Kirill Lassounski

BioSearch Refinement: Um sistema para facilitar a busca de artigos relevantes no PubMed

Orientadora: Sahudy Montenegro González

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Campos dos Goytacazes/RJ

2011

Sumário

Resumo

A quantidade de informação na *Internet* está em constante crescimento, o que demanda técnicas modernas de recuperação de informação para organizá-la. O NCBI é um concentrador de informação biomédica e possui em sua estrutura o banco de dados PubMed que contém artigos científicos de jornais e revistas e outros recursos. O objetivo do *BioSearch Refinement* é ajudar o pesquisador a encontrar informação relevante com mais facilidade e rapidez no banco de dados PubMed. O sistema permite a sumarização dos artigos através da representação de seus principais conceitos utilizando palavras-chave, e estas palavras podem ser usadas para refinar o conjunto de artigos exibidos. Neste trabalho também é apresentado o PubMed Dataset que provê o acesso ao NCBI e permite a criação de conjuntos de dados para teste com os dados do PubMed.

Abstract

The amount of information on the Internet is at constant growth and it demands modern information retrieval techniques to organize it. The NCBI is a biomedic information center, having in its structure the PubMed database that contains scientific articles and other resources. The BioSearch Refinement goal is to help researchers to find relevant information on PubMed easily and quickly. The system provides article summarization through the representation of its basic concepts using keywords, these keywords can be used to refine the set of articles displayed on the interface. This paper also presents the PubMed Dataset that provides access to NCBI and allows the creation of data sets using PubMed articles to test IR systems.

1 Introdução

A capacidade de visualizar as informações de maneira clara e organizada, é importante para qualquer usuário. Geralmente, as informações com as quais se lida no dia-a-dia são textuais e necessitam de tempo para serem interpretadas. Quando a quantidade destes dados é grande, se leva um tempo considerável para encontrar a informação de relevância. Nas décadas de 50, a informação começou a migrar do papel para os dispositivos eletrônicos, onde pode ser mais facilmente armazenada, conservada e organizada. O crescimento da quantidade de documentos armazenados tornou difícil a recuperação daqueles que são de interesse em um dado momento. A recuperação de informação é uma área da ciência da computação que lida com a armazenamento, organização, representação e acesso à itens de informação (BAEZA-YATES; RIBEIRO-NETO, 1999).

O tráfego na Internet tem crescido numa escala de 100% a cada ano a partir da década de 90. Atualmente esta taxa diminuiu e está em 50% (ODLYZKO, 2003). A previsão é de que esta taxa diminua para 35% no período até 2013 (CISCO, 2009). Apesar da diminuição da taxa de crescimento do tráfego na Internet, a quantidade de informações sendo lidas e armazenadas na grande rede cresce diariamente e a capacidade de extrair conhecimento relevante se torna uma dificuldade. Muitas vezes a informação está acessível, mas os métodos de busca atuais tornam difícil o seu acesso.

A crescente massa de informação criou a necessidade da concentração desta em um lugar em comum onde a informação possa ser visualizada, organizada e classificada. Existem diversos bancos de dados e bibliotecas que disponibilizam informação, seja ela científica ou tecnológica. Como exemplos pode-se destacar:

- PubMed[1]: possui mais de 21 milhões de citações de literatura biomédica do MEDLINE, periódicos científicos e e-books.
- Science Direct[2]: banco de dados do SciVerse com mais de 10 milhões de artigos

[1] http://www.ncbi.nlm.nih.gov/pubmed
[2] http://www.sciencedirect.com/

científicos e capítulos de livros.

- CiteULike[3]: serviço gratuito para encontrar e gerenciar referências acadêmicas, possui atualmente cerca de 5 milhões de documentos.
- IEEE Xplore[4]: possuindo 3 milhões de documentos, o portal da IEEE provê acesso à informação técnica em engenharia e tecnologia.
- ACM Digital Library[5]: coleção de artigos e registros bibliográficos sobre os campos de computação e tecnologia da informação.

Independente do mecanismo de busca, se a base de dados for muito grande, o que acontece em muitos casos, serão retornadas grandes quantidades de documentos que tornam tedioso o processo de busca no conjunto de resultados. Quando esta quantidade de resultados ultrapassa centenas de itens, o processo de busca se torna cansativo mesmo fazendo uma leitura dinâmica nos resultados. Os sistemas de recomendação sugerem documentos relevantes baseado em informações providas por outros usuários de um sistema, gerando uma melhora no processo de busca (JANNACH; ZANKER; FELFERNIG, 2010), mas este tipo de algoritmo não reflete na quantidade de documentos retornados. O PubMed utiliza um sistema de recomendação no campo "*Related searches*", que disponibiliza frases de busca relacionadas com a submetida pelo usuário.

Para reduzir a quantidade de documentos disponibilizados em uma busca, pode-se dividi-los em categorias rotuladas por palavras-chave. Cada palavra-chave representa uma porção do total de documentos exibidos e representa um tema ou assunto abordado pelo documentos do sub-cojunto. A extração de palavras-chave é uma técnica que obtém termos descritores a partir dos textos de documentos, que podem ser usados para indexar/recuperar ou sumarizar os dados documentos.

Neste trabalho é apresentada uma abordagem de extração de palavras-chave a partir de documentos, e além disso é apresentada uma maneira de utilizar estas palavras para resumir sub-conjuntos destes documentos. Para avaliar os resultados obtidos foi criada uma biblioteca de construção de conjuntos de dados de teste obtidas do PubMed.

[3]http://www.citeulike.org/
[4]http://ieeexplore.ieee.org/Xplore/guesthome.jsp
[5]http://dl.acm.org/

1.1 Formulação do problema

A grande massa de informação disponível criou a necessidade de mecanismos que possam organizá-la de maneira clara para os usuários. Esta dificuldade não é restrita à Internet, mas também existe em grandes empresas e instituições. Como uma grande parte desta informação está na forma de texto, muito esforço tem sido feito nas comunidades de recuperação de informação para melhorar os algoritmos de indexação/ordenação de documentos.

Como um exemplo pode-se destacar o *National Center for Biotechnology Information* (NCBI), que provê informações científicas sobre biomedicina e genomas. É um grande portal mantido pelo governo dos Estados Unidos da América utilizado por pesquisadores de todas partes do mundo servindo informações em seus estudos. Um de seus bancos de dados mais acessado é o PubMed, que guarda cerca de 21 milhões de artigos científicos na área de saúde, um número que cresce continuamente. Os resultados exibidos pelo PubMed formam um lista linear de artigos científicos que não possibilitam o usuário de ter um panorama dos tópicos sendo abordados por estes artigos ou de refinar os artigos até atingir uma necessidade em particular. Esta grande quantidade de informação gera dificuldade no processo de localização do material desejado com os métodos de busca tradicionais.

O processo de busca no portal se utiliza do modelo booleano[6] onde o usuário especifica palavras-chave e as conecta utilizando operadores booleanos (*AND,OR* e *NOT*). As palavras-chave podem ser buscadas em diversos atributos de um documento como autor, ano de publicação, termos MeSH, resumo, título entre outros. Neste tipo de busca, os documentos retornados são filtrados de acordo com os termos que estão contidos nele e os que não estão contidos. Este tipo de busca é conhecido por retornar grandes conjuntos de resultados, além de possuir uma complicada lógica de formulação das buscas, logo um usuário inexperiente muitas vezes tem dificuldade em efetuar buscas satisfatórias (JACKSON; MOULINIER, 2007).

Neste cenário, a quantidade de artigos retornados em uma busca geralmente é muito extensa e para diminuir o número de resultados o usuário se vê obrigado a inserir mais termos para refinar sua busca. Para exemplificar o problema, ao se fazer uma busca no PubMed com um termo de busca bem geral como *"tuberculosis"*, foram retornados 192.499 artigos. Ao se especificar este termo para *"human [Title/Abstract] AND mycobacterium*

[6] http://www.nlm.nih.gov/pubs/techbull/ja97/ja97_pubmed.html

tuberculosis [Mesh]" a quantidade de resultados diminuiu para 3.084, mas ainda assim permanece grande para uma leitura rápida e o pesquisador se vê obrigado a especificar mais a sua busca. Como observado por (PEREZ-IRATXETA; BORK; ANDRADE, 2001), em uma busca exploratória, onde o pesquisador seleciona termos de seu conhecimento para melhorar os resultados da busca, muitas vezes não se tem conhecimento de um determinado termo até que este seja identificado no texto.

1.2 Objetivos

Como objetivo principal, se propõe um método de refinamento de artigos do banco de dados PubMed através da utilização de palavras-chave. Cada palavra-chave representa um tema que é amplamente abordado por artigos retornados de uma busca ao banco. O método proposto permitirá o(a) pesquisador(a) encontrar com mais facilidade e rapidez artigos relevantes no PubMed. O objetivo secundário é a construção de uma *interface* que melhore a navegação e visualização dos artigos.

Para atingir tal objetivo foi desenvolvido o sistema BioSearch Refinement, uma aplicação que pode sumarizar e dar uma visão geral dos artigos retornados em uma busca. Para destacar os temas sendo mais abordados em uma determinada busca são utilizadas palavras-chave. As palavras-chave podem ajudar o leitor a decidir se um determinado documento é de seu interesse ou não. Elas dão uma breve descrição sobre o conteúdo mais importante sendo abordado, mas muitas vezes os documentos não possuem palavras-chave e o processo de atribuição é muito custoso (LUI, 2007). Além da sumarização, palavras-chave podem ser usadas para indexar documentos e também para tornar uma busca mais precisa (TURNEY, 1999). Neste trabalho usamos as palavras-chave para fazer sumarização de grupos de documentos, onde cada palavra-chave representa uma parte deste grupo. Com estas palavras pode-se reduzir a quantidade de resultados retornados facilitando o processo de busca.

A ideia central do algoritmo é extrair conceitos a partir dos resumos dos artigos, que serão utilizados para determinar quais artigos são referenciados por um determinado conceito. Os conceitos que referenciarem um maior número de artigos serão considerados mais generalistas e serão escolhidos como palavras-chave que sumarizam os principais assuntos de uma busca. A extração se utiliza de técnicas de processamento de linguagem natural e frequência de termos para determinar os conceitos. O algoritmo é aplicado no sistema BioSearch Refinement que é um sistema com o propósito de simplificar e ajudar

o pesquisador a encontrar material relevante.

1.3 Organização

No capítulo 2 são apresentados os trabalhos relacionados e as metodologias usadas para a sumarização de documentos e extração de palavras-chave. Também são introduzidos os conceitos da recuperação de informação necessários para compreender o capítulo 3. No capítulo 3 é apresentado o sistema *BioSearch Refinement* e as partes que o compõem, o *Extraction Engine* e o *PubMed Dataset*. O capítulo 4 apresenta os resultados obtidos para as partes que compõem o sistema e também discute estes resultados. No capítulo 5 são tiradas as conclusões sobre o trabalho feito como também são apresentadas as contribuições e possíveis trabalhos futuros.

2 Recuperação de informação

Um sistema que recupera informação permite um usuário manusear a informação contida em documentos, que na maioria das vezes estão em formato de texto. O objetivo do sistema é ajudar a encontrar a informação indicando a sua localização, ou informar se ela de fato existe no domínio buscado. O resultado da recuperação é um conjunto de documentos, dos quais alguns são uteis para o usuário (relevantes) e outros não (irrelevantes). Basicamente um sistema de recuperação possui: uma forma de representar os documentos, uma forma de representar a necessidade do usuário e a comparação entre as duas representações (GöKER; DAVIES, 2009).

As palavras-chave são unidades textuais que servem para representar documentos, elas possuem a essência do documento, descrevendo de maneira clara e precisa o seu conteúdo. Neste trabalho palavras-chave representam tanto termos simples *"influenza"* como termos compostos (*keyphrases*) *"influenza h1n1 virus"*. Os termos compostos são mais valorizados, pois são considerados como sendo mais informativos para descrever um documento (FAGAN, 1989).

Para determinar os temas principais sendo abordados em um conjunto de documentos, é preciso identificar cada documento com um conjunto de um ou mais descritores que reflitam os temas principais sendo abordados, tarefa da indexação automática.

2.1 Indexação automática

A indexação automática associa termos descritivos aos documentos de um conjunto através do programa de computador, no intuito de permitir a eficiente recuperação e armazenamento destes em um sistema informatizado (GONZALEZ; LIMA; LIMA, 2006). O início da indexação automática se deu nos anos 50, naquela época a quantidade de informação guardada em computadores era limitada, logo não eram necessárias técnicas muito avançadas para organizá-la. Basicamente cada documento era indexado manual-

mente com alguns termos que o descreviam, e quando o usuário fazia um busca, os termos da busca eram comparados com os termos do documento e exibidos aqueles documentos que satisfaziam a igualdade de termos de busca para termos descritores. Mas a rotulação manual de cada documento exige um profissional especializado com conhecimento do domínio deste documento e tempo para identificar os descritores e provavelmente a quantidade de pessoal especializado em indexação de documentos não é suficiente para indexar com qualidade a crescente massa de documentos que vem surgindo.

