos':9,27 'sac':28 'saco':12 'bolsa':18 'divid':7 'epipl':3
Espaço recoberto pelo peritônio. É dividido em duas pa	'é':5 'dois':27 'duas':8 'saco':12,16 'atrás':23 'bolsa':1
Natural openings in the subdiaphragmatic lymphatic pl	'cell':15 'open':2 'cavity':31 'plexus':7 'system':35 'con
Aberturas naturales del plexo linfático subdiafragmátic	'vía':20 'celul':12 'estom':16 'plexo':4 'drenaj':24 'aber
Aberturas naturais no plexo linfático subdiafragmático	'plex':4 'vias':20 'estom':15 'células':11 'periton':16 'si
An area occupying the most posterior aspect of the AB	'area':2 'brim':31 'true':34 'bound':14 'space':43 'aspe
Area que ocupa el lado más posterior de la CAVIDAD A	'are':1 'bord':17 'lado':5 'ocup':3 'borde':31 'cuadr':21

Figura 5.4: campos com a definição textual original (definition) e sua versão indexada (definition_ index) reduzida aos seus lexemas

O GIN adiciona escalabilidade ao Tsearch2. Seu índice consiste em um par (chave, lista) onde a chave é a palavra (lexema) e a lista é uma lista dos documentos onde a chave ocorre. A lista foi então armazenada em ordem, o que possibilitou uma busca rápida e eficiente quando várias palavras são utilizadas.

5.6 Desenvolvimento

O ponto principal desse trabalho é efetuar busca textual completa com extração de conteúdo semântico. Para isso, foi utilizado um conjunto variado de ferramentas e técnicas já descritas na capítulo 4.

A primeira grande vantagem do uso do Tsearch2 é que ele traz uma funcionalidade que não está presente no *kernel* do PostgreSQL: um poderoso sistema de indexação de documentos. Um etapa fundamental para que a busca textual utilizando o Tsearch2 fosse executada, foi a melhoria dos dicionários descrita na seção 5.3. A inclusão de *stop words*, está diretamente

ligada com a qualidade dos resultados da busca, por serem termos muito comuns em qualquer documento textual, esses termos tendem a pertubar os resultados trazendo respostas com pouca ou nenhuma relevância semântica com a busca inicial.

O uso de dicionários permitiu um controle refinado dos lexemas. Nessa aplicação, está sendo utilizado o dicionário Ispell nos idiomas português, inglês e espanhol. Ao utilizar dicionários apropriados e aprimorados, foi possível:

- definir *stop words* que não serão indexadas.
- mapear símbolos a uma única palavra utilizando o Ispell.
- mapear sentenças a uma única palavra utilizando um *thesaurus*.
- mapear diferentes variações de uma palavra para uma forma canônica.

Ao utilizar o Tsearch2 para efetuar indexação prévia, obtemos uma maior performance nas consultas textuais. Esse pré-processamento inclui as seguintes etapas:

- utilizando uma técnica de *parsing* os documentos são convertidos e armazenados em vários *tokens* ou símbolos. Esses símbolos englobam classes de números, palavras, palavras compostas, e-mail, endereços, assim cada classe é processada diferentemente, em teoria as classes de símbolos variam de acordo com a aplicação, mas na maioria dos casos são utilizadas classes pré-definidas pelo *parser* padrão utilizado no PostgreSQL.
- conversão dos símbolos em lexemas. Nessa etapa os símbolos são convertidos em lexemas normalizados, removendo as *stop words* palavras muito comuns com pouca relevância que são inúteis para a busca. O PostgreSQL utiliza dicionários para mapear esses lexemas.
- armazenamento de documentos indexados previamente para uma melhor perfomance durante as buscas. Cada documento é representado por um array de lexemas normalizados e suas respectivas posições no documento, essas posições são utilizadas para medir a relevância desse documento com relação a consulta efetuada.

Os documentos pré-processados foram então armazenados no banco de dados detalhado na seção 5.4. Com a etapa de integração das informações clínicas das ontologias, tornou-se possível a interação entre si, já que tratam de informações similares. Como exemplo, ao pesquisar por “Doença de Alzheimer”, cada ontologia retorna:

SNOMED CT - Alzheimer's disease (disorder) código 26929004.

MeSH - Alzheimer Disease - código C10.228.140.380.100.

DeCS - Doença de Alzheimer - código C10.228.140.380.100.

O Tsearch2 também provê uma sintaxe própria derivada da linguagem SQL. Esta sintaxe é utilizada para inquirir sobre os descritores indexados. Para enviarmos consultas complexas ao Tsearch2, é necessário fazer uso de seus "operadores especiais", os quais usamos dentro do parâmetro que passamos a ele a exemplo da figura 5.5.

```
SELECT * FROM decs_definition WHERE definition @@ to_tsquery('(tratamento|câncer)
 & tratamento de câncer') ORDER BY rank;
```

Figura 5.5: consulta SQL utilizando Tsearch2

Nessa aplicação, foi escolhido o algoritmo classificador CDR ou *Cover Density Ranking* detalhado no capítulo 4, nele a proximidade, seqüência e número de ocorrências entre as palavras contidas em um documento, marcam mais pontos do que o método comum em buscas por palavras-chave, onde somente o maior número de ocorrências de palavras é levado em consideração.

A última etapa se refere ao algoritmo na linguagem PHP, que trata os dados retornados ao inquirir o banco de dados. Esses dados são então enviados a uma interface *web*, que exibe os resultados da busca em todas as ontologias, permitindo ao usuário avaliador comparar a equivalência dos termos.

Durante a implementação dos métodos de buscas na linguagem PHP, um ponto crítico foi identificado: o SNOMED CT só estava disponível no idioma Inglês. Sabendo que a maioria das buscas será efetuada em laudos médicos no idioma português (já que a aplicação está sendo testada em ambiente de língua portuguesa), foi constatado um problema com a identificação da busca em português em uma ontologia de língua inglesa. A solução para amenizar esta dificuldade veio através do DeCS, já que ele possui em sua maioria os conceitos nos três idiomas. Logo, além de efetuar uma busca semântica de uma sentença em português no DeCS, é necessário efetuar uma segunda busca semântica entre o correspondente no idioma inglês no DeCS e os conceitos em inglês do SNOMED CT, retornando os conceitos do SNOMED CT mais próximos da sentença original em português.

A aplicação final foi desenvolvida no padrão MVC (*model-view-controller*), o que mantém o código bem definido e classificado. Para implementar o modelo foi utilizado *PHP Data Object*

(PDO), uma extensão leve e confiável que oferece uma camada abstrata de acesso a banco de dados.

6 *Resultados*

6.1 Resultados

Como solução para o problema da interoperabilidade entre as ontologias, deu-se início ao desenvolvimento de um banco de dados integrado, onde tanto os conceitos do SNOMED CT quanto os do DeCS estão indexados, como detalhado no capítulo 5. Os campos indexados pelo Tsearch2 são de extrema importância para o funcionamento da metodologia proposta, já que eles guardam o posicionamento exato de cada lexema em uma determinada sentença.

Agora, ao executar uma busca pelo código do termo Alzheimer na ontologia DeCS, resulta em um campo indexado através de sua definição, conforme descrito no exemplo:

- código - C10.228.140.380.100.
- termo - Doença de Alzheimer.
- definição - Doença degenerativa do CÉREBRO caracterizada pelo início traiçoeiro de DEMÊNCIA. Falhas da MEMÓRIA, no julgamento, no momento da atenção e na habilidade em resolver problemas são seguidos de APRAXIAS severas e perda global das habilidades cognitivas. A condição ocorre principalmente após os 60 anos de idade e é marcada patologicamente pela atrofia cortical severa e tríade de PLACAS SENIS, EMARANHADOS NEUROFIBRILARES e FILAMENTOS DO NEURÓPILO.
- campo indexado em português - 'doença':1 'alzheim':3.
- campo indexado em inglês - 'alzheim':1 'disease':2.

Através do algoritmo CDR combinado com a comparação dos campos indexados, a aplicação desenvolvida é capaz de identificar o conceito do SNOMED CT mais relevante ao resultado obtido na busca por um termo DeCS:

- código - 26929004.

- termo - Alzheimer's disease.
- definição - Alzheimer's disease (disorder).
- campo indexado em inglês - 'alzheim':1 'disease':3 'disorder':4.

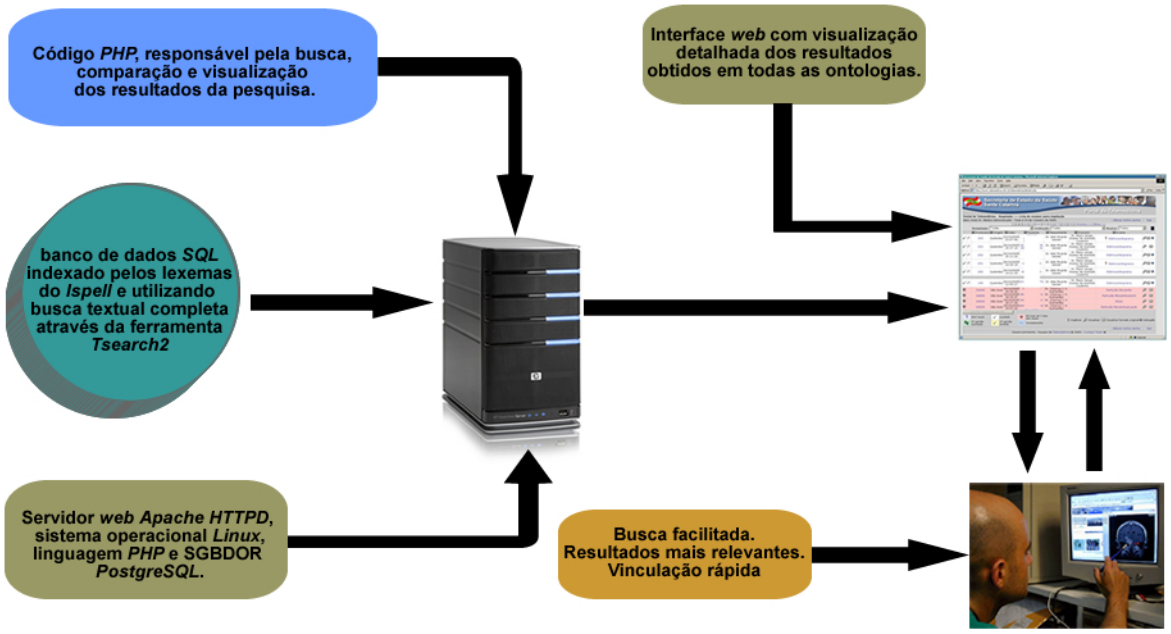

Figura 6.1: diagrama do resultado obtido.

Outro resultado é a interação entre os laudos médicos, que agora é possível já que estes agora possuem identificadores das ontologias. Isso permite ao usuário médico pesquisar por todos os laudos que foram indexados por um determinado conceito SNOMED CT, ou descritor Mesh e DeCS.

A aplicação resultante possui um conjunto variado de funcionalidades, dentre as quais podem ser citadas: criação de sentenças do tipo *tsquery* sobre um campo *tsvector*, submissão de sentenças *tsquery* com variado número de palavras, submissão da mesma sentença em três idiomas (português, inglês e espanhol), composição e exibição hierárquica dos conceitos das ontologias, escolha de tipo de classificação (*ranking*) e exibição dos resultados.