Uma ótima análise do processo de obtenção de descritores foi feita por Gonzalez et al. (GONZALEZ; LIMA; LIMA, 2006) e a seguir são abordados os seus principais pontos. Nem todos os termos existentes em um texto podem ser considerados descritivos, por isso existem diversas técnicas e abordagens para a obtenção de descritores para um documento. Mas quase todas as abordagens possuem uma fase em comum, a fase de pré processamento do texto. Nesta fase, o texto é preparado para a obtenção dos descritores, e suas operações estão de acordo com a abordagem sendo adotada.

Geralmente o texto passa pela tokenização (identificação dos itens léxicos do texto), seleção dos descritores e conflação (BAEZA-YATES; RIBEIRO-NETO, 1999). Se a abordagem utilizada se utiliza de informações linguísticas, técnicas de processamento de linguagem natural podem ser usadas para determinar as relações morfo-sintáticas entre as palavras. Estas relações são obtidas através do *part-of-speech tagging*, explicado adiante.

Na fase de seleção de descritores ocorre a eliminação de *stopwords*, que elimina palavras pouco representativas como preposições, artigos e conjunções. A eliminação destas palavras diminui a quantidade de possíveis descritores ganhando economia, mas por outro lado diminui a representatividade. Uma maneira de aproveitar estas palavras é apenas permiti-las como ligação entre palavras relevantes na análise morfo-sintática, gerando assim descritores mais representativos. Vale ressaltar que em alguns domínios existem palavras que podem ser consideradas *stopwords*, mas em outros domínio não são. Um exemplo disso são palavras como remédio, doença, paciente, resultado e análise no domínio da medicina. Palavras assim tem pouco valor de indexação e recuperação, pois a maior parte dos documentos possui estes termos.

Uma técnica muito utilizada para a representação de termos similares como sendo um só é a normalização, onde palavras com variações linguísticas são interpretadas como sendo a mesma palavra ou conceito. Utilizando a normalização, o cálculo de frequência pode ser ampliado para um determinado termo através da similaridade com outros termos do texto. A forma de normalização que é usada neste trabalho, é a conflação. Segundo

(GONZALEZ; LIMA; LIMA, 2006) as formas de conflação mais usadas é o *stemming* e a lematização. Um algoritmo de *stemming* ou em português, radicalização, trunca as palavras para deixar apenas o radical desta palavra, permitindo assim remover plurais e sufixos que causam a variação de uma mesma palavra. Já a lematização reduz a palavra até sua forma canônica: *lemma*(working) = work ; *lemma*(worked) = work. A lematização possui resultados melhores, porém é mais difícil de ser aplicada, já que em alguns casos é necessário o uso de um dicionário para poder determinar o lema corretamente: *lemma*(better) = good.

Como mostrado no livro de (MOENS, 2000), na indexação automática a seleção de descritores pode ser subdividida em duas abordagens. Na primeira, os termos são extraídos diretamente do texto através de um processo de filtragem afim de deixar apenas os termos mais descritivos. Na segunda abordagem, os termos são selecionados de um vocabulário controlado (*thesaurus*), ou seja, atribuídos aos documentos. Um *thesaurus* difere de um dicionário pelo fato de possuir relações entre palavras e sinônimos ao invés de definições, fornecendo assim agrupamentos de palavras com mesmo significado.

2.1.1 Extração de palavras-chave

A extração de termos não se baseia no uso de um vocabulário controlado para a seleção das palavras-chave para um documento, em vez disso, utiliza termos do próprio texto. Pode trabalhar usando abordagens léxicas, estatísticas ou híbridas para extrair descritores do texto de um documento.

A abordagem léxica segue uma sequencia de passos pré definidos para a extração dos descritores de um texto:

1. Análise léxica: nesta fase são determinadas as classes gramaticais de cada palavra de um frase, com estas informações pode-se decidir quais palavras podem ser importantes ou não. Um *part-of-speech tagger* (POST) serve para fazer uma rotulação das palavras que compõem o texto, onde cada rótulo representa a classe gramatical de uma palavra.

2. Análise sintática: esta análise serve para gerar descritores compostos por mais de uma palavra, os descritores são criados baseado em padrões sintáticos definidos previamente. Uma máquina de estado é alimentada com os rótulos e trunca a frase em termos candidatos(*chunking*).Geralmente os padrões sintáticos são montados para selecionar palavras-chave compostas por substantivos (*noun chunking*), pois o

> substantivo é a classe gramatical que nomeia objetos, pessoas, substâncias, lugares e serve para identificar sobre o que se está discursando (GUCKER, 1966).

De todos os termos resultantes do processo de *chunking* são selecionados os melhores descritores usando-se de alguma metodologia específica. Geralmente são calculados pesos para cada termo candidato e são selecionados aqueles que possuem maior peso.

As técnicas estatísticas fazem cálculos para determinar a relevância de um termo em um texto e então decidir quais destes termos identificarão o documento. Alguns sistemas trabalham as relações de termos no conjunto de documentos total (*corpus*) e outros se utilizam apenas do domínio de um documento, mas o que todos estes sistemas tem em comum é a utilização de *corpus* de treinamento para criar modelos de predição que determinam se um termo é ou não relevante. (BERRY; KOGAN, 2010) afirmam que os sistemas que se utilizam de métodos orientados a *corpus* geralmente trabalham com unigramas (palavra-chave de apenas uma palavra) o que limita a qualidade dos descritores por serem usados em múltiplos e diferentes contextos.

Em uma abordagem híbrida são usadas ambas as técnicas citadas anteriormente buscando aproveitar as melhores qualidades de cada uma para se determinar descritores com uma maior qualidade. Neste trabalho utilizamos esta abordagem, primeiramente determinamos os possíveis descritores de um documento através da obtenção das classes gramaticais das palavras do texto utilizando o POST e em seguida extraímos *noun chunks* utilizando padrões sintáticos. Para determinar quais destes termos são relevantes é utilizado um novo algoritmo que descobre a relação entre os termos candidatos fazendo um cruzamento de informações e determina as palavras-chave através de um peso que envolve a frequência de termos e o número de palavras que compõem uma palavra-chave.

Part of speech tagger (rotulador)

Para o processamento de linguagem natural utiliza-se a API LingPipe[1] da Alias-i sob a licença *Alias-i ROYALTY FREE LICENSE VERSION 1.* Uma API (*Application Programming Interface*), também chamada de biblioteca, é um conjunto de códigos de uma determinada linguagem de programação que permitem ao utilizador desta, acelerar o seu processo de desenvolvimento através da reutilização destes códigos. Neste caso, a funcionalidade essencial é o rotulamento, que é implementado pela API utilizando HMM (*Hidden Markov Model*, uma máquina de estados finitos. Foi mostrado que as cadeias

[1] http://alias-i.com/lingpipe/

de Markov são muito eficientes na extração de informação quando se trata de informação sensível ao contexto como texto em linguagem natural oferecendo equilíbrio entre simplicidade e expressividade de contexto(FREITAG; MCCALLUM, 2000). As cadeias de Markov determinam a probabilidade da classe gramatical de uma palavra baseado na classe gramatical de outras palavras da frase.

Como todo modelo estatístico, o rotulador necessita de um *corpus* para ser treinado e neste caso é usado o MEDLINE corpus já que o domínio de linguagem é biomédico. Rotuladores desenvolvidos para textos gerais não funcionam bem quando aplicados ao MEDLINE. O rotulador de Brill (BRILL, 1992) aplicado a 1000 frases selecionadas aleatoriamente do MEDLINE, obteve uma exatidão de 86.8% usando o conjunto de rótulos de *Penn Treebank*. Esta baixa performance esta relacionada à especificidade do vocabulário do MEDLINE. O rotulador MedPost (SMITH; RINDFLESH; WILBUR, 2004) foi desenvolvido para satisfazer a necessidade de uma alta precisão no rotulamento treinado sobre o MEDLINE corpus. No teste de 1000 frases, ele obtêm exatidão de 97.43% usando o conjunto de rótulos nativo.

2.1.2 Atribuição de palavras-chave

O método de atribuição de termos seleciona os descritores de um vocabulário controlado que melhor descrevem um documento. Uma maneira muito utilizada atualmente para a representação de domínios se chama ontologia, onde palavras-chave são utilizadas para representar conceitos.

Ontologias

Segundo Gruber (GRUBER, 1993) uma ontologia é a especificação de um vocabulário para uma determinado domínio, definição de classe, relações, funções e outros objetos. Ela descreve formalmente um domínio e é composta por termos e o relacionamento entre estes (ANTONIOU; HARMELEN, 2004). As ontologias são muito utilizadas nos campos de biologia e medicina, pois estas possuem uma grande quantidade de termos e relações entre eles. Tem sido feito muito esforço nas comunidades acadêmicas de vários países na pesquisa e desenvolvimento desta tecnologia. As principais ontologias na área de biomedicina atualmente são a *Gene Ontology* (GO) (ASHBURNER et al., 2000) que faz um mapeamento semântico de genes e produtos genéticos como proteínas, *Medical Subject Headings* (MeSH) (LIPSCOMB, 2000) utilizado pelo NCBI na indexação de documentos científicos e a *Foundational Model of Anatomy* (FMA) (ROSSE; MEJINO, 2003) que é

uma ontologia para representação de anatomia humana.

GoPubMed

Uma ferramenta que utiliza uma abordagem baseada em ontologias para organização dos resultados obtidos do PubMed é o GoPubMed[2] (DOMS; SCHROEDER, 2005), que utiliza o Gene Ontology e o MeSH. Este sistema divide as informações em quatro categorias principais (o que, quem, onde e quando) através das informações obtidas do PubMed e permite o usuário restringir sua busca utilizando estes campos.

1. A categoria "o que", é processada utilizando-se dos resumos retornados a partir de uma busca à base de dados do PubMed e através de ontologias o sistema cria um árvore onde cada nível representa um tópico sendo abordado pelos artigos da pesquisa. O uso de ontologias permite a descrição de termos que são identificados na ontologia, descrevendo sinônimos e mostrando uma árvore de relacionamentos.

2. Na categoria "quem", os artigos podem ser filtrados por autores e centros. Nesta categoria podem ser visualizados os dados das instituições e cientistas. O sistema leva em conta que um determinado autor publica sobre temas similares, com os mesmo co-autores e nas mesmas revistas, estas associações são mapeadas em uma rede semântica que determina a probabilidade de dois artigos serem escritos pela mesma pessoa.

3. Em "onde", estão disponibilizadas as localizações de pessoas, universidades e centros de pesquisa, além disso são listadas também as melhores revistas (com alto fator de impacto).

4. A categoria "quando", mostra os anos das publicações e permite selecionar a data de publicação para reduzir o número de artigos sendo exibidos.

Neste trabalho não optamos pela utilização de ontologias, pois o vocabulário destas é controlado e deve ser constantemente atualizado. Alguns artigos não possuem termos MeSH por serem inovadores ou antigos. As palavras-chave retiradas diretamente do resumo e título são mais representativas e mais descritivas para um determinado artigo do que termos predefinidos.

[2] http://www.gopubmed.com

2.2 Sistemas de refinamento

A seguir são apresentados alguns trabalhos de sistemas que permitem a categorização de artigos retornados a partir de uma busca do PubMed. Como se pode perceber existem diversas abordagens para o agrupamento de artigos similares, mas poucos dos trabalhos pesquisados fazem a utilização de um algoritmo próprio de extração de palavras-chave para classificar documentos. (MARON; KUHNS, 1958) tiveram a ideia de organizar os documentos baseando-se em uma probabilidade de relevância. Os termos de busca inseridos pelo usuário eram processados pelo teorema de Bayes e era determinada a probabilidade de cada documento ser relevante para o usuário. A vantagem desta abordagem era eliminar a busca booleana que é ineficiente quanto à seleção de documentos relevantes.

2.2.1 XplorMed

A ferramenta XplorMed de (PEREZ-IRATXETA; BORK; ANDRADE, 2001) é uma das primeiras a propor uma maneira de melhorar a busca por artigos relevantes no PubMed e se utiliza de cálculos aritméticos simples para criar relacionamentos entre palavras nos resumos e, assim, identificar palavras-chave. Neste sistema, a relação entre palavras é criada utilizando lógica fuzzy onde há um relacionamento entre duas palavras x e y medido por um valor entre 0 e 1. O valor é medido pela seguinte equação:

núm. de resumos que contém x e y / núm. total de resumos que contém x ou y

O grau de inclusão mostra que palavras mais gerais incluem palavras mais especificas, a inclusão de x em y é:

núm. de resumos que contém x e y / núm. total de resumos que contém x

Palavras importantes são aquelas que possuem um alto valor associativo, valor que é calculado pelo somatório do grau de inclusão de todas as palavras que são relacionadas à palavra em questão. O problema desta abordagem é que ela restringe o relacionamento entre palavras a apenas duas palavras. O usuário do sistema pode restringir sua busca navegando pelos relacionamentos das palavras geradas e assim reduzir o número de artigos sendo exibidos. Também, é possível visualizar as frases que possuem uma determinada relação. O processo é iterativo e pode gerar uma cadeia de palavras, ou seja, a cada redução do conjunto de resultados é feita uma nova análise do relacionamento entre as palavras.