7 Conclusão

7.1 Conclusão

Partindo inicialmente de duas ontologias que não interagiam entre si e que foram recebidas em formatos texto e XML como no SNOMED CT e no DeCS, foi um desafio que resultou em um banco de dados capaz de armazenar todas as suas informações clínicas de ambas as ontologias no formato *tsvector*.

A possibilidade de obter resultados mais relevantes, com menor discrepância e isentos de conteúdos não pertinentes por meio da integração dessas ontologias aumenta consideravelmente a agregação de conhecimento em laudos médicos, resultando em diagnósticos mais completos e precisos.

A universalidade das ontologias abordadas possibilita que médicos e pacientes de diferentes nacionalidades possam interagir e compreender as informações clínicas contidas em laudos redigidos em diferentes idiomas. A metodologia utilizada permite ainda cruzar informações de um laudo previamente indexado pelo MeSH ou DeCS, com o SNOMED CT. Neste caso, ao indexar um laudo na ontologia DeCS pode-se facilmente visualizar o conceito equivalente no SNOMED CT (JUNIOR et al., 2008).

7.2 Trabalhos futuros

A aplicação dessa metodologia pode ser utilizada para auxiliar o preenchimento de laudos estruturados utilizando DICOM SR. Existe uma forte resistência à adoção de um padrão de laudo estruturado por parte da comunidade médica em virtude do grande número de campos a serem preenchidos. Uma abordagem utilizando esta metodologia pode facilitar o preenchimento destes laudos através do algoritmo de preenchimento semântico. Pode-se ainda conceber um modelo para transcrever um laudo não estruturado para o padrão DICOM SR, onde os termos são identificados e vinculados a um determinado tipo de campo.

Referências Bibliográficas

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, v. 5, 2001. Disponível em: <http://www.sciam.com/>. Acesso em: 25 julho 2008.

BIREME. *Biblioteca Virtual em Saúde*. 2008. Disponível em: <http://www.bireme.org/>. Acesso em: 23 julho 2008.

CDR. *Cover Density Ranking (CDR)*. 2002. Disponível em: <http://www2002.org/CDROM/refereed/643/node7.html>. Acesso em: 23 julho 2008.

CIMINO, J. J.; MIN, H.; PERL, Y. Consistency across the hierarchies of the umls semantic network and metathesaurus. *Journal of Biomedical Informatics*, v. 36, n. 6, p. 450–461, Dec 2003.

DECS. *DeCS - Descritores em Ciências da Saúde*. 2008. Disponível em: <http://decs.bvs.br/>. Acesso em: 23 julho 2008.

GIN. *GIN - Generalized Inverted Index*. 2008. Disponível em: <http://www.sigaev.ru/gin/>. Acesso em: 18 julho 2008.

GRAMáTICA DESCRITIVA. 2008. Disponível em: <http://www.radames.manosso.nom.br/gramatica/lexema.htm>. Acesso em: 23 julho 2008.

GRUBER, T. R. et al. Toward principles for the design of ontologies used for knowledge sharing. In: *In Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, in press. Substantial revision of paper presented at the International Workshop on Formal Ontology*. [S.l.: s.n.], 1993.

JUNIOR, C. L. B. et al. Busca semântica aplicada a informações clínicas. *XI Congresso Brasileiro de Informática em Saúde, CBIS'2008*, Nov 2008.

MESH. *MeSH - Medical Subject Headings*. 2008. Disponível em: <http://www.nlm.nih.gov/mesh/>. Acesso em: 22 julho 2008.

NHS. *UK National Health Service (NHS)*. 2006. Disponível em: <http://www.connectingforhealth.nhs.uk/>. Acesso em: 23 julho 2008.

NLM. *U.S. National Library of Medicine*. 2008. Disponível em: <http://www.nlm.nih.gov/>. Acesso em: 22 julho 2008.

POSTGRES. *University POSTGRES*. 1994. Disponível em: <http://db.cs.berkeley.edu/postgres.html>. Acesso em: 25 julho 2008.

POSTGRESQL. *PostgreSQL - Global Development Group*. 2008. Disponível em: <http://www.postgresql.org/>. Acesso em: 25 julho 2008.

RAMALHO, R. A. S.; VIDOTTI, S. A. B. G.; FUJITA, M. S. L. Web semÂntica: Aspectos interdisciplinares para a organizaÇÃo e recuperaÇÃo de informaÇÕes. *ENCONTRO NACIONAL DE PESQUISA EM CIÊNCIA DA INFORMAÇÃO (ENANCIB)*, Jun 2006.

SNOMED CT. *SNOMED CT - Systematized Nomenclature of Medicine Clinical Terms*. 2006. Disponível em: <http://www.ihtsdo.org/snomed-ct/>. Acesso em: 25 julho 2008.

VILELA, M. *Estruturas Léxicas do Português*. [S.l.]: Almedina, 1979.

VSM. *Vector Space Model (VSM)*. 2002. Disponível em: <http://www2002.org/CDROM/refereed/643/node5.html>. Acesso em: 27 julho 2008.

WELTY, C.; GUARINO, N. *Suporting Ontological Analysis of Taxonomic Relationships*. [S.l.: s.n.], 2001. 51-74 p.

Apêndices

Apêndice A - Código Fonte

DecsBase.php

```php
<?php
/**
 * CLBJ's TCC
 *
 * @category    CLBJ
 * @package     application/classes
 * @license     Parte integrante do trabalho de conclusão do curso de Ciências da Computação
 * da Universidade Federal de Santa Catarina, a reprodução, alteração e utilização sem autorização
 * do autor são proibidas.
 * @author      Cloves Langendorf Barcellos Junior
 * @version     0.2 27/02/2008
 */
class DecsBase
{
    protected $sentence;
    protected $language;
    protected $dictionary;
    protected $stopWords;
    protected $db;
    protected $base;
    protected $roots;
    protected $urlBase;
    protected $searchResult;

    //tabelas Decs
    protected $decs;
    protected $decsDecendant;
    protected $decsAncestor;
    protected $decsDescriptor;
    protected $decsDefinition;
```

DecsBase.php

```
33    public $auxiliary;
34
35    public function DecsBase()
36    {
37        $this->db = Zend_Registry::get('db');
38        $this->base = Zend_Registry::get('base');
39        $this->roots = array
  ("A","B","C","D","E","F","G","H","HP","I","J","K","L","M
  ","N","SH","SP","V","VS","Z");
40        $this->searchResult = array();
41
42        $this->decs = new Decs();
43        $this->decsDecendant = new DecsDescendant();
44        $this->decsAncestor = new DecsAncestor();
45        $this->decsDescriptor = new DecsDescriptor();
46        $this->decsDefinition = new DecsDefinition();
47        $this->decsSynonym = new DecsSynonym();
48        $this->decsAnnotation =  new
  DecsIndexingAnnotation();
49
50        $this->auxiliary = new Auxiliary();
51    }
52
53    /**
54     * Método para atribuir o diretório raiz da
  aplicação.
55     *
56     * @param string $url
57     */
58    function setUrlBase($url)
59    {
60        $this->urlBase = $url;
```

Page 2

DecsBase.php

```
65      *
66      * @param string $language pode ser 'pt','es' ou 'en'
67      */
68     function setLanguage($language)
69     {
70         $this->language = $language;
71 
72         //atribui os dicionários ispell
73         switch ($this->language) {
74             case 'pt':
75                 $this->dictionary = Zend_Registry::get
  ('dictPT');
76                 $this->stopWords = Zend_Registry::get
  ('stopWordsPT');
77                 break;
78             case 'en':
79                 $this->dictionary = Zend_Registry::get
  ('dictEN');
80                 $this->stopWords = Zend_Registry::get
  ('stopWordsEN');
81                 break;
82             case 'es':
83                 $this->dictionary = Zend_Registry::get
  ('dictES');
84                 $this->stopWords = Zend_Registry::get
  ('stopWordsES');
85                 break;
86         }
87     }
88 
89     /**
90      * Método responsável por atribuir a frase utilizada
```

DecsBase.php

```
 97         $this->sentence = $string;
 98     }
 99 
100     /**
101      * Método responsável por retornar a frase utilizada
102      * nas pesquisas.
103      *
104      */
105     function getSentence()
106     {
107         return $this->sentence;
108     }
109 
110     /**
111      * Método responsável por retornar o resultado da
    busca.
112      */
113     function getSearchResult()
114     {
115         return $this->searchResult;
116     }
117 
118     /**
119      * Método responsável por filtrar o texto a ser
    pesquisado,
120      * retornando uma versão otimizada para pesquisa com
    o tsearch2.
121      * @param string $sentence frase a ser avaliada
122      * @param string $language pode ser 'pt','es' ou 'en'
123      */
124     function getSqlWords($sentence = null, $language =
    null)
```

Page 4

DecsBase.php

```
129         if ($language != null) {
130             $this->setLanguage($language);
131         }
132 
133         $return = '';
134         $sentence = $this->sentence;
135         for ($i = 0; $i < count($this->stopWords); $i++)
    {
136             $comparador = " ".$this->stopWords[$i]." ";
137             if (substr_count($sentence,$comparador) > 0)
    {
138                 $arrayTemp = explode($comparador,
    $sentence);
139                 for ($j = 0; $j < count($arrayTemp); $j+
    +) {
140                     if ($j == 0) {
141                         $return = trim($arrayTemp[$j]);
142                     } else {
143                         $return .= ' '.trim($arrayTemp
    [$j]);
144                     }
145                     $sentence = $return;
146                 }
147             }
148         }
149         //$arrayTemp = explode(' ',$return);
150         $arrayTemp = explode(' ',$sentence);
151         $return = '';
152         $returnOR = '';
153         for ($i = 0; $i < count($arrayTemp); $i++) {
154             if ($i == 0) {
155                 $return = trim($arrayTemp[$i]);
```

DecsBase.php

```
161         }
162         return array('AND_WORDS'=>pg_escape_string
    ($return),'OR_WORDS'=>pg_escape_string($returnOR));
163     }
164 
165     function getSentenceFragments($sentence = null,
    $language = null, $param = 2)
166     {
167         $words = $this->getSqlWords($sentence,
    $language);
168         $words = explode(' & ',$words['AND_WORDS']);
169         $return = array();
170         if (count($words) > 1) {
171             $nWords = count($words);
172             for ($condition = count($words) - 1;
    $condition > $param; $condition--){
173                 for ($i = 0; $i <= (($nWords -
    $condition)); $i++){
174                     $string = '';
175                     for ($index = $i; $index < $i +
    $condition; $index++){
176                         if ($index == $i) {
177                             $string = $words[$index];
178                         }else{
179                             $string .= ' & '.$words
    [$index];
180                         }
181                     }
182                     array_push($return,$string);
183                 }
184             }
185         }
```