Para avaliar a capacidade de melhorar a precisão de resultados efetuados no PubMed, foram selecionadas 4 revisões. Foram selecionadas cuidadosamente palavras-chave que representassem a revisão e foi feita uma busca no PubMed, a precisão foi calculada com os artigos retornados pelo PubMed e os artigos contidos na revisão sendo analisada. Utilizando uma cadeia de palavras-chaves geradas pelo XplorMed se chegou à uma precisão mais elevada (0.034 -> 0.092 em um dos artigos). A precisão é a divisão do número de documentos relevantes pelo número total de documentos retornados.

2.2.2 Anne O'Tate

Um trabalho similar com o que está sendo proposto é o Anne O'Tate, publicado por (SMALHEISER; ZHOU; TORVIK, 2008). Neste trabalho é descrita uma aplicação que facilita o processo de busca e navegação para um usuário do PubMed. Os artigos podem ser analisados sob diversos pontos de vista, como por exemplo palavras mais importantes de resumo e títulos, tópicos e autores. Ao clicar em um destes itens, os artigos são filtrados e exibidos aqueles que satisfazem as condições dos filtros selecionados. Também é implementado um algoritmo de agrupamento de artigos por categorias que permite a distribuição de artigos em tópicos em comum.

Os termos extraídos dos artigos do PubMed são guardados em um banco de dados e são atribuídos às categorias semânticas do *Unified Medical Language System*(UMLS) através do programa *MetaMap*[3] do *National Library of Medicine* (NLM). Este programa faz o mapeamento de textos da área de biomedicina para um tesauro do UMLS que possui categorias pré-definidas. Estas categorias são usadas como palavras-chave para referenciar os assuntos principais sendo abordados em um artigo. Os artigos exibidos pelo Anne O'Tate podem ser filtrados através destas categorias.

Outra funcionalidade interessante é a expansão de literatura que ocorre quando o número de artigos sendo exibidos para o usuário é menor do que cinquenta. O sistema busca artigos com temas similares utilizando o sistema de artigos relacionados do PubMed. Os artigos relacionados são selecionados apenas se forem relacionados com mais de 40% dos artigos originais.

Para dar uma visão geral dos artigos retornados a partir de uma busca, é utilizada a função de categorização por tópicos. Esta função utiliza os termos MeSH para dividir os artigos em áreas afins. Artigos recentes e antigos não possuem termos MeSH e são

[3]http://metamap.nlm.nih.gov/

colocados em categorias "Muito recentes"e "Não indexados"respectivamente, os outros artigos são categorizados em não mais do que 15 categorias MeSH utilizando um algoritmo de agrupamento.

A ferramenta Anne O'Tate implementa diversas funcionalidades interessantes como agrupamento de artigos por tópicos e organização por autores e ano, dando a possibilidade do pesquisador encontrar material relevante sem a necessidade de modificar o termo de busca do PubMed. Fato notável é que esta ferramenta tem a capacidade de processar até 25.000 artigos e distribuí-los em categorias em um tempo aceitável para um sistema *Web*.

A maioria dos sistemas que foram estudados utilizam-se dos dados do PubMed para fazer o agrupamento de artigos em sub-tópicos, a utilização de campos como datas de publicação, autores e localidades são óbvias. A divergência entre estes sistemas está relacionada às técnicas e os algoritmos usados para determinar os sub-domínios em grupos de artigos retornados a partir do PubMed. Muitos dos sistemas utilizam-se de ontologias para atribuir descritores aos artigos como o GoPubMed (DOMS; SCHROEDER, 2005), PubReMiner [4] e PubMedAssistant (DING et al., 2006), sistemas como os de XplorMed (PEREZ-IRATXETA; BORK; ANDRADE, 2001) e BioIE (DIVOLI; ATTWOOD, 2005) extraem relacionamentos entre as palavras contidas em resumos e títulos para representar e agrupar artigos similares e ainda existem sistemas que mostram artigos relacionados e artigos ordenados por relevância como em HubMed (EATON, 2006) e Relemed (SIADATY; SHU; KNAUS, 2007) respectivamente.

O sistema apresentando neste trabalho utiliza uma abordagem diferente para o agrupamento de artigos, através do processamento de linguagem natural são extraídos possíveis descritores para os artigos e estes por sua vez passam por um processo para detectar os mais descritivos. Essa abordagem fornece descritores independentes de vocabulários controlados como o MeSH ou GO, resultando em conceitos mais genuínos para um determinado artigo. Utilizando estes conceitos extraídos localmente, são obtidos os conceitos globais que descrevem temas em comum.

2.3 Métricas de avaliação

Para avaliar a qualidade dos resultados obtidos em um sistema de RI são utilizadas métricas que fazem comparações dos resultados obtidos por um sistema com os resultados que se esperam deste sistema. Basicamente são operações matemáticas de conjuntos que

[4] http://bioinfo.amc.uva.nl/human-genetics/pubreminer/

revelam valores percentuais de um determinado aspecto do sistema. As duas métricas mais conhecidas e bem aceitas na comunidade de RI são o *recall* e o *precision* (TURPIN; SCHOLER, 2006).

O *recall* serve para representar a quantidade de documentos relevantes que foram recuperados, não levando em conta os documentos irrelevantes que também foram recuperados.

recall = (núm. de documentos corretamente recuperados) / (núm. total de documentos relevantes)

Precision avalia o que o *recall* não leva em conta, verificando quantos dos documentos recuperados são relevantes.

precision = (núm. de documentos corretamente recuperados) / (núm. total de documentos recuperados)

O número de palavras-chave corretamente extraídas na avaliação justa é feito calculando o somatório dos pesos, assim se temos duas correspondências exatas (1) e uma de Sufixo Classe 1 (0.9) e uma de Prefixo Classe 2 (0.5), o número de correspondências corretas é 3.4. Cada correspondência fica entre 0 e 1.

A *F1-measure* é uma métrica que expressa a média balanceada entre o *recall* e o *precision* (FELDMAN; SANGER, 2007).

$$\textbf{\textit{F1-measuere}} = \mathbf{2} \;/\; (\;(\mathbf{1} \;/\; \textbf{\textit{recall}}) + (\mathbf{1} \;/\; \textbf{\textit{precision}}))$$

COMO SERÁ APRESENTADO NOS PRÓXIMOS CAPÍTULOS

3 *BioSearch Refinement*

O *BioSearch Refinement* foi desenvolvido para auxiliar o processo de busca no portal NCBI, especificamente no banco de dados PubMed. Possui uma estrutura de aplicação *Web* utilizando o padrão MVC (*Model View Controller*), para que qualquer usuário com acesso à *Internet* possa utilizar o sistema. A implementação foi feita na linguagem de programação Java, visto que ela é flexível, computacionalmente eficiente (ORACLE, 2010), tem suporte à plataforma *Web* e sendo a linguagem mais utilizada no mundo (TIOBE, 2011), possui uma extensa documentação e uma grande quantidade de API's de processamento de linguagem natural. O fluxo básico de funcionamento da aplicação pode ser visto na Figura 1.

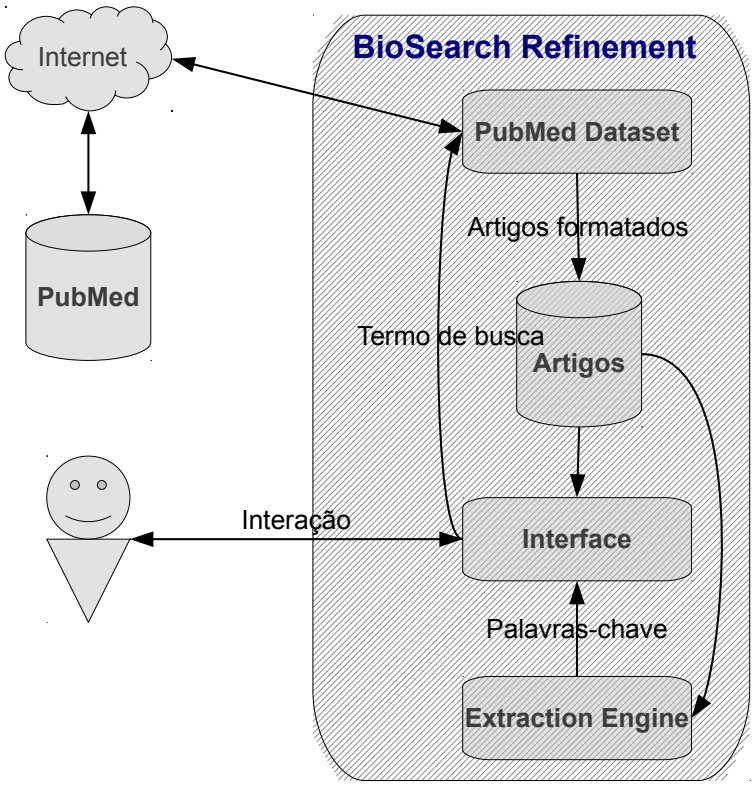

Figura 1: Processo geral do sistema *BioSearch Refinement*

O *BioSearch Refinement* é composto pelo *PubMed Dataset* (biblioteca de criação de conjuntos de dados explicada adiante) que se comunica com o NCBI e acessa o PubMed para obter os artigos e pelo *Extraction Engine* que é o algoritmo que determina as palavras-chave.

A interação com o sistema é dada da seguinte maneira:

1. O usuário interage com aplicação fornecendo o termo de busca.
2. O termo buscado é passado para o *PubMed Dataset* que faz a comunicação com o PubMed.
3. Os 20 primeiros artigos mais recentes são exibidos na tela para que o usuário possa navegar. Um mecanismo de paginação é implementado, para que a aplicação seja leve e funcione com rapidez. Mais artigos poderão ser carregados se o usuário fizer uma requisição para exibir outra página.
4. Para executar o algoritmo de extração de palavras-chave (*Extraction Engine*) o usuário clica em um botão "*Generate keywords*", mostrada na Figura 3.
5. Ao finalizar o processamento, as palavras-chave mais relevantes aparecem para o usuário permitindo a filtragem dos artigos exibidos.
6. Ao clicar em uma das palavras-chave, uma nova aba é aberta dentro da *interface* do sistema. Nesta aba são exibidos os artigos relacionados com aquela palavra-chave e são geradas novas palavras-chave para aquele conjunto de artigos.

A seguir são explicados o três principais componentes do sistema *BioSearch Refinement*, a *interface*, o *PubMed Dataset* e o *Extraction Engine*, respectivamente.

3.1 *Interface*

A camada de visão do sistema *BioSearch Refinement* foi desenvolvida utilizando a tecnologia JSF 2 (*Java Server Faces*) que permite o desenvolvimento de aplicações *Web* utilizando a linguagem Java. O JSF estabelece um padrão para o desenvolvimento de *interfaces server-side*, ou seja, os eventos que ocorrem na tela (lado do cliente) são enviados para processamento no lado do servidor de aplicação. A implementação do JSF utilizada foi a Mojarra (Sun) e o servidor de aplicação Tomcat 7.0.14.0 (Apache).

O JSF possibilita a utilização de bibliotecas de componentes prontas que aceleram o processo de desenvolvimento, poupando o desenvolvedor de escrever grandes quantidades de código JavaScript, HTML e CSS. Existem diversas bibliotecas de componentes no mercado como o ICEFaces, RichFaces, ADFFaces, etc. A biblioteca de componentes que melhor se adequou ao propósito do projeto foi a PrimeFaces da Prime Technology, possuindo componentes como tabelas e abas, essenciais na *interface* do projeto.

Na Figura 2 é mostrada a tela inicial do sistema onde o usuário irá digitar um termo que será buscado, e na Figura 3 a tela com os resultados obtidos. Ao pressionar o botão "*Search*", o *PubMed Dataset* primeiramente busca o número total de artigos disponíveis no PubMed e em seguida faz o *download* dos artigos mais recentes para serem exibidos na primeira aba, isto é feito através de uma requisição AJAX.

Um componente chamado *ajaxStatus* do PrimeFaces mostra ao usuário que uma requisição AJAX está sendo processada através de um gif animado. Requisições AJAX permitem que a página seja dinamicamente carregada, permitindo atualizar apenas partes do HTML, dando a impressão de um sistema *desktop*. Outro beneficio da utilização de AJAX é a rapidez, pois apenas partes da página são recarregadas.

Figura 2: Tela inicial do *BioSearch Refinement*

Na Figura 3, a aba foi aberta com o termo de busca solicitado e foram carregados os primeiros 20 artigos. Os artigos que são exibidos ficam guardados na memória (cacheados), se for feita a requisição para exibir outra página, e os artigos já exibidos ficam em *cache* na sessão do navegador. Os artigos são exibidos em uma tabela similares à exibição do PubMed, informando o título, os autores, a revista na qual o trabalho foi publicado e o ano de publicação. Para visualizar informações complementares como o resumo do artigo, é necessário clicar na seta situada á esquerda para expandir um artigo.

Ao clicar no botão "*Generate keywords*", o algoritmo *Extraction Engine* é chamado para gerar as palavras-chave nos artigos que estão em *cache*. Na Figura 4 são mostradas as palavras-chave geradas para os artigos mostrados na figura anterior. Ao clicar em uma

Figura 3: Tela após efetuar uma busca

palavra-chave, uma nova aba será aberta. O título desta aba será o título da aba anterior concatenado com a palavra-chave selecionada, no exemplo da figura seria "*mycobacterium avium virulence genotype / mycobacterium avium subsp*".