DecsBase.php

```
193         $return = array();
194         if (count($words) > 1) {
195             $nWords = count($words);
196             for ($condition = count($words) - 1;
    $condition > $param; $condition--){
197                 for ($i = 0; $i <= (($nWords -
    $condition)); $i++){
198                     $string = '';
199                     for ($index = $i; $index < $i +
    $condition; $index++){
200                         if ($index == $i) {
201                             $string = $words[$index];
202                         }else{
203                             $string .= ' | '.$words
    [$index];
204                         }
205                     }
206                     array_push($return,$string);
207                 }
208             }
209         }
210         return $return;
211     }
212
213
214     function highestRelevancySearch($table, $sentence =
    null, $language = null)
215     {
216         if ($sentence != null) {
217             $this->setSentence($sentence);
218         }
219         if ($language != null) {
```

DecsBase.php

```
225         $select = $this->db->select()
226             ->from(array('query' => new Zend_Db_Expr
    ($table.", to_tsquery('".$this->dictionary."', '".
    $sqlWords['AND_WORDS']."')")),new Zend_Db_Expr
    ($tableColumns[0].",".$tableColumns[1].",ts_rank_cd(".
    $tableColumns[2].",query) AS rank"))
227             ->where('lang = ?',$this->language)
228             ->where($tableColumns[2].' @@ ?',new
    Zend_Db_Expr('query'))
229             ->where($tableColumns[1].' ~* ?','.*'.$this-
    >sentence.'*.')
230             ->order(new Zend_Db_Expr('rank DESC'));
231         //echo $select->__toString()."<br><br>";
232         $rowSet = $select->query()->fetchAll();
233         $text = array();
234         foreach ($rowSet as $row) {
235             if (!in_array($row[$tableColumns[1]],$text))
    {
236                 array_push($text,$row[$tableColumns[1]]);
237                 $relevancy[$row[$tableColumns[0]]] = $row
    ['rank'];
238             }
239         }
240         return $relevancy;
241     }
242
243     function veryHighRelevancySearch($table, $sentence =
    null, $language = null)
244     {
245         if ($sentence != null) {
246             $this->setSentence($sentence);
247         }
```

DecsBase.php

```
257             ->where($tableColumns[2].' @@ ?',new
    Zend_Db_Expr('query'))
258             ->orWhere($tableColumns[1].' ~* ?','.*'.
    $this->sentence.'*.')
259             ->order(new Zend_Db_Expr('rank DESC'));
260         //echo $select->__toString()."<br><br>";
261         $rowSet = $select->query()->fetchAll();
262         $text = array();
263         foreach ($rowSet as $row) {
264             if (!in_array($row[$tableColumns[1]],$text))
    {
265                 array_push($text,$row[$tableColumns[1]]);
266                 $relevancy[$row[$tableColumns[0]]] = $row
    ['rank'];
267             }
268         }
269         return $relevancy;
270     }
271
272     function highRelevancySearch($table, $sentence =
    null, $language = null)
273     {
274         if ($sentence != null) {
275             $this->setSentence($sentence);
276         }
277         if ($language != null) {
278             $this->setLanguage($language);
279         }
280         $tableColumns = $this->auxiliary->getTableColumns
    ($table);
281         $relevancy = array();
282         $sqlWords = $this->getSqlWords();
```

DecsBase.php

```
289                     ->where($tableColumns[2].' @@ ?',new
   Zend_Db_Expr('query'))
290                     ->orWhere($tableColumns[1].'
   ~* ?','.*'.$this->auxiliary->getCleanSentence
   ($senteceFragments[$i]).'*.')
291                     ->order(new Zend_Db_Expr('rank
   DESC'));
292                 //echo $select->__toString()."<br><br>";
293                 $rowSet = $select->query()->fetchAll();
294                 $text = array();
295                 foreach ($rowSet as $row) {
296                     echo $row[$tableColumns[1]];
297                     if (!in_array($row[$tableColumns[1]],
   $text)) {
298                         array_push($text,$row
   [$tableColumns[1]]);
299                         if (!array_key_exists($row
   [$tableColumns[0]],$relevancy)) {
300                             $relevancy[$row[$tableColumns
   [0]]] = $row['rank'];
301                             // caso já esteja no array e
   o ranking é mais baixo que o atual, substitui
302                         } elseif ($relevancy[$row
   [$tableColumns[0]]] < $row['rank']) {
303                             $relevancy[$row[$tableColumns
   [0]]] = $row['rank'];
304                         }
305                     }
306                 }
307             }
308             arsort($relevancy,SORT_NUMERIC);
309         }
```

DecsBase.php

```
321         $tableColumns = $this->auxiliary->getTableColumns
    ($table);
322         $relevancy = array();
323         $sqlWords = $this->getSqlWords();
324         $senteceFragments = $this->getSentenceFragments
    (null, null, 1);
325         if ($senteceFragments != false) {
326             for ($i = 0; $i < count($senteceFragments);
    $i++) {
327                 if (count(explode(' & ',$senteceFragments
    [$i])) < 3) {
328                     $select = $this->db->select()
329                         ->from(array('query' => new
    Zend_Db_Expr($table.", to_tsquery('".$this-
    >dictionary."', '".$senteceFragments[$i]."')")),new
    Zend_Db_Expr($tableColumns[0].",".$tableColumns
    [1].",ts_rank_cd(".$tableColumns[2].",query) AS rank"))
330                         ->where('lang = ?',$this-
    >language)
331                         ->where($tableColumns[2].'
    @@ ?',new Zend_Db_Expr('query'))
332                         ->orWhere($tableColumns[1].'
    ~* ?','.*'.$this->auxiliary->getCleanSentence
    ($senteceFragments[$i]).'*.')
333                         ->order(new Zend_Db_Expr('rank
    DESC'));
334                     //echo $select->__toString
    ()."<br><br>";
335                     $rowSet = $select->query()->fetchAll
    ();
336                     $text = array();
337                     foreach ($rowSet as $row) {
```

DecsBase.php

```
353         return $relevancy;
354     }
355 
356     function lowRelevancySearch($table, $sentence = null,
    $language = null)
357     {
358         if ($sentence != null) {
359             $this->setSentence($sentence);
360         }
361         if ($language != null) {
362             $this->setLanguage($language);
363         }
364         $tableColumns = $this->auxiliary->getTableColumns
    ($table);
365         $relevancy = array();
366         $sqlWords = $this->getSqlWords();
367         $senteceFragments = $this->getOrSentenceFragments
    (null, null, 1);
368         if ($senteceFragments != false) {
369             for ($i = 0; $i < count($senteceFragments); $i+
    +) {
370                 if (count(explode(' | ',$senteceFragments
    [$i])) < 3) {
371                     $select = $this->db->select()
372                         ->from(array('query' => new
    Zend_Db_Expr($table.", to_tsquery('".$this->dictionary."',
    '".$senteceFragments[$i]."')")),new Zend_Db_Expr
    ($tableColumns[0].",".$tableColumns[1].",ts_rank_cd(".
    $tableColumns[2].",query) AS rank"))
373                         ->where('lang = ?',$this->language)
374                         ->where($tableColumns[2].'
    @@ ?',new Zend_Db_Expr('query'))
```

DecsBase.php

```
385                             } elseif ($relevancy[$row
   [$tableColumns[0]]] < $row['rank']) {
386                                 $relevancy[$row
   [$tableColumns[0]]] = $row['rank'];
387                             }
388                         }
389                     }
390                 }
391
392             }
393             arsort($relevancy,SORT_NUMERIC);
394         }
395         return $relevancy;
396     }
397
398     function searchDefinition($sentence = null, $language =
   null)
399     {
400         $moreRelevant = $this->moreRelevantSearch
   ('decs_definition', $sentence, $language);
401         $lessRelevant = $this->lessRelevantSearch
   ('decs_definition', $sentence, $language);
402         $this->searchResult['definition'] = $this-
   >auxiliary->doParcialRank($moreRelevant, $lessRelevant);
403     }
404
405     function searchDescriptor($sentence = null, $language =
   null)
406     {
407         $moreRelevant = $this->moreRelevantSearch
   ('decs_descriptor', $sentence, $language);
            $lessRelevant = $this->lessRelevantSearch
```

Page 13

DecsBase.php

```
417     }
418
419     /**
420      * Método que retorna os códigos Decs dos descendentes
421      * do parâmetro $decsId.
422      *
423      * @param string $decsId código Decs
424      * @param string $language pode ser 'pt','es' ou 'en'
425      */
426     function getDescendants($decsId, $language = null)
427     {
428         if ($language != null) {
429             $this->setLanguage($language);
430         }
431         $result = $this->db->select()
432                 ->from(array('p' => 'decs_descriptor'),array
    ('descriptor'))
433                 ->join(array('l' =>
    'decs_descendant'),'p.decs_id = l.descendant',array
    ('descendant'))
434                 ->where('l.decs_id = ?',$decsId)
435                 ->where('p.lang = ?',$this->language);
436         return $result->query()->fetchAll();
437     }
438
439     /**
440      * Método que gera uma árvore hierárquica com os
    códigos Decs
441      * dos descendentes do parâmetro $decsId.
442      *
443      * @param string $decsId código Decs
444      * @param string $language pode ser 'pt','es' ou 'en'
```

DecsBase.php

```
449             $descendants = $this->getDescendants($decsId);
450             $descriptor = $this->getDescriptor($decsId);
451         } else {
452             $descendants = $this->getDescendants($decsId,
    $language);
453             $descriptor = $this->getDescriptor($decsId,
    $language);
454         }
455         if (count($descendants) == 0) {
456             return false;
457         } else {
458             $arvoreDescendentes = "<div class=\"dtree
    \"><script type=\"text/javascript\"> d = new dTree('d','".
    $this->urlBase."/public/images/'); d.add(0,-1,'".$decsId."
    - ".$descriptor."');";
459             for ($i = 0; $i < count($descendants); $i++) {
460                 $arvoreDescendentes .= "d.add(".($i
    +1).",0,'".$descendants[$i]['descendant']." - ".$descendants
    [$i]['descriptor']."');";
461             }
462             $arvoreDescendentes .= 'document.write(d);
    d.openAll(); </script></div>';
463             return $arvoreDescendentes;
464         }
465     }
466
467     /**
468      * Método que retorna o código Decs do antecessor
469      * do parâmetro $decsId.
470      *
471      * @param string $decsId código Decs
472      */
```

DecsBase.php

```
481                     $return .= $arrayTemp[$i];
482                 } else {
483                     $return .= '.'.$arrayTemp[$i];
484                 }
485             }
486             return $return;
487         } elseif (count($arrayTemp)==1 && strlen($arrayTemp
    [0])>2) {
488             if (substr_count($arrayTemp[0], 'HP')>0 ||
    substr_count($arrayTemp[0], 'SH')>0
489             || substr_count($arrayTemp[0], 'SP')>0 ||
    substr_count($arrayTemp[0], 'VS')>0) {
490                 return substr($arrayTemp[0],0,2);
491             } else {
492                 return substr($arrayTemp[0],0,1);
493             }
494         } else {
495             return false;
496         }
497     }
498 
499     /**
500      * Método que retorna os códigos Decs dos antecessores
501      * do parâmetro $decsId.
502      *
503      * @param string $decsId código Decs
504      * @param string $language pode ser 'pt','es' ou 'en'
505      */
506     function getAncestors($decsId, $language = null)
507     {
508         if ($language != null) {
509             $this->setLanguage($language);
```