Existem dois tipos de abas no sistema, o primeiro se chama *LoadingTab*, esta aba representa o termo de busca inicial e implementa a paginação de artigos. A outra aba é a *OpenedTab*, gerada a partir de uma *LoadingTab* e serve para mostrar os artigos refinados. Quando o usuário clica em uma palavra-chave, é aberta uma *OpenedTab*, e nesta aba as palavras-chave são recalculadas usando o *Extraction Engine*. O processo de abertura de abas e processamento de palavras-chave é recursivo.

As palavras-chave passam também por um processo de filtragem. Por exemplo, os termos que compõem o título da *LoadingTab* da Figura 3 (*mycobacterium*,*avium*,*virulence*,*genotype*) são guardados em uma lista de *stop words*. Se os termos da palavra-chave gerada pelo algoritmo *Extraction Engine* estiverem na lista de *stop words*, esta palavra-chave não será exibida, pois ela é irrelevante para o usuário. Ao abrir uma aba, a lista é incrementada com os termos da palavra-chave selecionada.

Figura 4: Palavras chave geradas pelo *Extraction Engine*

3.2 *PubMed Dataset*

Para avaliar o *BioSearch Refinement* quanto à qualidade de suas predições e obter estatísticas importantes sobre os artigos do PubMed, é necessário um conjunto de programas que possibilitem a gestão destas informações. O *PubMed Dataset* é uma API desenvolvida para a criação de conjuntos de dados teste com os artigos do PubMed, e também, serve como ponto de acesso ao PubMed no sistema *BioSearch Refinement*.

O objetivo principal do *PubMed Dataset* é oferecer ao programador de um sistema de R.I. o acesso aos dados do PubMed através do E-Utils (NCBI, 2010) e a capacidade de gravar e ler estes dados do disco através de um programa Java. Isto é feito através de conjuntos de dados *datasets* que são salvos em disco através de serialização (uma técnica de persistência para objetos). Um conjunto de dados é constituído de artigos científicos que são recuperados do PubMed utilizando um termo de busca e uma configuração de *download*.

3.2.1 E-Utils

O E-Utils é um *Web Service* composto por um conjunto de programas que utilizam URL's para fazer requisições e retornam dados em diversos formatos com HTML ou XML. Dentre estes programas, existem o E-Search, utilizado para fazer buscas e o E-Fetch para recuperar os dados. O arquivo estruturado XML do PubMed atende o padrão DTD do NLM Journal Archiving and Interchange, disponibilizado pelo NLM. O objetivo foi fornecer um formato comum que permitisse o intercâmbio de informação. Foram criados vários conjuntos de *tags*, cada um com um propósito. Mas vale ressaltar que nem todos os artigos possuem todas as *tags*, fato que deve cuidadosamente tratado com mecanismo de exceções. Dentre os elementos de um XML estão os seguintes elementos:

- <PMID>: identificador único de um determinado artigo
- <Article>: é o elemento raiz do documento, que contém todos os meta-dados e conteúdo do artigo.
- <ArticleTitle>: contém o título de um artigo.
- <Abstract>: descrição sumária do conteúdo de um artigo.
- <KeywordList>: elemento que contém conjuntos de palavras-chave associadas com todo o documento e que podem ser usadas para fins de identificação e indexação.

Estas palavras chave são atribuídas pelos autores do artigo.

- <MeshHeadingList>: lista de termos MeSH.

Estas informações são usadas pelo sistema *BioSearch Refinement* para determinar as palavras-chave e exibir o conteúdo dos artigos.

3.2.2 Funcionalidades

A Figura 5 apresenta o funcionamento geral da biblioteca PubMed Dataset. A partir de parâmetros de entradas especificados, a ferramenta se conecta à base de dados PubMed. O processo de obtenção do conjunto de dados envolve o processamento de documentos XML, que são retornados dos servidores do NCBI. Destes documentos são extraídos os dados requisitados pelo usuário através de uma configuração de *download*.

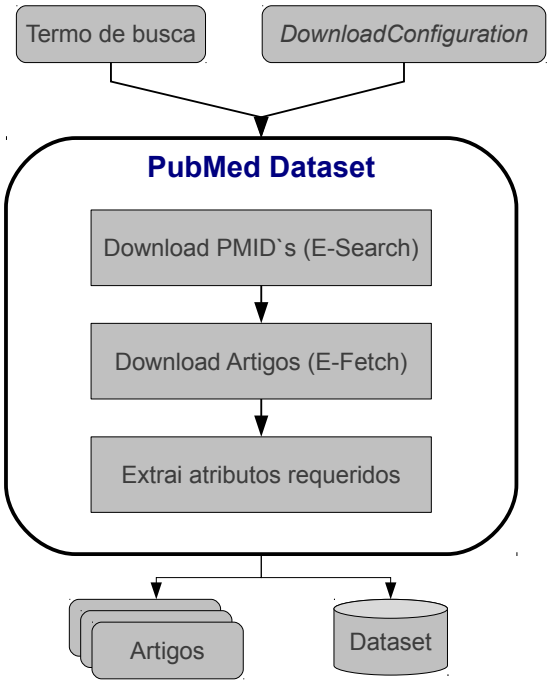

Figura 5: Processo de obtenção de artigos do *PubMed Dataset*

Os parâmetros de entrada obrigatórios são:

- o termo de busca que será utilizado para fazer uma query no banco de dados do NCBI e que irá retornar um conjunto de artigos;
- número máximo de artigos a serem buscados para criar o conjunto de dados;

- os atributos que serão extraídos e usados para a criação das instâncias de um artigo. Se um dos atributos não for encontrado em um artigo do XML, este artigo será descartado. Como atributo de saída pode ser especificado qualquer elemento presente no arquivo XML.

O arquivo de saída é um arquivo serializado no formato (*termo de busca_ número de artigos_ articles.ser*, ex: *mycobacterium tuberculosis_ 2754_ articles.ser*). Este arquivo pode ser desserializado a qualquer momento e os artigos contidos neste estarão disponíveis prontamente. Uma instância do diagrama de classes da biblioteca é apresentada na Figura 6.

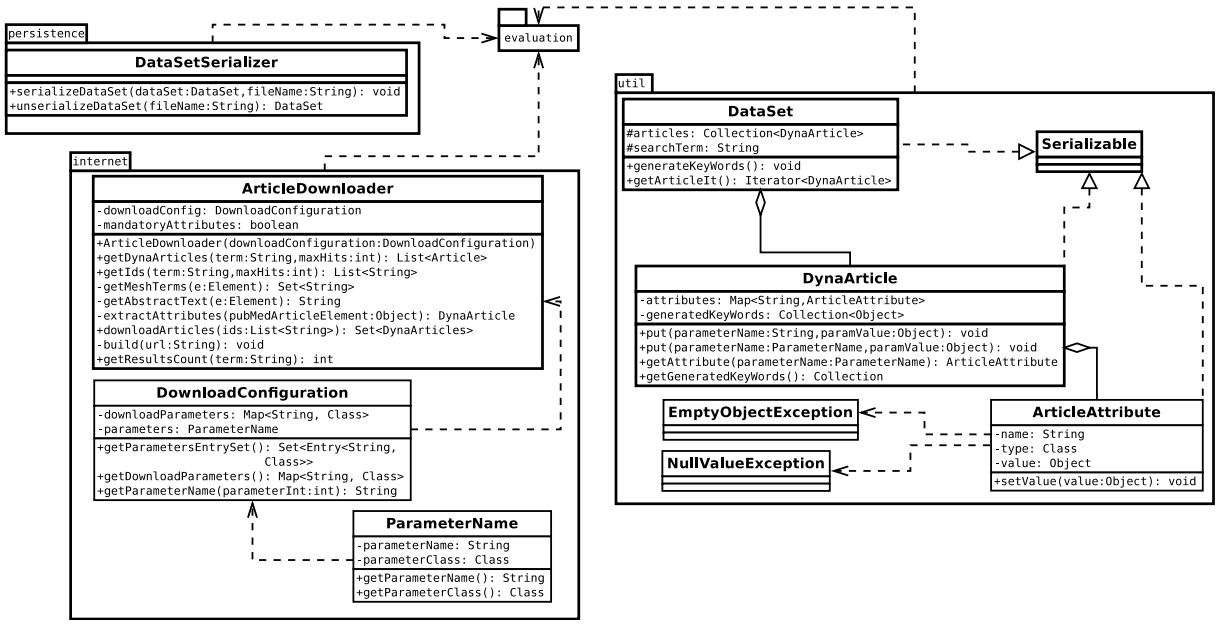

Figura 6: Diagrama de classes UML do *PubMed Dataset*

O *ArticleDownloader* é a classe responsável por obter os artigos do banco de dados PubMed e é construída com uma configuração $DownloadConfiguration$. Se o número máximo de artigos solicitados for muito grande, o processo de obtenção de artigos é fracionado em partes para evitar time out por parte do servidor, isto é implementado no método $getDynaArticles()$ de $ArticleDownloader$. Primeiramente são obtidos os $PMID$'s através do E-Search e em seguida, utilizando estes, os artigos são recuperados e colocados em um DOM (*Document Object Model*) utilizando o E-Fetch. O DOM é uma convenção para representar documentos XML e outros dentro de uma linguagem de programação. Em seguida, os elementos são percorridos e são recuperados aqueles que foram selecionados pelo usuário no $DownloadConfiguration$.

- *DownloadConfiguration* – é onde são definidas as configurações do download. São especificados os atributos que estarão presentes em cada artigo recuperado.

```
DownloadConfiguration config = new DownloadConfiguration(ABSTRACT,
    AUTHOR,TITLE);
```

- *DownloadParameter* – É um enum no qual estão definidos os atributos que podem ser recuperados, estes atributos estão contidos em um $Map < String, Class >$ que define o nome e a classe de um atributo. Um título de artigo, por exemplo, possui $String =" Title"$ e $Class = java.lang.String$, já uma lista de termos MeSH, possui $String =" MeshTerms"$ e $Class = java.util.Set$, pois representa uma lista de termos.

Ao receber um *DownloadConfiguration* o *ArticleDownloader* pode efetuar a recuperação dos artigos através do método $getDynaArticles(termoDeBusca, maxHits)$ ou $getDynaArticles(termoDeBusca, primeiroArtigo, tamanhoDePagina)$. O primeiro método recebe um termo de busca, e o numero máximo de artigos que serão buscados no banco. O segundo método recupera uma fração de documentos, através dos dois últimos argumentos e pode ser usado para paginação. Ambos métodos retornam uma $List < DynaArticle >$ que possui as instâncias dos artigos que foram recuperados.

- *DynaArticle* – Esta classe representa um artigo do PubMed. Possui uma $Collection < Object >$ que serve para armazenar as palavras-chave geradas por um algoritmo qualquer, e possui um $Map < String, ArticleAttribute >$ que serve para guardar os atributos extraídos do PubMed. Um ponto importante é que esta classe por possuir esta estrutura característica, permite que sejam criadas instâncias de artigos dinamicamente, ou seja, com uma quantidade variável de atributos.

- *ArticleAttribute* – O atributo de um artigo é um dos fatores que permite a criação de artigos dinâmicos. Esta classe possui nome do atributo, tipo do atributo e seu valor.

O *Dataset* é quem vai encapsular os artigos e que será utilizado como unidade de persistência, ou seja, será gravado em disco utilizando serialização. Mas esta classe nunca será utilizada diretamente, pois possui um método abstrato chamado $generateKeyWords()$ que será utilizado para o processo de geração de palavras-chave. Então, a utilização correta de um *Dataset* é se estendendo ele e implementando o método $generateKeyWords()$.

Como dito anteriormente, alguns artigos no PubMed podem não possuir determinados atributos, por isso o *ArticleDownloader* possui uma variável booleana chamada *mandatoryAttributes*. Se o valor da variável for setado como *true*, todos os atributos que foram passados na $DownloadConfiguration$ serão considerados obrigatórios, e os artigos que não possuírem algum destes atributos serão descartados.

Finalmente, a classe *DatasetSerializer* é que faz a serialização e desserialização dos *Dataset*'s. Na serialização, ela recebe o *Dataset* a ser salvo em disco e um nome para o arquivo que será salvo, geralmente um bom nome é o termo utilizado na criação do *Dataset*. A este nome é acoplado o número de artigos existentes no *Dataset* e a extensão ".ser" é concatenada automaticamente. A desserialização é o processo inverso, no qual a partir do nome de arquivo, o *Dataset* é carregado para uma instancia. É importante observar que quando o *Dataset* é muito grande, pode ocorrer um erro de *java.lang.OutOfMemoryError: Java heap space*, pois a memória da máquina virtual Java estoura e não consegue terminar a serialização ou o processo inverso. Para solucionar este problema, a memória da maquina virtual deve ser aumentada, passando um parâmetro na linha de comando:

```
$ java -Xmx=256M seuProgramaQueEstouraAMemória
```

3.3 *Extraction Engine*

O algoritmo *Extraction Engine* foi desenvolvido como parte do sistema *BioSearch Refinement* e é o componente do sistema que faz a extração das palavras-chave dos artigos e os organiza em tópicos mais abordados. A API de processamento de linguagem natural escolhida foi a **Alias-i Ling Pipe** que possui licença de uso gratuita para propósitos de pesquisa. Na recuperação e indexação dos artigos as palavras-chave jogam um papel integral, a extração de palavras-chave do corpo de um artigo tenta capturar a essência do assunto do mesmo, sem trabalhar com anotações humanas. A extração de palavras-chave é executada sobre os resumos dos artigos, pois a maioria dos artigos não disponibiliza o texto completo, mas apenas o resumo (HULTH, 2003).

A Figura 7 apresenta o esquema geral do algoritmo proposto para a obtenção das palavras-chave e a Figura 8 destaca um fluxo de execução para a obtenção de palavras-chave em um artigo. O algoritmo é dividido em vários passos, explicados a seguir. Para cada resumo:

1. **Divisão dos resumos em frases**: o texto de um resumo é composto por frases,

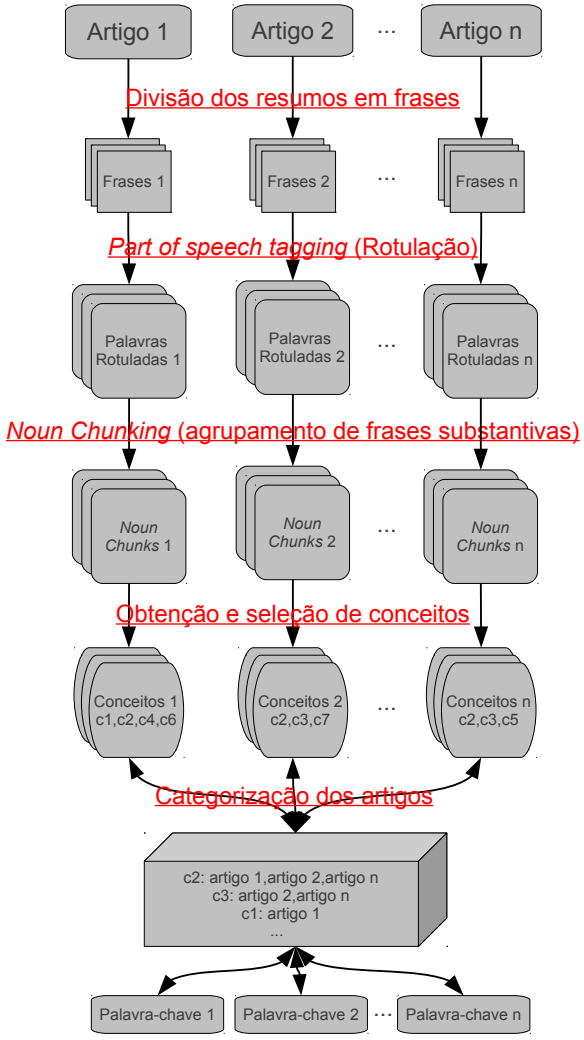

Figura 7: Processo geral do algoritmo *Extraction Engine*

um tokenizador trunca o texto em frases para utilizar na próxima etapa.

2. **Rotulação ou POST (*Part Of Speech Tagging*)**: determina as classes gramaticais (*tags*) de cada palavra na frase.

3. ***Noun Chunking***: baseado no princípio de que os substantivos são os melhores descritores, o algoritmo seleciona termos compostos obrigatoriamente por substantivos. Para isto, foram definidos padrões sintáticos compostos por advérbios, adjetivos, numerais e alguns símbolos para conectar substantivos e formar termos mais descritivos.

4. **Obtenção e seleção dos conceitos**: os conceitos servem para capturar a essência dos documentos e agrupar diversos *noun chunks* similares em uma estrutura única. Um conceito é composto pelas palavras substantivas que compõem um *chunk*, assim

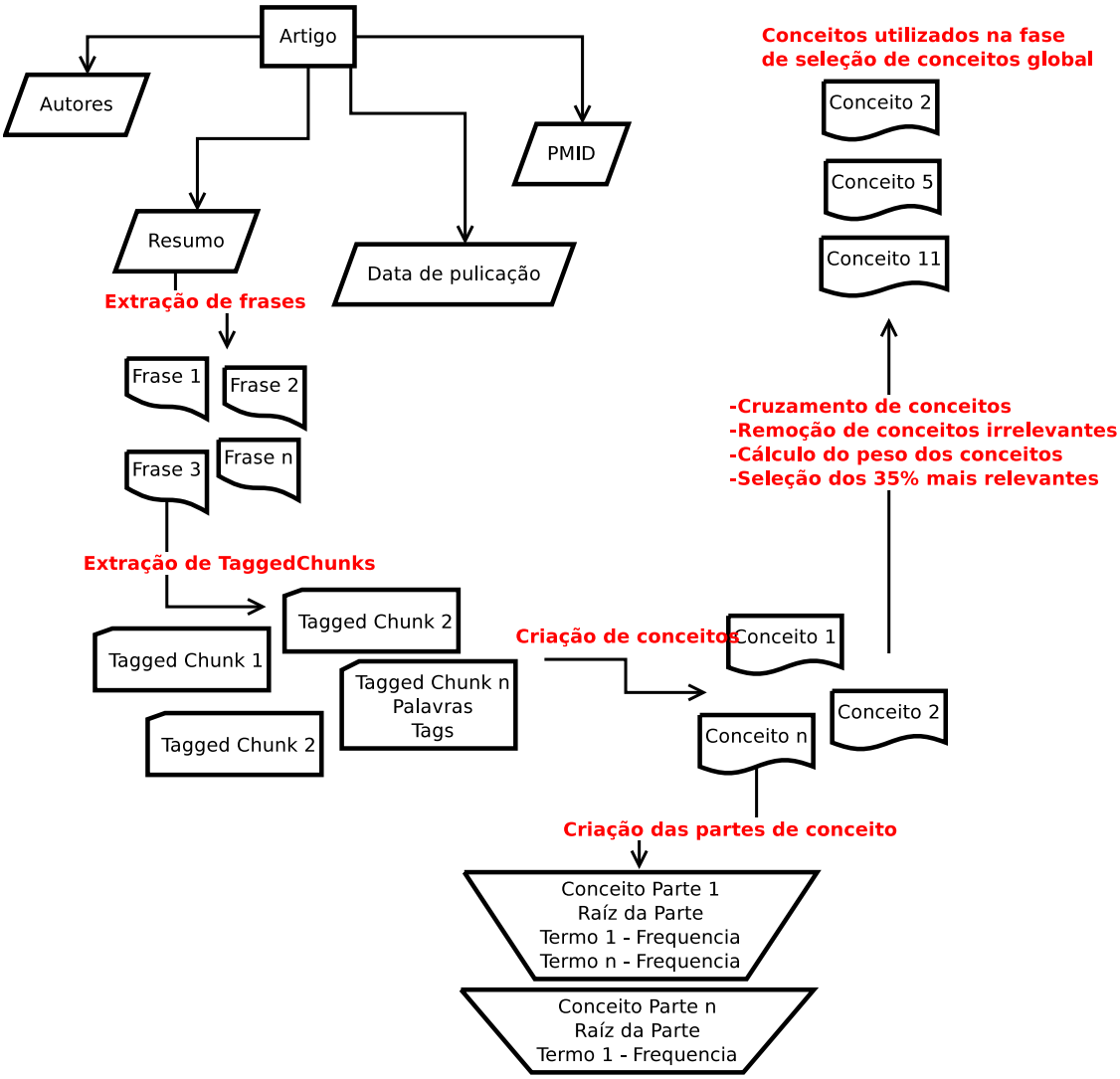

Figura 8: Exemplo de obtenção de palavras-chave para um artigo

os termos *green cells* e *blue cells* fariam parte de um mesmo conceito *cells*. São usados apenas os substantivos, pois o objetivo é generalizar o assunto sendo abordado no texto e agrupar assuntos semelhantes em um único conceito.

Para ilustrar a necessidade do uso de conceitos, observe os seguintes termos extraídos de um dos resumos: "*other vegetable system*", "*the vegetable systems*" e "*the vegetables*". Se analisarmos as duas primeiras frases percebemos que elas tratam de um mesmo assunto "*vegetable systems*", porém a máquina irá tratar "*system*" e "*systems*" como duas palavras diferentes. Para contornar este tipo de discrepância é utilizado um algoritmo de normalização (*stemming*). Usamos uma implementação do algoritmo de Porter (PORTER, 1980) fornecida pela API LingPipe, que permite remover sufixos de palavras em inglês.

Visto que as palavras são normalizadas, as frases "*other vegetable system*" e "*the vegetable systems*" são tratadas como sendo um mesmo conceito. Por outro lado, a terceira palavra só possui "*vegetable*" como substantivo, o que acarreta na criação de um segundo conceito "*vegetable*".

3.3.1 Obtenção de conceitos

O processo de conceitualização se divide em duas etapas. Na primeira, os conceitos são extraídos para cada artigo e são determinados os mais relevantes. Na segunda etapa, são selecionados os conceitos que abordam a maior quantidade de artigos diferentes, determinando assim os temas mais abordados pelos artigos.

Conceitualização local

O processo de conceitualização se inicia com a criação de uma estrutura de armazenamento de conceitos, mostrada na Figura 9 chamada *ArticleConcepts*, que irá conter os conceitos de um determinado artigo. Um conceito é composto por um ou mais termos e cada um destes termos possui uma frequência no texto e pode possuir variações no texto como por exemplo plurais. Por isso um *Concept* é composto por diversas partes *ConceptPart*, onde cada parte possui:

- Peso (*weight*): representa a interação das partes de outros conceitos com este conceito, um processo de cruzamento de informações explicado adiante.
- Raiz da parte (*partStem*): a raiz da palavra que compõe esta parte para a comparação com outras partes.
- Mapa de frequência de ocorrência (*partWords*): usado para contar as frequências das diferentes palavras que em algum momento fizeram parte deste 0conceito.

Ao se criar um conceito, automaticamente, são criadas suas *ConceptPart* e este conceito é adicionado em *ArticleConcepts*. Termos iguais ou similares são agrupadas sob o mesmo conceito. Por exemplo: o conceito "*other vegetable system*" possui três partes: "*other*", "*vegetable*" e "*system*". Ao aparecer "*the vegetable systems*", os conceitos seriam integrados utilizando a estrutura a seguir. A comparação entre os termos é feita na raiz de suas partes substantivas.

Conceito: *vegetable system*

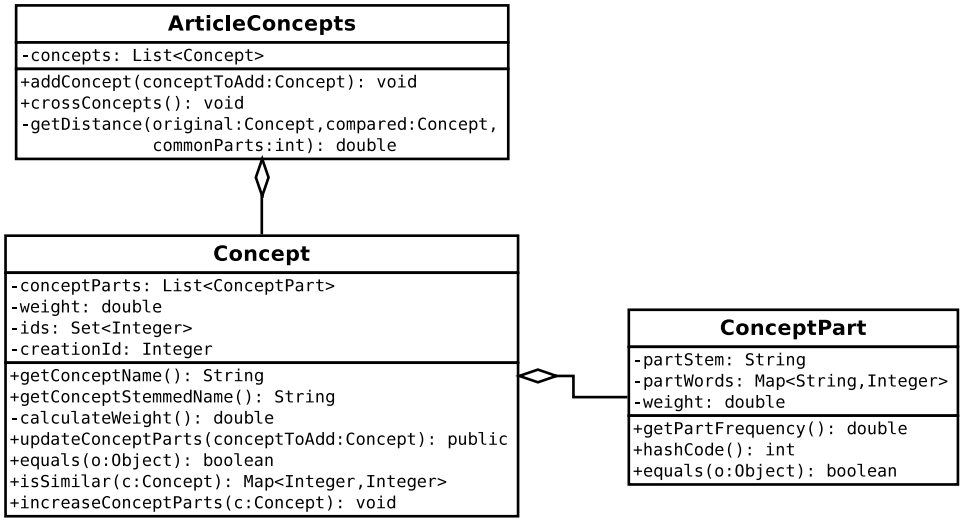

Figura 9: Diagrama de classes de conceitualização local

Parte 1:*vegetable*, Raiz:*vegetable*, Mapa:*vegetable* (frequência: 2)

Parte 2:*system*, Raiz:*system*, Mapa:*system* (frequência: 1), *systems* (frequência: 1)

Cruzamento de informações

Para melhor descrever um artigo, é necessário avaliar cada conceito, valorizando os conceitos realmente relevantes e diminuindo o peso de conceitos pouco relevantes.

O processo de atribuição de peso de um conceito localmente é feito através um cruzamento de informação das partes de todos os conceitos similares a ele. Isso faz com que conceitos diferentes e com partes em comum recebam um peso maior, evidenciando ,assim, o grupo de conceitos que são mais abordados naquele artigo. Um conceito é similar a outro conceito, se estes possuem pelo menos uma parte em comum. A similaridade é calculada através da distância entre conceitos, e é a divisão do número de partes em comum pelo número de partes total do maior conceito.