DecsBase.php

```
513                 ->join(array('l' =>
    'decs_ancestor'),'p.decs_id = l.ancestor',array('ancestor'))
514                 ->where('l.decs_id = ?',$decsId)
515                 ->where('p.lang = ?',$this->language);
516         return $result->query()->fetchAll();
517     }
518 
519     /**
520      * Método que gera uma árvore hierárquica com os
    códigos Decs
521      * dos antecessores do parâmetro $decsId.
522      *
523      * @param string $decsId código Decs
524      * @param string $language pode ser 'pt','es' ou 'en'
525      */
526     function ancestorsTree($decsId, $language = null)
527     {
528         if ($language == null) {
529             $ancestors = $this->getAncestors($decsId);
530         } else {
531             $ancestors = $this->getAncestors($decsId,
    $language);
532         }
533         if (count($ancestors) == 0) {
534             return false;
535         } else {
536             $arrayArvore = array();
537             $flag = 0;
538             $index = 0;
539             $ancestorsTree = "<div class=\"dtree\"><script
    type=\"text/javascript\"> d = new dTree('d','".$this-
    >urlBase."/public/images/');";
```

DecsBase.php

```
545                         $arrayArvore[$ancestors[$i]
   ['ancestor']] = array('nodo'=> $index,'nodoPai'=>
   -1,'codigo'=> $ancestors[$i]['ancestor'],'descriptor'=>
   $ancestors[$i]['descriptor']);
546                         $ancestorsTree .= "d.add(".
   $index.",-1,'".$ancestors[$i]['ancestor']." - ".$ancestors
   [$i]['descriptor']."');";
547                     } else {
548                         $arrayArvore[$ancestors[$i]
   ['ancestor']] = array('nodo'=> 0,'nodoPai'=> -1,'codigo'=>
   $ancestors[$i]['ancestor'],'descriptor'=> $ancestors[$i]
   ['descriptor']);
549                         $ancestorsTree .=  "d.add(0,-1,'".
   $ancestors[$i]['ancestor']." - ".$ancestors[$i]
   ['descriptor']."');";
550                         $flag = 1;
551                     }
552                 } else {
553                     $index = $index + 1;
554                     $parentIndex = $arrayArvore[$this-
   >getAncestor($ancestors[$i]['ancestor'])]['nodo'];
555                     $arrayArvore[$ancestors[$i]
   ['ancestor']] = array('nodo'=> $index,'nodoPai'=>
   $parentIndex,'codigo'=> $ancestors[$i]
   ['ancestor'],'descriptor'=> $ancestors[$i]['descriptor']);
556                     $ancestorsTree .= "d.add(".$index.",".
   $parentIndex.",'".$ancestors[$i]['ancestor']." - ".
   $ancestors[$i]['descriptor']."');";
557                 }
558             }
                $ancestorsTree .= "d.add(".($index+1).",".
   $index.",'".$decsId." - ".$this->getDescriptor($decsId,
```

DecsBase.php

```
577             $this->setLanguage($language);
578         }
579         $result = $this->decsDefinition->select();
580         $result->where('decs_id = ?',$decsId);
581         $result->where('lang = ?',$this->language);
582         return $this->decsDefinition->fetchRow($result)-
   >definition;
583     }
584 
585     /**
586      * Método que retorna a descrição do código Decs passado
587      * na $língua especificada.
588      *
589      * @param string $decsId código Decs
590      * @param string $language pode ser 'pt','es' ou 'en'
591      */
592     function getDescriptor($decsId, $language = null)
593     {
594         if ($language != null) {
595             $this->setLanguage($language);
596         }
597         $result = $this->decsDescriptor->select();
598         $result->where('decs_id = ?',$decsId);
599         $result->where('lang = ?',$this->language);
600         return $this->decsDescriptor->fetchRow($result)-
   >descriptor;
601     }
602 
603     /**
604      * Método que retorna o sinônimo do código Decs passado
605      * no idioma $language especificada.
606      *
```

DecsBase.php

```
609     */
610    function getSynonym($decsId, $language = null)
611    {
612        if ($language != null) {
613            $this->setLanguage($language);
614        }
615        $result = $this->db->select()
616                ->distinct()
617                ->from(array('p' => 'decs_synonym'),array
   ('synonym'))
618                ->where('p.decs_id = ?',$decsId)
619                ->where('p.lang = ?',$this->language);
620        return $result->query()->fetchAll();
621    }
622
623    /**
624     * Método que retorna o indexing annotation(anotação de
   indexação)
625     * do código Decs passado na $língua especificada.
626     *
627     * @param string $decsId código Decs
628     * @param string $language pode ser 'pt','es' ou 'en'
629     */
630    function getIndexingAnnotation($decsId, $language =
   null)
631    {
632        if ($language != null) {
633            $this->setLanguage($language);
634        }
635        $result = $this->decsAnnotation->select();
636        $result->where('decs_id = ?',$decsId);
637        $result->where('lang = ?',$this->language);
```

DecsBase.php

```
641     /**
642      * Método que retorna o unique_identifier e o
    register_number
643      * do código Decs passado.
644      *
645      * @param string $decsId código Decs
646      */
647     function getIdentifiers($decsId)
648     {
649         $result = $this->decs->select();
650         $result->where('decs_id = ?',$decsId);
651         return array('identificador' => $this->decs-
    >fetchRow($result)->unique_identifier,
652                      'registro' => $this->decs->fetchRow
    ($result)->register_number);
653     }
654 }
```

SnomedBase.php

```php
<?php
/**
 * CLBJ's TCC
 *
 * @category    CLBJ
 * @package     application/classes
 * @license     Parte integrante do trabalho de conclusão do curso de Ciências da Computação
 * da Universidade Federal de Santa Catarina, a reprodução, alteração e utilização sem autorização
 * do autor são proibidas.
 * @author      Cloves Langendorf Barcellos Junior
 * @version     0.2 03/03/2008
 */
class SnomedBase
{
    protected $db;
    protected $sentence;

    protected $arvore;
    protected $treeNodes;
    protected $parentNode;
    protected $childNode;

    protected $rootConcept;
    protected $conceitosBase;
    protected $link;

    //tabelas Snomed
    protected $sctConcept;
    protected $sctDescription;
    protected $sctRelationship;
```

SnomedBase.php

```
33    {
34        // concept_id do SNOMED CT Concept (SNOMED RT
  +CTV3)  XU05D   R-00000 1
35        $this->rootConcept = 138875005;
36
37        // Top-Level Concepts
38        $this->conceitosBase = array
  (123037004,243796009,308916002,272379006,246061005,
39
  404684003,363787002,257495001,373873005,78621006,26078700
  4,71388002,362981000,
40        48176007,123038009,254291000,105590001);
41
42        $this->treeNodes = array();
43
44        // conceito IS_A utilizado para linkar pais e
  filhos na árvore hierárquica
45        $this->link = 116680003;
46
47        $this->db = Zend_Registry::get('db');
48
49        //instanciando tabelas
50        $this->sctConcept = new SctConcept();
51        $this->sctDescription = new SctDescription();
52        $this->sctRelationship = new SctRelationship();
53    }
54
55    /**
56     * Método para atribuir o diretório raiz da
  aplicação.
57     *
58     * @param string $url
```

SnomedBase.php

```
65      /**
66       * Método para atribuir o idioma e o dicionário
  correntes.
67       *
68       * @param string $language pode ser 'pt','es' ou 'en'
69       */
70      function setLanguage($language)
71      {
72          $this->language = $language;
73
74          //atribui os dicionários ispell
75          switch ($this->language) {
76              case 'pt':
77                  $this->dictionary = Zend_Registry::get
  ('dictPT');
78                  $this->stopWords = Zend_Registry::get
  ('stopWordsPT');
79                  break;
80              case 'en':
81                  $this->dictionary = Zend_Registry::get
  ('dictEN');
82                  $this->stopWords = Zend_Registry::get
  ('stopWordsEN');
83                  break;
84              case 'es':
85                  $this->dictionary = Zend_Registry::get
  ('dictES');
86                  $this->stopWords = Zend_Registry::get
  ('stopWordsES');
87                  break;
88          }
89      }
```

SnomedBase.php

```
 97     function setSentence($string)
 98     {
 99         $this->sentence = $string;
100     }
101 
102     /**
103      * Método responsável por retornar a frase utilizada
104      * nas pesquisas.
105      *
106      */
107     function getSentence()
108     {
109         return $this->sentence;
110     }
111 
112     /**
113      * Método responsável por retornar o resultado da
    busca.
114      */
115     function getSearchResult()
116     {
117         return $this->searchResult;
118     }
119 
120     /**
121      * Método responsável por filtrar o texto a ser
    pesquisado,
122      * retornando uma versão otimizada para pesquisa com
    o tsearch2.
123      * @param string $sentence frase a ser avaliada
124      * @param string $language pode ser 'pt','es' ou 'en'
125      */
```

SnomedBase.php

```
129             $this->setSentence($sentence);
130         }
131         if ($language != null) {
132             $this->setLanguage($language);
133         }
134 
135         $return = '';
136         $sentence = $this->sentence;
137         for ($i = 0; $i < count($this->stopWords); $i++)
    {
138             $comparador = " ".$this->stopWords[$i]." ";
139             if (substr_count($sentence,$comparador) > 0)
    {
140                 $arrayTemp = explode($comparador,
    $sentence);
141                 for ($j = 0; $j < count($arrayTemp); $j+
    +) {
142                     if ($j == 0) {
143                         $return = trim($arrayTemp[$j]);
144                     } else {
145                         $return .= ' '.trim($arrayTemp
    [$j]);
146                     }
147                     $sentence = $return;
148                 }
149             }
150         }
151         //$arrayTemp = explode(' ',$return);
152         $arrayTemp = explode(' ',$sentence);
153         $return = '';
154         $returnOR = '';
155         for ($i = 0; $i < count($arrayTemp); $i++) {
```

SnomedBase.php

```
161                 $returnOR .= ' | '.trim($arrayTemp[$i]);
162             }
163         }
164         //return pg_escape_string($return);
165         return array('AND_WORDS'=>pg_escape_string
    ($return),'OR_WORDS'=>pg_escape_string($returnOR));
166     }
167 
168     function getSentenceFragments($sentence = null,
    $language = null, $param = 2)
169     {
170         $words = $this->getSqlWords($sentence,
    $language);
171         $words = explode(' & ',$words['AND_WORDS']);
172         $return = array();
173         if (count($words) > 1) {
174             $nWords = count($words);
175             for ($condition = count($words) - 1;
    $condition > $param; $condition--){
176                 for ($i = 0; $i <= (($nWords -
    $condition)); $i++){
177                     $string = '';
178                     for ($index = $i; $index < $i +
    $condition; $index++){
179                         if ($index == $i) {
180                             $string = $words[$index];
181                         }else{
182                             $string .= ' & '.$words
    [$index];
183                         }
184                     }
185                     array_push($return,$string);
```