A distância entre o *Conceito*1 do exemplo anterior e o *Conceito*2, a seguir, é calculado como:

Conceito2: *vegetable oil*

Partes em comum: 1

Distância: $\frac{1}{2} = 0.5$

O cruzamento de dois conceitos é feito para determinar o peso das partes similares, mostrado na Listagem 1:

```
double weight = (fromPart.getPartFrequency() * distance);
weight += toPart.getWeight();
toPart.setWeight(weight);
```

Listagem 1: Cruzamento de dois conceitos

Quanto maior a distância entre os conceitos, menor a influência de um no outro. O peso é atualizado para ambos os conceitos. Ao finalizar este processo, é feito o cálculo do peso dos conceitos utilizando-se do peso de suas partes, da frequência das partes e do número de partes que o conceito possui. O peso de um conceito é:

$$\left(\sum_{partes} parte.frequencia + parte.peso\right) * (1 + fator\,de\,ajuste * num.\,de\,partes)$$

Em primeiro lugar, é calculada a importância do conceito no texto do resumo. Para todas as partes de um conceito, é feito o somatório da sua frequência no texto e da sua interação com outros conceitos similares (peso do conceito). Este valor é então multiplicado pelo segundo parenteses da equação, onde é dada prioridade aos conceitos que possuem mais partes. Conceitos com mais partes são mais representativos, logo mais importantes. Além de aumentar a representatividade dos conceitos é feito o balanceamento entre a pequena quantidade de conceitos com muitas partes e pouca frequência e de conceitos com poucas partes e frequência alta. O fator de ajuste foi considerado 0.05 para que a cada parte que o conceito possui, ele receba um aumento no seu peso de 5%, logo em um conceito de 4 partes este aumento é de 20%. O valor do fator de ajuste foi determinado empiricamente.

Ao fim do processo, conceitos de apenas uma palavra e com menos de 4 letras são removidos, considerados pouco representativos. Os conceitos com o maior peso são aqueles que foram mais referenciados no resumo, logo, são os melhores descritores. Cada conceito obtido pelo algoritmo é linearmente mapeado em uma predição de palavra-chave do artigo. Este processo é repetido para todos os resumos.

Conceitualização global

O processo de obtenção de conceitos é realizado em cada resumo de artigo. O conceito que referencia o maior número de artigos distintos tem maior importância, pois ele trata de um assunto comum, ou seja, é mais abordado em todos os artigos e passa a formar parte das palavras-chave globais. As palavras-chave globais formam o melhor conjunto de descritores para todos os artigos recuperados em uma consulta.

Cada conjunto de conceitos local é ordenado pelo peso decrescentemente, e deste

conjunto apenas uma parte é selecionada para a conceitualização global. O principal motivo deste corte é a exclusão de conceitos pouco descritivos ou muito gerais que possuem pontuação relativamente baixa, e em segundo lugar a melhora da performance do sistema através da diminuição de conceitos a serem processados na fase global.

Resumos com maior quantidade de texto possuem mais palavras-chave do que resumos com algumas linhas, por isso a quantidade de conceitos selecionada é uma porcentagem do total de conceitos extraídos de um resumo. Foi verificado através de testes que os melhores resultados eram obtidos com a seleção de 35% dos conceitos, assim eliminando 65%.

No processo de globalização, os conceitos locais passam por uma verificação de integração, que permite que um mesmo conceito extraído de vários artigos seja unificado em um só. A unificação de conceitos implica na atualização da frequência desse conceito. Após a análise de todos os conceitos de todos os artigos, é feita a ordenação decrescente, seguindo os critérios: frequência (número de artigos que o referenciam) e peso do conceito. Os conceitos que possuem o maior número de artigos sendo referenciados, são os conceitos com maior abrangência, logo são considerados as melhores palavras-chave para sumarização.

Na *interface* são exibidas as cinco melhores predições, mas algumas podem ser descartadas pelo algoritmo por serem consideradas como *stop words*.

3.3.2 Implementação

Na Figura 10 está representado o diagrama de classes do *Extraction Engine*. A classe que faz o processamento dos artigos é a $KeyWordProcessor$, ela também dá acesso aos conceitos mais referenciados através de um $Iterator < Concept >$ pelo método $getConceptIt()$. O método *processArticles* recebe os artigos a serem processados e chama *conceptualize*. Este método irá percorrer todos os artigos, recuperar os resumos de cada um deles e tokenizá-los em frases usando o $SentenceSplitter$.

As frases são passadas para o $MedLineSentenceChunker$ que determina os $TaggedChunk$'s de cada frase. Estes *chunks* são usados na conceitualização local explicada anteriormente para determinar os conceitos mais relevantes de um artigo.

Os conceitos obtidos localmente são guardados no $GlobalConceptHolder$ através do método $insertConcept$. A cada conceito que é inserido, é feita uma busca para determinar se um conceito parecido conceitualmente já existe. A comparação de dois conceitos é feita

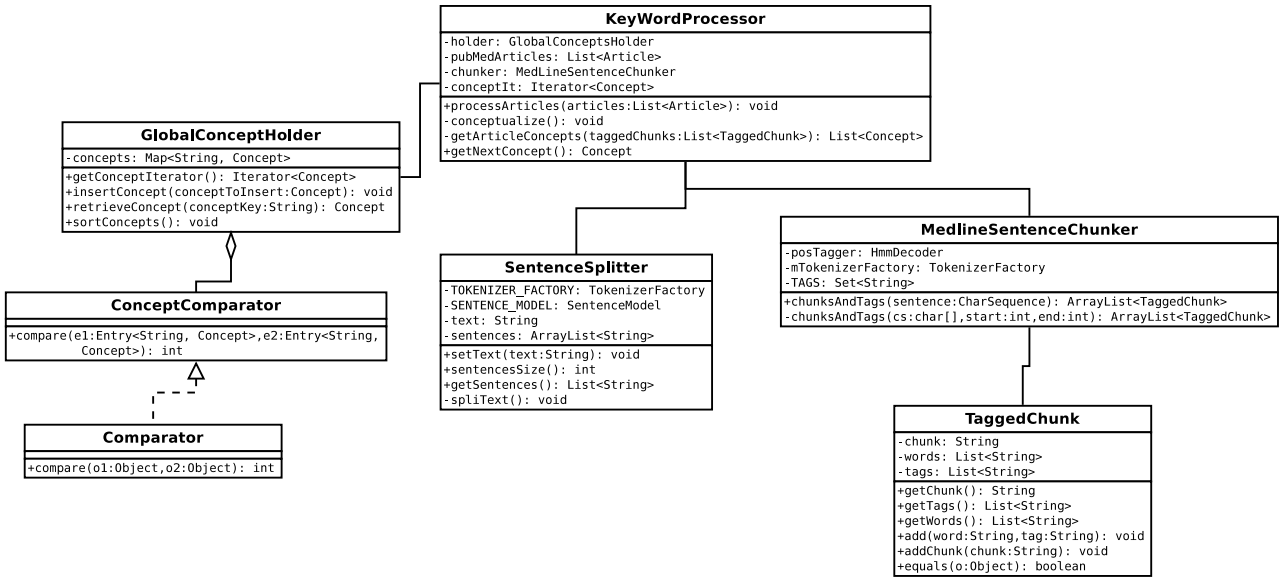

Figura 10: Diagrama de classes de conceitualização

através de uma chave, que é composta pelas raízes das partes de um conceito. Se dois conceitos são similares, a lista de PMID‘s deste conceito é incrementada como também a frequência de ocorrência de suas partes.

Após todos os conceitos serem inseridos no *GlobalConceptHolder*, é chamado o método *sortConcepts* para que os conceitos sejam ordenados segundo o seu número de PMID‘s e depois por frequência de ocorrência. Nesta etapa o *BioSearch Refinement* já pode obter o $Iterator < Concept >$ para navegar pelos conceitos mais referenciados e selecionar as palavras-chave.

4 *Resultados e discussões*

Neste capítulo são apresentados os resultados para a biblioteca *PubMed Dataset*, o algoritmo de sumarização *Extraction Engine* e o sistema *BioSearch Refinement*.

4.1 Resultados do *PubMed Dataset*

Para a criação dos conjuntos de dados primeiramente foram definidos os termos de busca, o número máximo de artigos a ser buscado e em seguida foram definidos os atributos dos artigos que seriam necessários nos conjunto de dados. Cada artigo necessita conter o resumo que será utilizado para a obtenção das palavras chave geradas pelo algoritmo e também das palavras chave definidas pelos autores, pois estas são a referência da comparação com as palavras automaticamente extraídas pelo *Extraction Engine*.

Existem dois tipos de palavras-chave usadas na base de dados do PubMed: os termos de indexação MeSH e as palavras-chave fornecidas pelos autores do artigo. Os experimentos foram realizados utilizando como referência os termos MeSH, pois se encontraram poucos artigos com palavras-chave dos autores.

A biblioteca foi avaliada sob os seguintes aspectos: facilidade de uso (linhas de código para executar a tarefa desejada) e performance (tempo de execução). O objeto de estudo foi a criação de dois conjuntos de dados que seriam utilizados para analisar o *Extraction Engine*, mostrados na Tabela 1.

Após a criação dos conjuntos de dados, foram executadas duas rotinas implementadas na classe *ConceptDataSet* que estende *DataSet*. A primeira (*removeSearchTermFromData*()), remove o termo de busca dos conjuntos de palavras-chave geradas e termos MeSH, pois são irrelevantes para o pesquisador já que ele as usou como termo de busca. A segunda rotina (*intersectKws_Abstract*) desconsidera os termos MeSH que não estão no texto do resumo, pois como o algoritmo avaliado é de extração, só se pode comparar palavras chave extraídas com palavras do próprio resumo.

Conjunto de dados 1	
Termo de busca	*mycobacterium tuberculosis*
Número de artigos solicitados	10000
Número de artigos recuperados	8045
Conjunto de dados 2	
Termo de busca	*h1n1 influenza virus*
Número de artigos solicitados	5000
Número de artigos recuperados	2858

Tabela 1: Conjuntos de dados criado com o PubMed Dataset

Como pode-se perceber, a quantidade de artigos recuperados difere da quantidade solicitada. Alguns dos motivos que causaram a redução do número de artigos nos conjuntos de dados foram: alguns artigos no PubMed não possuem os atributos solicitados na configuração de *download*. As rotinas descritas no parágrafo acima causam a exclusão de alguns artigos, quando estes ficam sem palavras-chave geradas ou termos MeSH.

Para a criação de um conjunto de dados (*dataset*), são necessárias no mínimo 5 linhas de código. O método da Listagem 2 recebe o termo de busca e o número de artigos a ser buscado, em seguida, é criada a configuração de *download* com os atributos necessários e o *ArticleDownloader* com esta configuração. Os atributos são configurados como obrigatórios, ou seja, todos os artigos devem possuir os atributos solicitados e então é feito o *download* dos artigos. Finalmente os artigos são passados para o serializador que os salva em disco com o nome do termo para identificação.

```
import static com.uenf.pubmeddataset.internet.ParameterName.*;

public void criarDataSet(String termo, int quantidade) throws IOException
    {
    DownloadConfiguration config = new DownloadConfiguration(PMID,ABSTRACT
    ,MESH_TERMS);
    ArticleDownloader downloader = new ArticleDownloader(config);
    downloader.setMandatoryAttributes(true);
    List<DynaArticle> downloadedArticles = downloader.getDynaArticles(
    termo, quantidade);
    DataSet conceptDS = new ConceptDataSet(articles, "mycobacterium
    tuberculosis");
    DataSetSerializer.serializeDataSet(conceptDS, termo);
}
```

Listagem 2: Criação de um *DataSet*

4.1.1 Discussão dos resultados do *PubMed Dataset*

Com a utilização da ferramenta, a construção dos conjuntos de dados aconteceu de maneira rápida e fácil, bastando apenas especificar os parâmetros iniciais. A utilização do conjunto de dados se torna muito flexível, pois os seus dados podem ser alterados e gravados novamente, seja para atualizar as palavras-chave geradas ou algum outro parâmetro.

A partir da Tabela 2, pode-se observar que, apesar do tempo para download ter sido consideravelmente grande para 10.000 artigos, esta fase só é executada uma vez, pois o conjunto de dados é gravado em disco após a execução e sempre vai ser dependente do tráfego na rede. Pode-se ver que o tempo levado para carregar o conjunto de dados é muito pequeno, tornando sua utilização muito conveniente para testes. A quantidade de artigos recuperados varia muito de acordo com os atributos que foram especificados pelo usuário, quanto maior o numero destes, menos artigos serão recuperados, pois menor será a probabilidade de um artigo possuir todos os atributos solicitados. A qualidade dos dados recuperados dependerá do que for disponibilizado no PubMed, pois os conjuntos de dados criados são uma reflexão dos dados contidos no banco de dados de origem.

Conjunto de dados	**Etapas**	**Tempo de execução**
1	*Download*	7,45 minutos
	Serialização	2 segundos
	Desserialização	3 segundos
2	*Download*	3,11 minutos
	Serialização	859 milissegundos
	Desserialização	1 segundo

Tabela 2: Desempenho da ferramenta PubMed Dataset na construção dos conjuntos de dados

Em um ambiente de teste e avaliação, os testes são criados progressivamente e muitas vezes são adicionados casos de teste mais específicos. Dado esse cenário, é óbvio o fato de que os testes são executados diversas vezes. Sem a utilização desta biblioteca, o tempo gasto para a criação dos dados de teste e execução dos testes propriamente ditos levariam um tempo considerável. Como podemos observar na Tabela 2, o tempo gasto para o download dos dados é muito grande e também devemos ressaltar que a consistência do conjunto de dados não é garantida visto que os dados retornados pelo PubMed podem variar com o passar do tempo.

4.2 Resultados do *Extraction Engine*

O ponto crucial do *Extraction Engine* é extrair descritores (palavras-chave) válidos para cada artigo, pois se os descritores selecionados localmente são de qualidade, a sumarização dos artigos no processo global será eficiente. Como o processo de extração de descritores é composto por várias etapas, deve ser feita uma avaliação dos resultados de cada uma destas etapas.

A primeira etapa na qual é importante obter resultados confiáveis é a rotulação das palavras de um resumo. Se as classes gramaticais associadas às palavras forem incorretas ou imprecisas, os conceitos também serão gerados incorretamente. Nesta etapa é utilizado o rotulador MedPost, que possui uma exatidão de 97.43% segundo (SMITH; RINDFLESH; WILBUR, 2004). Conclui-se que as palavras extraídas dos resumos são realmente as que se deseja (*noun chunks*).

A próxima etapa é a obtenção dos conceitos, que são os descritores propriamente ditos. Para determinar a qualidade dos descritores gerados localmente para um artigo pelo *Extraction Engine*, é necessário comparar estes descritores com palavras-chave pré-definidas associadas ao artigo. Estas palavras-chave pré definidas podem ser atribuídas ao artigo pelos autores ou provenientes de alguma outra fonte confiável.

Existem dois tipos de palavras-chave usadas na base de dados do PubMed: os termos de indexação Medical Subject Headings (MeSH), que são termos de um vocabulário controlado atribuído por especialistas do NLM, e as palavras-chave fornecidas pelos autores do artigo. Os experimentos apresentados a seguir foram avaliados utilizando como referência os termos MeSH, pois se encontraram poucos artigos com palavras-chave atribuídas pelos autores.

Foram criados dois *datasets* com a utilização da biblioteca *PubMed Dataset* apresentada anteriormente. Ambos os *datasets* foram criados com um número máximo de artigos igual a 4.500. Cada conjunto de dados passou por um processo de filtragem descrito a seguir:

- dentro do conjunto de artigos inicialmente recuperados, foram excluídos aqueles artigos que não possuíam resumo ou termos MeSH;
- foram desconsiderados os termos MeSH que não estavam contidos no resumo;
- artigos que ficam sem termos MeSH pela exclusão são retirados do dataset

- foi desconsiderada a palavra-chave da busca inicial, no conjunto de palavras-chave geradas e no conjunto de termos MeSH;

Os conjuntos de dados resultantes são mostrados na Tabela 3:

Conjunto de dados 1	
Termo de busca	*mycobacterium tuberculosis*
Número de artigos solicitados	4500
Número de artigos recuperados	3119
Adequados para teste	2773
Conjunto de dados 2	
Termo de busca	*h1n1 influenza virus*
Número de artigos solicitados	4500
Número de artigos recuperados	3487
Adequados para teste	2474

Tabela 3: Conjuntos de dados experimentais

4.2.1 Avaliação

O experimento consistiu na comparação entre as palavras-chave geradas pelo algoritmo (predições) e os termos MeSH obtidos do PubMed. A comparação pode ser feita de duas formas: exata e parcial. A comparação exata ou estrita permite apenas predições positivas de palavras-chave com 100% de exatidão (que coincidem totalmente), sendo que predições que coincidem parcialmente são consideradas negativas ou erros.

A metodologia de avaliação parcial seguida neste trabalho é baseada nas propostas de (CHINCHOR; SUNDHEIM, 1995) e (TSAI et al., 2006). A avaliação parcial possui dois tipos de avaliação: a avaliação relaxada e a avaliação justa. A avaliação parcial considera as predições que coincidem parcialmente como positivas (similaridade = 1).

Ao fazer comparações com os termos MeSH, resultados parciais podem ser bastante próximos das palavras definidas nos termos MeSH e não devem ser tratadas como erro ou fora do conjunto relevante. A proposta de avaliação parcial justa introduz alguns tipos de resultados parciais que podem ser considerados como resultados positivos próximos por similaridade. Para cada caso, os exemplos fornecidos foram obtidos na execução dos conjuntos de dados.

- Coincidência total (Exata): mapeamento completo entre a palavra-chave extraída e a palavra-chave definida manualmente.

Termo MeSH: “rats diseases”
Predição: “rats diseases”

- Correspondência de sufixo Classe 1: correspondência parcial entre as palavras-chave, que compartilham um sufixo comum. Assume-se que a palavra-chave extraída é um subconjunto do termo manual. Semanticamente, o termo manual é mais específico que a palavra-chave obtida automaticamente.
  Termo MeSH: “h1n5 influenza virus”
  Predição: “influenza virus”

- Correspondência de sufixo Classe 2: correspondência parcial entre as palavras-chave, que compartilham um sufixo comum. Assume-se o contrário da Classe 1, o termo manual é um subconjunto da palavra-chave extraída. Semanticamente, a palavra-chave obtida automaticamente é mais específica que o termo manual.
  Termo MeSH: “patients”
  Predição: “HIV patients”

- Correspondência de prefixo Classe 1: correspondência parcial entre as palavras-chave, que compartilham um prefixo comum. Assume-se que o termo manual pode ter um significado diferente que a palavra-chave extraída. Semanticamente, em inglês, a palavra (substantivo) mais à direita define o significado. Portanto, esta classe define que a palavra-chave está contida no prefixo do termo manual.
  Termo MeSH: “h1n1 influenza virus”
  Predição: “h1n1 influenza”

- Correspondência de prefixo Classe 2: correspondência parcial entre as palavras-chave, que compartilham um prefixo comum. Assume-se que o termo manual pode ter um significado diferente que a palavra-chave extraída. Semanticamente, em inglês, a palavra (substantivo) mais à direita define o significado. Portanto, esta classe define que o termo manual está contido no prefixo da palavra-chave.
  Termo MeSH: “mycobacterium tb”
  Predição: “mycobacterium tb infection”

- Correspondência por *substring* interno: correspondência parcial de um termo MeSH com uma predição. Ocorre quando um termo MeSH faz parte do interior de uma predição. Assume-se que a predição é mais especializada do que o termo MeSH, ou seja, possui *tokens* a mais. Correspondências de prefixo e sufixo podem ser considerados mais restritos que a correspondência por *substring* interno.

Termo MeSH: “lobe epilepsy”
Predição: “temporal lobe epilepsy disease”

Nesta avaliação, não foram considerados casos para diferenciar termos em singular e plural, pois tanto os termos MeSH, quanto o algoritmo definem termos no singular.

Cada variação de correspondências parciais vai ser associada a um peso, que corresponde a uma penalidade de acordo com a proximidade por similaridade da palavra-chave extraída do termo manual. Os pesos e penalidades por classe são mostrados na Tabela 4.

Prioridade	Tipo	Problema	Peso	Penalidade
1	Coincidência total	Nenhum	1	0
2	Correspondência de sufixo Classe 1	Generalização	0.9	0.1
3	Correspondência de sufixo Classe 2	Especialização	0.7	0.3
4	Correspondência de prefixo Classe 1	Variação semântica	0.5	0.5
5	Correspondência de prefixo Classe 2	Variação semântica	0.5	0.5
6	Correspondência substring	Pode ter variação semântica	0.5	0.5

Tabela 4: Pesos e penalidades por classe

A modalidade de Sufixo Classe 1 tem a menor das penalidades já que os termos MeSH são uma especificidade do extraído (o extraído é uma generalização do termo MeSH). A Classe 2 do sufixo é o caso contrário, ou seja, o termo MeSH é uma generalização do extraído, o que faz com que a penalidade também seja baixa, mas levando em consideração que está sendo mais específico que o termo sugerido pelos autores.

Quando a penalidade é baixa, o peso é alto. As prioridades definem a preferência de uma palavra extraída para um termo MeSH. Supondo que existem duas palavras extraídas com correspondência para um termo, a preferência vai ser daquela que teve maior prioridade.

Já as modalidades de prefixo podem gerar diferenças semânticas. Para a Classe 1, onde a palavra-chave pode introduzir mudanças semânticas, a penalidade é maior que a anterior

e atribui-se um peso e penalidade média, assim também para o caso da modalidade Prefixo Classe 2. Com essas considerações, é possível definir uma ordem parcial que reflita o grau de confiança ao calcular o *recall*, onde 1 significa correspondência total e 0 incompatibilidade total, para aplicar as métricas tradicionais de RI para uma avaliação justa.

Para avaliar a efetividade dos resultados foram utilizadas as duas métricas mais bem aceitas na comunidade de RI: *precision* e *recall*, além do *F1-measure*.

O número de palavras-chave corretamente extraídas na avaliação justa é calculado como o somatório dos pesos. Assim se temos duas correspondências exatas (1) e uma de Sufixo Classe 1 (0.9) e uma de Prefixo Classe 2 (0.5), o número de correspondências corretas é 3.4. Cada correspondência fica entre 0 e 1. A avaliação aconteceu para cada uma das etapas do algoritmo, focando, principalmente, nas predições da etapa de conceitualização local.

4.2.2 Discussão dos resultados do *Extraction Engine*

A discussão dos resultados é feita para cada conjunto de dados. Na Tabela 5 são mostrados alguns dados, relacionados à execução do algoritmo sobre os resumos analisados para cada *dataset*. Para os 2.773 documentos analisados no primeiro *dataset*, o número total de predições foi 24.149 e, na média, oito predições foram extraídas por resumo. Pode-se perceber que o número de predições é 3 vezes maior que o número de termos MeSH, o que irá acarretar um valor menor no *precision*.

	Dataset 1	**Dataset 2**
Número total de artigos analisados	2773	2474
Número total de predições	24149	18243
Número máximo de predições por resumo	53	27
Número mínimo de predições por resumo	1	1
Número médio de predições por resumo	8	7
Número total de termos MESH	8062	6910
Número máximo de termos MESH por resumo	14	12
Número mínimo de termos MESH por resumo	1	1
Número médio de termos MESH por resumo	3	3
Tempo de processamento médio por artigo	13.1 ms	10.3 ms
Tempo de processamento	24 secs.	19 secs.

Tabela 5: Dados da execução por *dataset*

Nas tabelas a seguir, as predições correspondem aos conceitos extraídos localmente para cada artigo. A Tabela 6 mostra a distribuição dos resultados para a avaliação justa,

já que ela representa a melhor avaliação dos resultados. A distribuição representa o número de palavras-chave extraídas em cada classe de correspondência. A correlação é o coeficiente de correlação para determinar a relação entre as palavras-chave extraídas distribuídas na classe e os termos MeSH. A razão de predição mostra a relação entre a distribuição por classes das palavras extraídas e o número total de predições.

Tipo	**Distribuição (Predições por classe)**		**Correlação (Distribuição/total de predições)**		**Razão de predição (Distribuição/total de termos MeSH)**	
	dataset 1	*dataset* 2	*dataset* 1	*dataset* 2	*dataset* 1	*dataset* 2
Coincidência total	316	204	0,013	0,011	0,039	0,03
Correspondência de sufixo Classe 1	12	3	0,000	0,000	0,001	0,000
Correspondência de sufixo Classe 2	1329	919	0,055	0,05	0,165	0,133
Correspondência de prefixo Classe 1	14	19	0,001	0,001	0,002	0,003
Correspondência de prefixo Classe 2	695	760	0,029	0,042	0,086	0,11
Correspondência *substring*	1144	1593	0,047	0,087	0,142	0,231
TOTAL de predições	24149	18243	**Total de termos MeSH**		8062	6910

Tabela 6: *Dataset* 1, 2 - Distribuição das correspondências de predições

Pode-se observar que a maior parte das correspondências ocorre por sufixo de classe 2 e por correspondência por *substring*. Isso implica que a maioria das palavras-chave geradas são ou uma especialização do termo MeSH, ou fazem parte deste termo. As predições que possuem mais palavras do que um termo MeSH não são identificadas. Isto pode ser devido a que os termos MeSH vêm de um vocabulário controlado, o que limita o poder de comparação com as predições. A correlação e a razão de predição também se vêm afetadas pelo número grande de predições e poucos termos MeSH no *dataset*.

Os resultados obtidos consideram (1) a avaliação exata, onde todos os casos parciais são considerados negativos e (2) a avaliação relaxada onde se analisam as correspondências

parciais como casos positivos e (3) a avaliação justa onde se analisam as correspondências parciais segundo seu peso. Ao executar o *Extraction Engine*, foram calculados os valores das métricas de *precision*, *recall* e *F1-measure* para cada *dataset*. Esses resultados são apresentados na Tabela 7.