SnomedBase.php

```
193     * Método que retorna todas as informações do
194     * concept_id passado;
195     *
196     * @param integer $conceptId concept Id
197     * @param string $tableColumn coluna da tabela que
   deseja retornar
198     */
199    function getInfo($conceptId,$tableColumn = null)
200    {
201        $select = $this->sctConcept->select();
202        $select->where('concept_id = ?',$conceptId);
203        if ($tableColumn != null) {
204            switch ($tableColumn) {
205            case 'nome':
206                return $this->sctConcept->fetchRow
   ($select)->fully_specified_name;
207            case 'status':
208                return $this->sctConcept->fetchRow
   ($select)->concept_status;
209            case 'ctv3':
210                return $this->sctConcept->fetchRow
   ($select)->ctv3_id;
211            case 'codigo':
212                return $this->sctConcept->fetchRow
   ($select)->snomed_id;
213            case 'primitivo':
214                return $this->sctConcept->fetchRow
   ($select)->is_primitive;
215            }
216        } else {
217            return array('nome' => $this->sctConcept-
   >fetchRow($select)->fully_specified_name,
```

SnomedBase.php

```
225     /**
226      * Método que retorna os campos de descrição do
    concept_id passado.
227      *
228      * @param integer $conceptId concept Id
229      * @param string $tableColumn coluna da tabela que
    deseja retornar
230      */
231     function getDescription($conceptId,$tableColumn =
    null)
232     {
233         $select = $this->sctDescription->select();
234         $select->where('concept_id = ?',$conceptId);
235         $select->where('description_type = ?',1);
236         $select->order(new Zend_Db_Expr
    ('description_status DESC'));
237         if ($tableColumn != null) {
238             switch ($tableColumn) {
239             case 'descricao':
240                 return $this->sctDescription->fetchRow
    ($select)->term;
241             case 'status':
242                 return $this->sctDescription->fetchRow
    ($select)->description_status;
243             case 'tipo':
244                 return $this->sctDescription->fetchRow
    ($select)->description_type;
245             case 'codigo':
246                 return $this->sctDescription->fetchRow
    ($select)->description_id;
247             }
248         } else {
```

SnomedBase.php

```
257     * Método que retorna o concept_id do antecessor
258     * do parâmetro $conceptId.
259     *
260     * @param integer $conceptId concept id
261     */
262    function getAncestor($conceptId)
263    {
264        $select = $this->sctRelationship->select();
265        $select->where('concept_id1 = ?',$conceptId);
266        $select->where('relationship_type = ?',$this-
   >link);
267        return $this->sctRelationship->fetchRow($select)-
   >concept_id2;
268    }
269
270    /**
271     * Método que retorna os concept_ids dos antecessores
272     * do parâmetro $conceptId.
273     *
274     * @param integer $conceptId concept id
275     */
276    function getAncestors($conceptId)
277    {
278        $return[0] = $conceptId;
279        $conceitoAtual = $conceptId;
280        while ($this->getAncestor($conceitoAtual) !=
   $this->rootConcept) {
281            array_push($return,$this->getAncestor
   ($conceitoAtual));
282            $conceitoAtual = $this->getAncestor
   ($conceitoAtual);
283        }
```

SnomedBase.php

```
289      * dos antecessores do parâmetro $conceptId.
290      *
291      * @param integer $conceptId concept id
292      */
293     function ancestorsTree($conceptId)
294     {
295         $ancestors = $this->getAncestors($conceptId);
296         $ancestorsTree = "<div class=\"dtree\"><script type=
    \"text/javascript\"> d = new dTree('d','".$this->urlBase."/
    public/images/');";
297         $ancestorsTree .= "d.add(0,-1,'".$conceptId." - ".
    $this->getDescription($this->rootConcept,'descricao')."');";
298         $parentNode = 0;
299         while ($antecessor = array_pop($ancestors)) {
300             $ancestorsTree .= "d.add(".($parentNode +
    1).",".$parentNode.",'".$antecessor." - ".$this-
    >getDescription($antecessor,'descricao')."');";
301             $parentNode++;
302         }
303         if ($ancestorsTree != '') {
304             $ancestorsTree .= 'document.write(d); d.openAll
    (); </script></div>';
305         }
306         return $ancestorsTree;
307     }
308
309     function addNode($conceptId)
310     {
311         $this->treeNodes[$conceptId] = array('nodo' =>
    $this->childNode,'antecessor' => $this->getAncestor
    ($conceptId));
         $antecessor = $this->treeNodes[$this->getAncestor
```

SnomedBase.php

```
321     */
322    function getDescendants($conceptId)
323    {
324        $select = $this->db->select()
325                ->from(array('p' =>
   'sct_relationship'),array('concept_id1'))
326                ->where('concept_id2 = ?',$conceptId)
327                ->where('relationship_type = ?',$this-
   >link);
328        $select =  $select->query()->fetchAll();
329        $return = array();
330        for ($i = 0; $i < count($select); $i++) {
331            array_push($return,$select[$i]['concept_id1']);
332        }
333        //return $return;
334
335        foreach ($return as &$value) {
336            $this->addNode($value);
337            $this->childNode++;
338            $this->getDescendants($value);
339        }
340    }
341
342    /**
343     * Método que gera uma árvore hierárquica com os
   concept_id
344     * dos descendentes do parâmetro $conceptId.
345     *
346     * @param integer $conceptId concept id
347     */
348    function descendantsTree($conceptId)
349    {
```

SnomedBase.php

```
353         $this->childNode = 1;
354         $this->getDescendants($conceptId);
355         $this->arvore .= 'document.write(d); d.openAll(); </
    script></div>';
356         return $this->arvore;
357     }
358 
359     function highestRelevancySearch($table, $sentence =
    null, $language = null)
360     {
361         if ($sentence != null) {
362             $this->setSentence($sentence);
363         }
364         if ($language != null) {
365             $this->setLanguage($language);
366         }
367         $tableColumns = $this->auxiliary->getTableColumns
    ($table);
368         $relevancy = array();
369         $sqlWords = $this->getSqlWords();
370         $select = $this->db->select()
371             ->from(array('query' => new Zend_Db_Expr
    ($table.", to_tsquery('".$this->dictionary."', '".$sqlWords
    ['AND_WORDS']."')")),new Zend_Db_Expr($tableColumns[0].",".
    $tableColumns[1].",ts_rank_cd(".$tableColumns[2].",query)
    AS rank"))
372             ->where('lang = ?',$this->language)
373             ->where($tableColumns[2].' @@ ?',new
    Zend_Db_Expr('query'))
374             ->where($tableColumns[1].' ~* ?','.*'.$this-
    >sentence.'*.')
375             ->order(new Zend_Db_Expr('rank DESC'));
```

SnomedBase.php

```
385         return $relevancy;
386     }
387 
388     function veryHighRelevancySearch($table, $sentence =
    null, $language = null)
389     {
390         if ($sentence != null) {
391             $this->setSentence($sentence);
392         }
393         if ($language != null) {
394             $this->setLanguage($language);
395         }
396         $tableColumns = $this->auxiliary->getTableColumns
    ($table);
397         $relevancy = array();
398         $sqlWords = $this->getSqlWords();
399         var_dump($sqlWords);
400         $select = $this->db->select()
401             ->from(array('query' => new Zend_Db_Expr
    ($table.", to_tsquery('".$this->dictionary."', '".$sqlWords
    ['AND_WORDS']."')")),new Zend_Db_Expr($tableColumns[0].",".
    $tableColumns[1].",ts_rank_cd(".$tableColumns[2].",query)
    AS rank"))
402             ->where('lang = ?',$this->language)
403             ->where($tableColumns[2].' @@ ?',new
    Zend_Db_Expr('query'))
404             ->orWhere($tableColumns[1].' ~* ?','.*'.$this-
    >sentence.'*.')
405             ->order(new Zend_Db_Expr('rank DESC'));
406         //echo $select->__toString()."<br><br>";
407         $rowSet = $select->query()->fetchAll();
408         $text = array();
```

SnomedBase.php

```
417
418     function highRelevancySearch($table, $sentence = null,
    $language = null)
419     {
420         if ($sentence != null) {
421             $this->setSentence($sentence);
422         }
423         if ($language != null) {
424             $this->setLanguage($language);
425         }
426         $tableColumns = $this->auxiliary->getTableColumns
    ($table);
427         $relevancy = array();
428         $sqlWords = $this->getSqlWords();
429         var_dump($sqlWords);
430         $senteceFragments = $this->getSentenceFragments();
431         var_dump($senteceFragments);
432         //$busca=str_replace("-","/",$_REQUEST['busca']);
433         if ($senteceFragments != false) {
434             for ($i = 0; $i < count($senteceFragments); $i+
    +) {
435                 $select = $this->db->select()
436                     ->from(array('query' => new Zend_Db_Expr
    ($table.", to_tsquery('".$this->dictionary."', '".
    $senteceFragments[$i]."')")),new Zend_Db_Expr($tableColumns
    [0].",".$tableColumns[1].",ts_rank_cd(".$tableColumns
    [2].",query) AS rank"))
437                     ->where('lang = ?',$this->language)
438                     ->where($tableColumns[2].' @@ ?',new
    Zend_Db_Expr('query'))
                    ->orWhere($tableColumns[1].' ~* ?','.*'.
    $this->auxiliary->getCleanSentence($senteceFragments
```

SnomedBase.php

```
449                             $relevancy[$row[$tableColumns
   [0]]] = $row['rank'];
450                             // caso já esteja no array e o
   ranking é mais baixo que o atual, substitui
451                         } elseif ($relevancy[$row
   [$tableColumns[0]]] < $row['rank']) {
452                             $relevancy[$row[$tableColumns
   [0]]] = $row['rank'];
453                         }
454                     }
455                 }
456 
457                 //var_dump($relevancy);
458                 //echo count($relevancy)."<br>";
459 
460             }
461             arsort($relevancy,SORT_NUMERIC);
462         }
463         return $relevancy;
464     }
465 
466     function mediumRelevancySearch($table, $sentence =
   null, $language = null)
467     {
468         if ($sentence != null) {
469             $this->setSentence($sentence);
470         }
471         if ($language != null) {
472             $this->setLanguage($language);
473         }
474         $tableColumns = $this->auxiliary->getTableColumns
   ($table);
```

SnomedBase.php

```
481                     $select = $this->db->select()
482                         ->from(array('query' => new
   Zend_Db_Expr($table.", to_tsquery('".$this->dictionary."',
   '".$senteceFragments[$i]."')")),new Zend_Db_Expr
   ($tableColumns[0].",".$tableColumns[1].",ts_rank_cd(".
   $tableColumns[2].",query) AS rank"))
483                         ->where('lang = ?',$this->language)
484                         ->where($tableColumns[2].'
   @@ ?',new Zend_Db_Expr('query'))
485                         ->orWhere($tableColumns[1].'
   ~* ?','.*'.$this->auxiliary->getCleanSentence
   ($senteceFragments[$i]).'*.')
486                         ->order(new Zend_Db_Expr('rank
   DESC'));
487                     echo $select->__toString()."<br><br>";
488                     $rowSet = $select->query()->fetchAll();
489                     $text = array();
490                     foreach ($rowSet as $row) {
491                         if (!in_array($row[$tableColumns
   [1]],$text)) {
492                             array_push($text,$row
   [$tableColumns[1]]);
493                             if (!array_key_exists($row
   [$tableColumns[0]],$relevancy)) {
494                                 $relevancy[$row
   [$tableColumns[0]]] = $row['rank'];
495                                 // caso já esteja no array
   e o ranking é mais baixo que o atual, substitui
496                             } elseif ($relevancy[$row
   [$tableColumns[0]]] < $row['rank']) {
497                                 $relevancy[$row
   [$tableColumns[0]]] = $row['rank'];
```

SnomedBase.php

```
513             $this->setSentence($sentence);
514         }
515         if ($language != null) {
516             $this->setLanguage($language);
517         }
518         $tableColumns = $this->auxiliary->getTableColumns
    ($table);
519         $lessRelevant = array();
520         $select = $this->db->select()
521             ->from(array('query' => new Zend_Db_Expr
    ($table.", to_tsquery('".$this->dictionary."', '".$this-
    >getSqlWords()."')")),new Zend_Db_Expr($tableColumns[0].",".
    $tableColumns[1].",ts_rank_cd(".$tableColumns[2].",query)
    AS rank"))
522             ->where('lang = ?',$this->language)
523             ->where($tableColumns[2].' @@ ?',new
    Zend_Db_Expr('query'))
524             ->order(new Zend_Db_Expr('rank DESC'));
525         echo $select->__toString();
526         $rowSet = $select->query()->fetchAll();
527         $text = array();
528         foreach ($rowSet as $row) {
529             if (!in_array($row[$tableColumns[1]],$text)) {
530                 array_push($text,$row[$tableColumns[1]]);
531                 $lessRelevant[$row[$tableColumns[0]]] = $row
    ['rank'];
532             }
533         }
534         return $lessRelevant;
535     }
536}
```

Auxiliary.php

```php
<?php
/**
 * CLBJ's TCC
 *
 * @category    CLBJ
 * @package     application/classes
 * @license     Parte integrante do trabalho de
 conclusão do curso de Ciências da Computação
 * da Universidade Federal de Santa Catarina, a
 reprodução, alteração e utilização sem autorização
 * do autor são proibidas.
 * @author      Cloves Langendorf Barcellos Junior
 * @version     0.1 30/08/2008
 */
class Auxiliary
{
    protected $db;
    protected $base;

    public function Auxiliary()
    {
        $this->db = Zend_Registry::get('db');
        $this->base = Zend_Registry::get('base');
    }

    function parseQuery( $term ) {
        $term = preg_replace('/\\\/', '', $term);

        $term = preg_replace('/\s*\(\s*/', ' (', $term);
        $term = preg_replace('/\s*\)\s*/', ') ', $term);

        $term = preg_replace('/:/', ' ', $term);
```

Auxiliary.php

```
33        $m = array();
34        if( preg_match_all('/([-!]?)(\S+)\s*/', $term,
  $m, PREG_SET_ORDER ) ) {
35            foreach( $m as $terms ) {
36                if (strlen($terms[1])) {
37                    $searchstring .= ' & !';
38                }
39                if (strtolower($terms[2]) === 'and') {
40                    $searchstring .= ' & ';
41                }
42                else if (strtolower($terms[2]) === 'or'
  or $terms[2] === '|') {
43                    $searchstring .= ' | ';
44                }
45                else if (strtolower($terms[2]) === 'not')
  {
46                    $searchstring .= ' & !';
47                }
48                else {
49                    $searchstring .= " & $terms[2]";
50                }
51            }
52        }
53
54        $searchstring = preg_replace('/^[\s\&\|]+/', '',
  $searchstring);
55
56        $searchstring = preg_replace('/([\!\&\|]) +(?:[\&
  \|] +)+/', "$1 ", $searchstring);
57
58        $searchstring = preg_replace('/([^ ])[\!\&\|]/',
  "$1", $searchstring);
```

Page 2

Auxiliary.php

```
65            $searchstring =  'NULL';
66        } else if ( is_bool( $searchstring ) ) {
67            $searchstring =  intval( $searchstring );
68        }
69        $searchstring =  "'" . pg_escape_string
  ($searchstring) . "'";
70        return $searchstring;
71    }
72
73    function getCleanSentence($fragments)
74    {
75        $temp = explode(' & ',$fragments);
76        $return = "";
77        for ($i = 0; $i < count($temp); $i++) {
78            $return .= $temp[$i]." ";
79        }
80        return trim($return);
81    }
82
83    function doParcialRank(array $moreRelevant, array
  $lessRelevant)
84    {
85        $return['moreRelevant'] = $moreRelevant;
86        $keys = array_keys($lessRelevant);
87        $values = array_values($lessRelevant);
88        if (count($lessRelevant) > 0) {
89            for ($i = 0; $i < count($lessRelevant); $i++) {
90                if (!array_key_exists($keys[$i],
  $moreRelevant)) {
91                    $return['lessRelevant'][$keys[$i]] =
  $values[$i];
92                }
```

Auxiliary.php

```
 97         return $return;
 98     }
 99 
100     function getTableColumns($table)
101     {
102         switch ($table) {
103             case 'decs_definition':
104                 return array
    ('decs_id','definition','definition_index');
105             case 'decs_descriptor':
106                 return array
    ('decs_id','descriptor','descriptor_index');
107             case 'decs_indexing_annotation':
108                 return array
    ('decs_id','indexing_annotation','annotation_index');
109             case 'decs_synonym':
110                 return array
    ('decs_id','synonym','synonym_index');
111             case 'decs_qualifier':
112                 return array
    ('acronym','qualifier','qualifier_index');
113             case 'sct_concept':
114                 return array
    ('concept_id','fully_specified_name','name_index');
115             case 'sct_description':
116                 return array
    ('description_id','concept_id','term','term_index');
117         }
118     }
119}
```

Page 4

dictionaries.php

```
1 <?php
2 // definição dos dicionários ispell a serem utilizados
  pelo tsearch2
3 Zend_Registry::set('dictPT','pg_br');
4 Zend_Registry::set('dictEN','pg_en');
5 Zend_Registry::set('dictES','pg_es');
6
7 // definição das stop words
8 Zend_Registry::set('stopWordsPT',array
  ("de","a","o","que","e","do","da","em","um","para","é","co
  m","não",
9 "uma","os","no","se","na","por","mais","as","dos","como","
  mas","foi","ao","ele","das","tem","à","seu",
10 "sua","ou","ser","quando","muito","há","nos","já","está","
  eu","também","só","pelo","pela","até","isso",
11 "ela","entre","era","depois","sem","mesmo","aos","ter","se
  us","quem","nas","me","esse","eles","estão",
12 "você","tinha","foram","essa","num","nem","suas","meu","às
  ","minha","têm","numa","pelos","elas","havia",
13 "seja","qual","será","nós","tenho","lhe","deles","essas","
  esses","pelas","este","fosse","dele","tu","te",
14 "vocês","vos","lhes","meus","minhas","teu","tua","teus","t
  uas","nosso","nossa","nossos","nossas","dela",
15 "delas","esta","estes","estas","aquele","aquela","aqueles
  ","aquelas","isto","aquilo","estou","está",
16 "estamos","estão","estive","esteve","estivemos","estiveram
  ","estava","estávamos","estavam","estivera",
17 "estivéramos","esteja","estejamos","estejam","estivesse","
  estivéssemos","estivessem","estiver","estivermos",
18 "estiverem","hei","há","havemos","hão","houve","houvemos",
  "houveram","houvera","houvéramos","haja","hajamos",
19 "hajam","houvesse","houvéssemos","houvessem","houver","hou
```

dictionaries.php

```
33 "couldn't","mustn't","let's","that's","who's","what's","he
   re's","there's","when's","where's","why's",
34 "how's","a","an","the","and","but","if","or","because","as
   ","until","while","of","at","by","for","with",
35 "about","against","between","into","through","during","bef
   ore","after","above","below","to","from","up",
36 "down","in","out","on","off","over","under","again","furth
   er","then","once","here","there","when","where",
37 "why","how","all","any","both","each","few","more","most",
   "other","some","such","no","nor","not","only",
38 "own","same","so","than","too","very","one","every","least
   ","less","many","now","ever","never","say","says"
39 ,"said","also","get","go","goes","just","made","make","put
   ","see","seen","whether","like","well","back",
40 "even","still","way","take","since","another","however","t
   wo","three","four","five","first","second","new",
41 "old","high","long"));
42 Zend_Registry::set('stopWordsES',array
   ("de","la","que","el","en","y","a","los","del","se","las",
   "por",
43 "un","para","con","no","una","su","al","es","lo","como","m
   ás","pero","sus","le","ya","o","fue","este",
44 "ha","sí","porque","esta","son","entre","está","cuando","m
   uy","sin","sobre","ser","tiene","también","me",
45 "hasta","hay","donde","han","quien","están","estado","desd
   e","todo","nos","durante","estados","todos","uno",
46 "les","ni","contra","otros","fueron","ese","eso","había","
   ante","ellos","e","esto","mí","antes","algunos",
47 "qué","unos","yo","otro","otras","otra","él","tanto","esa
   ","estos","mucho","quienes","nada","muchos","cual",
48 "sea","poco","ella","estar","haber","estas","estaba","esta
   mos","algunas","algo","nosotros","mi","mis","tú",
```

dictionaries.php

```
65 "tuvieras","tuviéramos","tuvierais","tuvieran","tuviese","tu
   vieses","tuviésemos","tuvieseis","tuviesen","teniendo","teni
   do",
66 "tenida","tenidos","tenidas","tened"));
```

Apêndice B - Artigo Relacionado

Busca Semântica Aplicada a Informações Clínicas

Cloves Langendorf Barcellos Junior[1], Rafael Andrade[2],
Leonardo Andrade Ribeiro[3], Aldo von Wangenheim[4]

[1,4] Departamento de Informática e Estatística
[2,4] Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento
Universidade Federal de Santa Catarina (UFSC), Brasil

[3] Dept. of Computer Science - University of Kaiserslautern, Germany

Resumo - Este artigo apresenta um modelo de aplicação especializada em efetuar busca semântica em laudos médicos, utilizando bases de dados PostgreSQL indexadas com lexemas provenientes de dicionários *Ispell*. Isso resulta em laudos mais claros e com informações mais detalhadas, além de busca com resultados mais relevantes que os de uma pesquisa por palavras-chave.

Palavras-chave: Busca Semântica, Informática Médica, SNOMED CT, MeSH e DeCS.

Abstract - This article presents an application proposal specialized in semantic search in medical findings, using PostgreSQL databases indexed by Ispell lexemes. The result is a unified, sharable structured representation of medical findings integrating the representations of findings in SNOMED CT, MeSH and DeCS.

Key-words: Semantic Search, Medical Informatics, SNOMED CT, MeSH e DeCS.

Introdução

O desenvolvimento e a utilização de padrões para dados produzidos na área de saúde são guiados pela necessidade de representar de forma coerente e com precisão a comunicação da informação, garantir o armazenamento, a recuperação efetiva desses dados e principalmente, permitir a interoperabilidade entre diferentes sistemas de informação [1]. A padronização da linguagem utilizada é um grande desafio da informática em medicina, devido à complexidade inerente da linguagem e das informações envolvidas. Enquanto grandes esforços têm sido aplicados há décadas para o desenvolvimento de terminologias controladas (ontologias), a adaptação, a recuperação e a indexação ainda não podem ser utilizadas como única fonte de informação. Embora clinicamente ricos e praticamente globais, o poder expressivo destas ontologias pode ser uma limitação, uma vez que oferece mais do que uma forma de representar um determinado conceito [1].

Os métodos tradicionais de busca em ontologias médicas nem sempre permitem obter resultados isentos de conteúdos não pertinentes. O pior ocorre quando esta parcela de conteúdos indesejados torna-se grande a ponto de desqualificar o resultado obtido. Surge então a necessidade de desenvolvimento de novas metodologias utilizando as diversas tecnologias disponíveis, a fim de organizar, pesquisar, qualificar e cruzar as informações contidas nos mais variados acervos [2].

O número de ontologias utilizadas em aplicações na Internet vem crescendo constantemente; além disso, uma única ontologia nem sempre é suficiente para representar conceitos e tarefas de um ambiente distribuído como a Internet [3].

Atualmente existem mais de cem ontologias médicas que representam informações sobre diversos conceitos [2]. Representar e recuperar estas informações não pode ser considerada uma tarefa simples. Os dados precisam ser re-formatados e padronizados. Isto implica em considerável esforço computacional para padronização dos dados. Para o desenvolvimento desta metodologia serão utilizadas as ontologias SNOMED CT [4] e DeCS [5]. Ambas definem informações clínicas, porém seus conceitos nem sempre são literalmente idênticos. Isso impossibilita a integração e interação entre estas duas ontologias de forma direta.

O mapeamento e a integração de ontologias pertencentes a domínios semelhantes e a utilização de um método de busca semântica é fundamental quando queremos agregar mais conhecimento a um laudo, melhorando a compreensão das informações clínicas contidas nele. Isso também cria a possibilidade do médico executar consultas através de identificadores em todo banco de dados

por laudos semelhantes. Este processo contribui para que o laudo que está sendo redigido contenha informações mais coerentes e detalhadas.

As particularidades encontradas na redação de um laudo médico podem dificultar a compreensão por outro profissional das informações nele contidas. Ao selecionar sentenças (trechos) desse laudo e utilizá-las para inquirir sobre as ontologias, parte destas particularidades são isoladas, e o algoritmo se concentra somente nos termos reconhecidos pela base indexada, retornando resultados mais relevantes. Além disso, a interação entre os laudos se torna possível, já que estes agora possuem identificadores das ontologias.

O objetivo geral deste artigo é abordar as características do SNOMED CT e do DeCS, apresentando um modelo de aplicação especializada em efetuar busca semântica em laudos médicos, utilizando bases de dados PostgreSQL [6] indexadas com lexemas provenientes de dicionários *Ispell* [7].

Ontologias

Uma ontologia é uma especificação de um conceito, isto é, uma descrição de conceitos e relações que existem em um determinado domínio. Basicamente, uma ontologia consiste em conceitos e relações, e suas definições, propriedades e restrições, descritas na forma de axiomas [8]. Nesta aplicação são utilizadas as ontologias SNOMED CT e DeCS.

A ontologia SNOMED CT é uma das mais completas nomenclaturas multiaxiais criadas para indexar o conjunto de registros médicos. É composta por dezenove eixos hierárquicos e várias subclassificações. SNOMED CT é distribuída em três idiomas (alemão, inglês e espanhol). A categorização é feita de acordo com a classe semântica à qual pertence determinado conceito. Está dividida em conceitos, hierarquias, relacionamentos e descrições. Dentre as dezenove hierarquias e sub-hierarquias, ela possui quase 1,45 milhão de relacionamentos, os quais ligam os conceitos às hierarquias [9].

O DeCS (Descritores em Ciências da Saúde) foi criado a partir do MeSH [10] com o objetivo de permitir o uso de terminologia comum para pesquisa em três idiomas (português, inglês e espanhol), proporcionando um meio consistente e único para a recuperação da informação independentemente do idioma. Isso qualifica a aplicação a também inquirir sobre conceitos do MeSH. Utiliza um vocabulário estruturado permitindo a recuperação da informação com o termo exato utilizado para descrever o conteúdo daquele documento científico. É um vocabulário dinâmico com aproximadamente 29.000 descritores, sendo destes 23.963 do MeSH, 218 de Ciência e Saúde, 1.951 de Homeopatia, 3.486 de Saúde Pública e 828 de Vigilância Sanitária. O número é maior que o total, pois um descritor pode ocorrer mais de uma vez na hierarquia. Por ser dinâmico, registra o processo constante de crescimento e mutação.

Metodologia

Inicialmente, foram selecionadas entre diversas opções de ferramentas o *PostgreSQL* como SGBD, os dicionários *Ispell* nos idiomas português, inglês e espanhol, o *GiST (Generalized Search Tree)* [11], que é uma estrutura de indexação de grandes quantidades de dados utilizando árvores B, *GIN (Generalized Inverted Index)* [12] e a linguagem *PHP* [13] para o desenvolvimento da aplicação. Além destas ferramentas, foi necessário encontrar uma maneira de classificar os resultados obtidos através de uma métrica ou algoritmo. Para isso foi escolhido o algoritmo conhecido como *CDR (Cover Density Ranking)* [14] onde a proximidade, seqüência e número de ocorrências entre as palavras na sentença marcam mais pontos do que o método comum em buscas por palavras-chave e somente o maior número de ocorrências de palavras é levado em consideração [15].

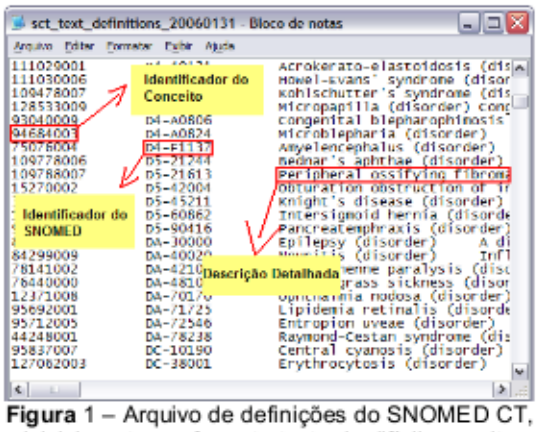

Figura 1 – Arquivo de definições do SNOMED CT, inicialmente em formato texto de difícil consulta

O próximo passo se deu no banco de dados, um ponto crítico da aplicação. Afinal, as duas ontologias não interagem entre si, apesar de tratarem de informações similares. Como exemplo, ao pesquisar por "Doença de Alzheimer", cada ontologia retorna:

SNOMED CT: Alzheimer's disease

(disorder) – código 26929004.
DeCS: Alzheimer Disease - código C10.228.140.380.100.

Os dois conceitos são equivalentes, porém não havia uma aplicação capaz de reconhecer essa equivalência e classificá-la. Além disso, os dados do SNOMED CT foram recebidos em grandes arquivos textos (figura 1), o que dificultava a sua consulta e entendimento.

O *Ispell* consiste em um conjunto de termos e lexemas que definem a maioria das palavras de um idioma. Além do *Ispell* um conjunto de termos sem grande relevância semântica como os termos "de", "do", "que", "e", entre outros são excluídos da sentença indexada a fim de melhorar a qualidade, já que estes termos tendem a deturpar os resultados finais.

Um *tsvector* é um campo que representa um documento (texto) reduzido aos seus lexemas. Para isso foi utilizada a combinação dos dicionários português, inglês e espanhol do *Ispell* e a ferramenta GIN. Nesta aplicação foi utilizado o GIN para indexar os campos do tipo *tsvector*. O GIN é um índice invertido, comumente utilizado em motores de busca de documentos de mesma natureza com um número variado de dados. Ele adiciona escalabilidade ao *Tsearch2*. Seu índice consiste em um par (chave, lista) onde a chave é a palavra (lexema) e a lista é uma lista dos documentos onde a chave ocorre. Normalmente a lista é armazenada em ordem, o que possibilita uma busca rápida e eficiente com várias palavras. A indexação executada com o GIN resulta em um campo do tipo *tsvector*, conforme exemplo:

'coxa':14 'baixa':9 'parte':7 'abdome':11 'região':1 'externa':3 'juncional':2 'localizada':4

Os lexemas são indexados seguindo o padrão *'lexema': 'posição na sentença'*.

Logo, a extensão *Tsearch2* é a principal ferramenta da aplicação. Ela utiliza os lexemas providos pelos dicionários *Ispell*, combinados ao GiST ou GIN para indexar os descritores dos conceitos de ambas as ontologias. Além disso, essa extensão provê uma sintaxe própria derivada da linguagem SQL. Esta sintaxe é utilizada para inquirir sobre os descritores indexados.

Um exemplo de uma consulta SQL executada pela aplicação:

```
SELECT * FROM decs_definition
WHERE definition @@
to_tsquery('(tratamento|câncer) &
tratamento de câncer') ORDER BY rank;
```

O *Tsearch2* também possibilita ao desenvolvedor escolher o algoritmo classificador, que nessa aplicação foi o CDR.

Outra etapa se refere ao algoritmo na linguagem PHP, que trata os dados retornados ao inquirir o banco de dados.

A aplicação foi desenvolvida no padrão MVC (*model-view-controller*), o que mantém o código bem definido e classificado. Para implementar o modelo foi utilizado PDO (PHP Data Object) [12], uma extensão leve e confiável que oferece uma camada abstrata de acesso a banco de dados.

Resultados

Como solução para o problema da interoperabilidade entre as ontologias, deu-se início ao desenvolvimento de um banco de dados integrado, onde tanto os conceitos do SNOMED CT quanto os do DeCS estão indexados. Para tal, foi de fundamental importância a escolha da ferramenta *PostgreSQL*, já que na versão 8.3 a extensão *Tsearch2 (Full Text Search Extension)* [16] foi integrada oficialmente ao SGBD.

O banco de dados resultante representado pela figura 2 contempla tanto os conceitos do SNOMED CT como os do DeCS.

Nesse banco de dados foram criados campos do tipo *tsvector*, indexados pelo Tsearch2 como, por exemplo: *name_index* (campo *fully_specified_name* da tabela *sct_concept* indexado), *term_index* (campo *term* da tabela *sct_definition* indexado), *definition_index* (campo *definition* da tabela *decs_definition*), *descriptor_index* (campo *descriptor* da tabela *decs_descriptor*), *synonym_index* (campo *synonym* da tabela *decs_synonym*), *annotation_index* (campo *indexing_annotation* da tabela *decs_indexing_annotation*) e *qualifier_index* (campo *qualifier* da tabela *decs_qualifier*).

Os campos indexados pelo Tsearch2 são de extrema importância para o funcionamento da metodologia proposta, já que eles guardam o posicionamento exato de cada lexema em uma determinada sentença.

Durante a implementação dos métodos de buscas na linguagem PHP, um segundo ponto crítico foi identificado: o SNOMED CT só estava disponível no idioma inglês. Sabendo que a maioria das buscas será efetuada em laudos médicos no idioma português (já que a aplicação está sendo testada em ambiente de língua portuguesa), foi constatado um problema com a identificação da busca em português em uma ontologia de língua inglesa. A solução para amenizar esta dificuldade veio através do DeCS, já que ele possui em sua

maioria os conceitos nos três idiomas. Logo, além de efetuar uma busca semântica de uma sentença em português no DeCS, é necessário efetuar uma segunda busca semântica entre o correspondente no idioma inglês no DeCS e os conceitos em inglês do SNOMED CT, retornando os conceitos do SNOMED CT mais próximos da sentença original em português.

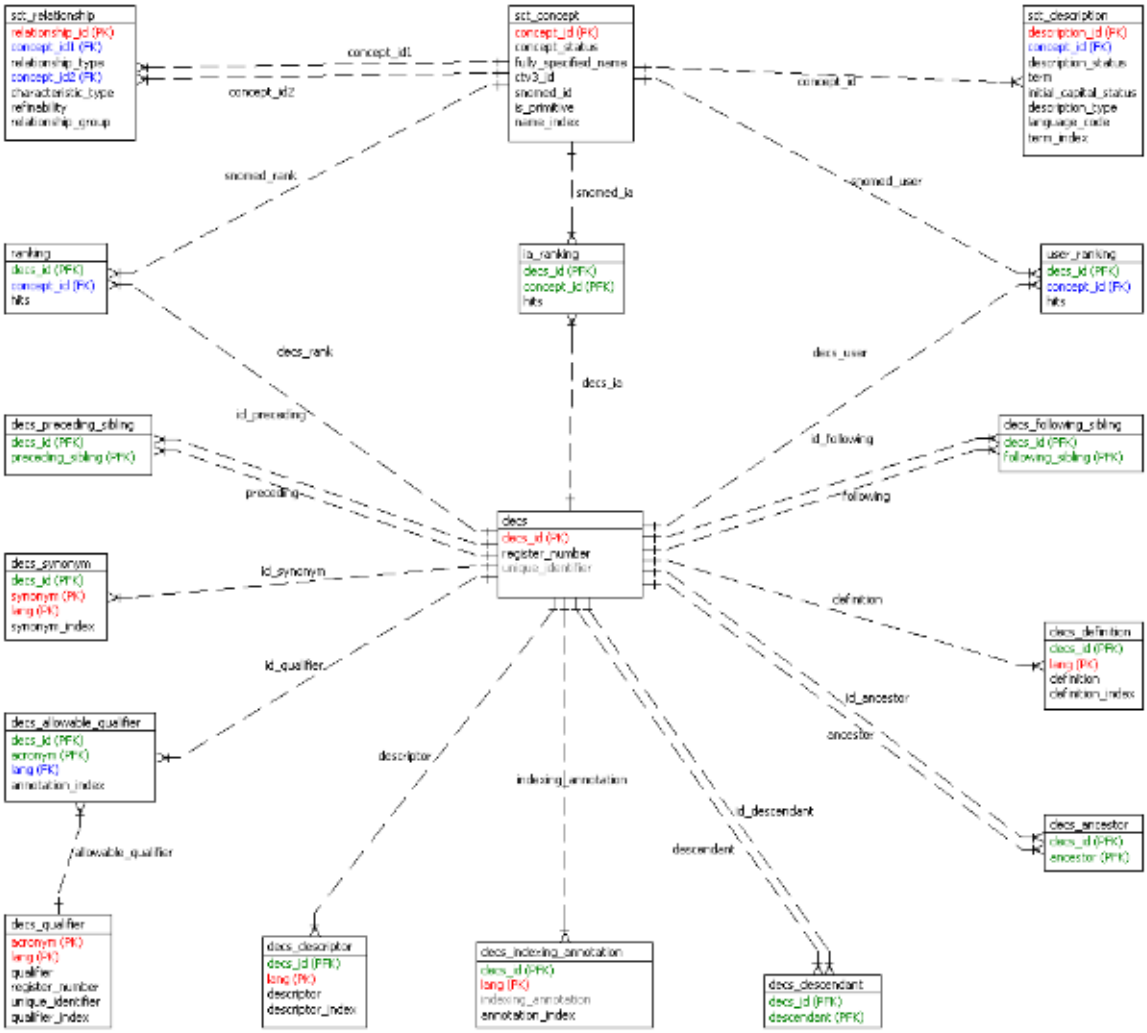

Figura 2 – Banco de dados resultante com campos indexados pelo Tsearch2

Em uma busca pelo código do termo "*Alzheimer*" na ontologia DeCS, resulta em um campo indexado através de sua definição, conforme descrito no exemplo:

> Código: *C10.228.140.380.100*
> Termo: *Doença de Alzheimer*
> Definição: *Doença degenerativa do CÉREBRO caracterizada pelo início traiçoeiro de DEMÊNCIA. Falhas da MEMÓRIA, no julgamento, no momento da atenção e na habilidade em resolver problemas são seguidos de APRAXIAS severas e perda global das habilidades cognitivas. A condição ocorre principalmente após os 60 anos de idade e é marcada patologicamente pela atrofia cortical severa e tríade de PLACAS SENIS, EMARANHADOS NEUROFIBRILARES e FILAMENTOS DO NEURÓPILO.*

Campo indexado em português:
'doença':1 'alzheim':3

Campo indexado em inglês:
'alzheim':1 'disease':2

Através do algoritmo CDR combinado com a comparação dos campos indexados, a aplicação desenvolvida é capaz de identificar o conceito do SNOMED CT mais relevante ao resultado obtido na busca por um termo DeCS:

Código: *26929004*
Termo: *Alzheimer's disease*.
Definição: *Alzheimer's disease (disorder)*.

Campo indexado em inglês:
"'alzheim':1 'disease':3 'disorder':4"

A aplicação resultante possui um conjunto variado de funcionalidades, dentre as quais podem ser citadas: criação de sentenças do tipo *tsquery* sobre um campo *tsvector*, submissão de sentenças *tsquery* com variado número de palavras, submissão da mesma sentença em três idiomas (português, inglês e espanhol), composição e exibição hierárquica dos conceitos das ontologias, escolha de tipo de classificação (ranking) e exibição dos resultados.

Discussão e Conclusões

Partir inicialmente de duas ontologias que não interagiam entre si e que foram recebidas em formato texto como o SNOMED CT e formato XML como o DeCS foi um desafio, que resultou em um banco de dados capaz de armazenar todas as suas informações clínicas de ambas as ontologias no formato *tsvector*.

A possibilidade de obter resultados mais relevantes, com menor discrepância e isentos de conteúdos não pertinentes através da integração dessas ontologias aumenta consideravelmente a agregação de conhecimento em laudos médicos, possibilitando diagnósticos mais completos e precisos.

Foram realizados testes utilizando laudos médicos publicados em ambiente de produção. Pode-se comprovar que essa metodologia aumenta consideravelmente a qualidade dos resultados pesquisados. Em um destes testes a busca utilizando métodos tradicionais por palavras-chave (busca sintática) retornou aproximadamente 40 resultados, pouco relevantes. Já ao aplicar a metodologia proposta, esse número foi reduzido a 8 resultados com um grande grau de relevância.

A universalidade das ontologias estudadas possibilita que médicos e pacientes de diferentes nacionalidades possam interagir e compreender as informações clínicas contidas em laudos redigidos em diferentes idiomas. A metodologia utilizada permite ainda cruzar informações de um laudo previamente indexado pelo MeSH ou DeCS, com o SNOMED CT. Neste caso, ao indexar um laudo na ontologia DeCS pode-se facilmente visualizar o conceito equivalente no SNOMED CT.

Uma discussão futura será a aplicação dessa metodologia, para auxiliar o preenchimento de laudos estruturados utilizando DICOM SR [17]. Atualmente existe uma forte resistência à adoção de um padrão de laudo estruturado por parte da comunidade médica em virtude do grande número de campos a serem preenchidos. Uma abordagem utilizando a metodologia desenvolvida pode facilitar o preenchimento destes laudos através do algoritmo de preenchimento semântico. Ou ainda o modelo pode transcrever um laudo não estruturado para o padrão DICOM SR, onde os termos são identificados e vinculados a um determinado tipo de campo.

Esta metodologia também pode ser aplicada em outros domínios com grande volume de informações, para obter resultados mais relevantes em suas buscas.

Referências

[1] Richesson RL, Krischer J. Data standards in clinical research: gaps overlaps, and challenges and future directions. Journal of the American Medical Informatics Association. 2007;14:687–696. DOI 10.1197/jamia.M2470.

[2] J. J. Cimino, H. Min, Y. Perl. Consistency across the hierarchies of the UMLS Semantic Network and Metathesaurus. Journal of Biomedical Informatics, Volume 36, Issue 6, December 2003, Pages 450-461.

[3] Y.Kalfoglou, M.Schorlemmer, (2003). "Ontology mapping: the state of the art". The Knowledge Engineering Review 18(1):1--31, January 2003.

[4] SNOMED Systematized Nomenclature of Medicine (2006). "SNOMED International, a division of the College of American Pathologists (CAP)". Disponível em: http://www.ihtsdo.org/. Acessado em 25 de julho de 2008.

[5] DeCS - Descritores em Ciências da Saúde. Disponível em: http://decs.bvs.br/. Acessado em 25 de julho de 2008.

[6] PostgreSQL Global Development Group. Disponível em: http://www.postgresql.org/.

Acessado em 25 de julho de 2008.

[7] Ispell - spelling checker. Disponível em: http://ficus-www.cs.ucla.edu/geoff/ispell.html. Acessado em 23 de julho de 2008.

[8] Berners-Lee, T., Hendler, J., Lassila, O. (2001). "The Semantic Web". Scientific American, Vol. 5/2001. Disponível em: http://www.sciam.com/. Acessado em 25 de julho de 2008.

[9] James E. Andrews, Timothy B. Patrick, Rachel L. Richesson, Hana Brown, Jeffrey P. Krischer. Comparing heterogeneous SNOMED CT coding of clinical research concepts by examining normalized expressions. Journal of Biomedical Informatics, In Press, Corrected Proof, Available online 5 February 2008.

[10] MeSH - Medical Subject Headings. Disponível em: http://www.nlm.nih.gov/mesh/. Acessado em 25 de julho de 2008.

[11] The GiST Indexing Project. Disponível em: http://gist.cs.berkeley.edu/. Acessado em 23 de julho de 2008.

[12] GIN - Generalized Inverted Index. Disponível em: http://gist.cs.berkeley.edu/. Acessado em 23 de julho de 2008.

[13] PHP - Scripting language. Disponível em: http://www.php.net/. Acessado em 25 de julho de 2008.

[14] Cover Density Ranking (CDR). Disponível em: http://www2002.org/CDROM/refereed/643/node7.html. Acessado em 25 de julho de 2008.

[15] Clarke, Charles L. A.; Cormack, Gordon V.; Tudhope, Elizabeth A. (2000) " Relevance Ranking for One to Three Term Queries". Information Processing & Management, v36 n2 p291-311 Mar 2000.

[16] Tsearch2 Extension. Disponível em: http://www.postgresql.org/docs/current/static/tsearch2.html. Acessado em 25 de julho de 2008.

[17] DICOM Structured Reporting. Disponível em: http://medical.nema.org/. Acessado em 25 de julho de 2008.

Contato

Cloves Langendorf Barcellos Junior (cloves@inf.ufsc.br) – Universidade Federal de Santa Catarina – Campus Universitário, s/n - Florianópolis/SC – Brasil - Tel.: (48) 3721-9166 - http://www.telemedicina.ufsc.br