	Dataset 1			**Dataset 2**		
Avaliação	*Precision*	*Recall*	*F1-measure*	*Precision*	*Recall*	*F1-measure*
Exata	0,035	0,396	0,064	0,03	0,302	0,054
Justa	0,064	0,729	0,118	0,067	0,675	0,122
Relaxada	0,084	0,952	0,155	0,095	0,962	0,174

Tabela 7: Resultados das três avaliações por dataset

Dados estes valores, observa-se que o valor do *precision* é baixo. Isto é explicado ao analisar o cálculo feito utilizando como denominador o conjunto total de predições. A Tabela 5 mostra que o número médio de termos MeSH por resumo é três, sendo que o número médio de predições é de sete a oito termos por resumo. O gráfico da Figura 11 revela que o número de predições aumenta em relação ao tamanho do resumo. Por outro lado, o número de termos MeSH permanece estável independentemente do tamanho do resumo.

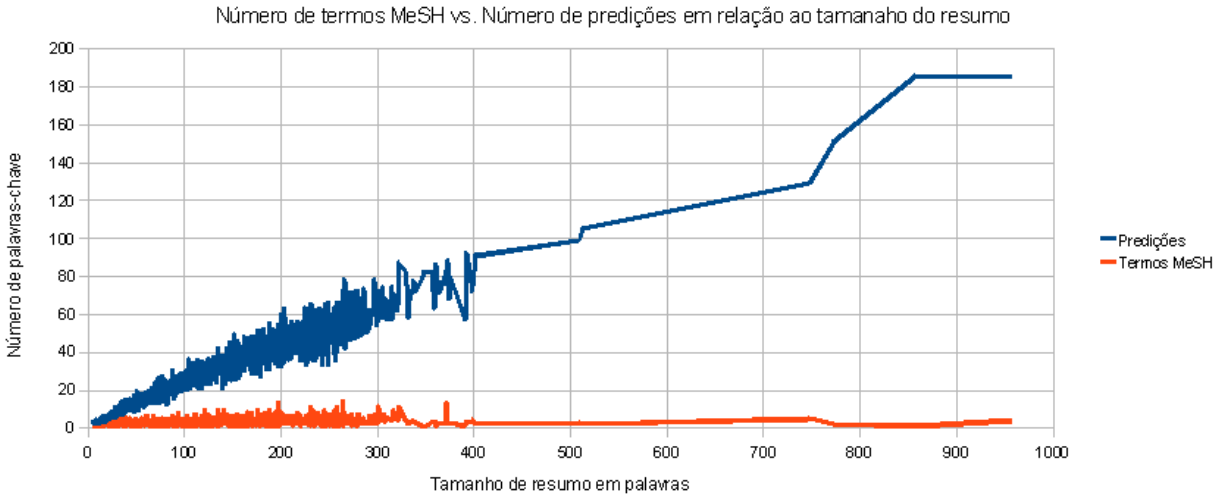

Figura 11: Número de termos MeSH vs. número de predições em relação ao tamanho em palavras do resumo (para *Dataset* 1)

A Tabela 8 mostra as palavras-chave que formam o conjunto global e o número de artigos pertencentes ao conjunto dos documentos recuperados que são indexados pela palavra-chave, assim como a porcentagem que representa dentro do conjunto de documentos.

Pela tabela acima pode-se observar que cada palavra chave global representa um

	Palavras-chave globais	**Num. de artigos**	**% de representação**	**Num. de artigos representados/total de artigos**
Dataset 1	*mycobacterium tuberculosis*	173	6,2	0,062
	ifn - gamma	74	2,6	0,027
	t cell	55	1,9	0,02
	tuberculosis infection	55	1,9	0,02
	drug resistant	52	1,8	0,019
Dataset 2	*pandemic influenza a*	181	7,3	0,073
	influenza a	168	6,7	0,068
	influenza virus	98	3,9	0,04
	season influenza	57	2,3	0,023
	acute respiratory distress syndrome	36	1,4	0,015

Tabela 8: Representação do conjunto de palavras-chave global por dataset

número reduzido de artigos comparado com a total de artigos retornado, o que facilitaria a busca por temas mais específicos.

4.3 Resultados *BioSearch Refinement*

Uma das características mais importantes em um sistema *Web* é a sua rapidez, para que o usuário possa ter uma navegação confortável. O *BioSearch Refinement* sendo um sistema deste tipo, deve funcionar em um tempo aceitável para um usuário de *Internet*. Para isto, foi feita uma analise da eficiência do sistema como um todo em relação ao tempo de execução.

Os testes de desempenho foram executados em um computador com processador Intel Core 2 Duo, 2.13 Ghz e 3GB de RAM. Como visto na Figura 8, observa-se que o algoritmo *Extraction Engine* está dividido em diversas etapas. A primeira etapa, que é a obtenção dos artigos do servidor PubMed, só depende das condições do próprio servidor e da conexão do usuário, logo o tempo de execução desta parte independe do sistema. O tempo levado em média para obter 1000 artigos do servidor do PubMed foi de 58 segundos. Notou-se que a maior parte do tempo, não é a execução do algoritmo propriamente dito, mas sim a obtenção dos artigos do servidor PubMed.

Foram medidos os tempos de carregamento de página no NCBI e no *BioSearch Refinement* para verificar o quão próxima é a velocidade de carregamento dos sistemas. Como

o banco de dados PubMed é local ao NCBI, o tempo de resposta será sempre mais rápido do que pela utilização do E-Utils, mas como mostrado na Tabela 9 o tempo do *BioSearch Refinement* é em média 50% maior do que do NCBI.

Termo de busca	**Tempo (segundos)**	
	NCBI	BioSearch Refinement
mycobacterium tuberculosis	3,4	6
whey protein effects	2,7	6,2
influenza h1n1 virus	3,3	5,3

Tabela 9: Tempo de carregamento de página NCBI e BioSearch Refinement

Outro fator que influencia no tempo de download é o fato de que o NCBI não carrega os artigos por completo para a exibição, ele utiliza o E-Summary que carrega menos atributos do que o E-Fetch. O *BioSearch Refinement* utiliza o E-Fetch, pois são necessários os resumos dos artigos para serem utilizados no *Extraction Engine*. A seguir são mostrados os resultdos obtidos para os tempos de execução de diversas etapas do *Extraction Engine* que impactam no desempenho do *BioSearch Refinement*.

Os tempos de execução exibidos na Tabela 10 abaixo são acumulativos das etapas, desde a tokenização de sentenças, PoST, *noun chunking* até a extração global de conceitos ou sumarização. Foram processados 50, 100, 200, 500, 1000 e 2000 artigos de ambos *datasets* com 100% de extração local de conceitos.

Pode-se observar que o tempo total para o processamento de 50 até 100 artigos é em torno de 1 segundo, um usuário que solicita a geração de palavras-chave geralmente navega nesta faixa de artigos, ou seja, de 2 a 4 páginas.

Artigos processados	**Tempo de execução (milissegundos)**			
Dataset 1	***Part of speech tagging***	***Noun chunking***	**Conceitualização local**	**Conceitualização global**
50	866	898	883	1043
100	1209	1275	1214	1350
200	2273	2217	2476	2549
500	5966	5497	6103	6231
1000	11391	11034	11971	12380
2000	21737	21693	23457	24376
Tempo médio por artigo	11,86	10,84	11,71	-
Dataset 2				
50	679	757	784	825
100	960	1037	1100	1146
200	2012	2082	1977	2034
500	4822	5096	5136	5561
1000	9361	9885	9572	10119
2000	18358	18616	18965	19753
Tempo médio por artigo	9,17	9,38	9,48	-

Tabela 10: Tempo de execução das etapas do algoritmo

5 *Conclusões*

Todos os objetivos definidos no início do projeto foram atingidos, resultando em um sistema funcional e eficiente. A seguir são dadas as conclusões finais sobre o sistema e suas partes, como também, são apresentadas as contribuições e trabalhos futuros.

5.1 *PubMed Dataset*

O *PubMed Dataset* se mostrou uma ótima ferramenta para a criação de conjunto de dados de teste, facilitando em grande parte o processo de avaliação do algoritmo de extração. O módulo de *download* que realiza a conexão com os servidores do NCBI foi projetado de maneira que seja flexível, possibilitando a adição de novos parâmetros com algumas linhas de código.

O tempo levado para carregar e escrever os *datasets* para a memória é de alguns segundos, sendo muito útil em baterias de teste. Já o tempo levado para obter os artgios depende de diversos fatores independentes da biblioteca, como condição de carga do banco de dados do PubMed e velocidade de conexão do usuário.

Por ser código aberto, pode ser aprimorado pela comunidade de bioinformática para ser uma biblioteca ainda mais funcional. O projeto se encontra no repositório Git do GitHub no endereço `https://github.com/lassounski/PubMed-Dataset` ou em formato .jar no repositório Maven. A dependência e o repositório que devem ser adicionado podem ser vistos nas Listagens 3 e 4.

```
<dependency>
    <groupId>com.uenf</groupId>
    <artifactId>PubMedDataset</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

Listagem 3: Dependência Maven

```
<repository>
    <id>lassounski-snapshots</id>
    <url>https://github.com/lassounski/mvn-repo/raw/master/snapshots</url>
</repository>
```

Listagem 4: Repositório Maven

A utilização do *PubMed Dataset* no *BioSearch Refinement* se mostrou muito útil, já que os artigos exibidos no sistema são obtidos usando a ferramenta. O tempo levado pela aplicação para exibir as páginas de artigos é aceitável, sendo um pouco maior do que o tempo levado pelo próprio NCBI.

5.2 *Extraction Engine*

O algoritmo de extração de palavras-chave obteve bons resultados possuindo um *recall* em torno de 70%, ou seja, dos termos MeSH atribuídos aos artigos que existem no texto, 70% foram recuperados pelo algoritmo.

As palavras-chave exibidas para o usuário destacam os assuntos mais abordados pelos artigos e representam sub-conjuntos do original, que podem ajudar no processo de busca. Uma expansão no algoritmo poderia permitir que o usuário navegue pelas palavras-chave através de um modelo de hierarquia, onde palavras-chave mais gerais contem palavras-chave mais específicas.

Vale notar que o *Extraction Engine* pode ser usado para outras áreas de aplicação diferentes da biomedicina. O que tem de ser modificado para que o algoritmo se adapte à uma determinada área de conhecimento é o seu *part of speech tagger* que vai determinar as classes gramaticais das palavras.

5.3 *BioSearch Refinement*

O sistema desenvolvido possui uma *interface* enriquecida com componentes visuais animados que favorecem a usabilidade, a navegação por abas facilita a organização da informação do usuário. O tempo de resposta da aplicação é aceitável, mas pode ser melhorado com técnicas de otimização.

5.4 Contribuições

O resultado do trabalho desenvolvido é de código aberto e é disponibilizado para utilização e/ou modificação, melhoramentos podem ser submetidos para a origem e serão avaliadas. Tanto a biblioteca *PubMed Dataset* como o sistema *BioSearch Refinement* podem ser encontrados no GitHub em `https://github.com/lassounski/PubMed-Dataset` e `https://github.com/lassounski/BioSearch-Refinement`.

O *PubMed Dataset* foi apresentado e publicado no BIOINFORMATICS 2012 (*International Conference on Bioinformatics Models, Methods and Algorithms*). E o *BioSearch Refinement* foi aceito para publicação na Conferência IADIS Ibero Americana de 2011.

5.5 Trabalhos futuros

As perspectivas de trabalhos futuros por parte do *Extraction Engine* visam uma mudança na arquitetura do algoritmo para permitir a categorização de conceitos por hierarquias. Mesmo conceitos diferentes como "platelets flow", "platelet", "platelet plasma" e "platelets expression", possuem "platelet" como conceito mestre. Este conceito estaria na raiz da árvore como conceito pai e os outros conceitos seriam englobados nestes como filhos. Um determinado conceito poderia ser filho de n conceitos pais e poderia ter m filhos, a relevância de um conceito na árvore seria representada pelo seu número de partes, número de filhos e frequência de ocorrência das partes similares.

Por parte da *interface*, uma provável melhora na performance de navegação seria utilizar o E-Summary em vez do E-Fetch para a obtenção dos artigos de uma busca. Isto acarretaria uma demora no processo de obtenção de palavras-chave visto que os resumos dos artigos deveriam ser obtidos através do E-Fetch. É necessário um estudo para determinar o que é mais importante para o usuário.

Referências Bibliográficas

ANTONIOU, G.; HARMELEN, F. van. *A Semantic Web Primer*. Cambridge: MIT Press, 2004.

ASHBURNER, M. et al. Gene ontology: tool for the unification of biology. *Nature Genetics*, v. 25, 2000.

BAEZA-YATES, R.; RIBEIRO-NETO, B. *Modern Information Retrieval*. New York: ACM Press, 1999.

BERRY, M. W.; KOGAN, J. *Text Mining: Applications and Theory*. [S.l.]: Wiley, 2010.

BRILL, E. A simple rule-based part of speech tagger. In: *In Proceedings of the Third Conference on Applied Natural Language Processing*. Trento, Italy: [s.n.], 1992.

CHINCHOR, N. A.; SUNDHEIM, B. *Message Understanding Conference (MUC) Tests of Discourse Processing*. [S.l.], 1995.

CISCO. Visual networking index: Forecast and methodology, 2008–2013. *White Paper*, 2009.

DING, J. et al. Pubmedassistant: a biologist-friendly interface for enhanced pubmed search. *Bioinformatics*, v. 22, p. 378–380, 2006.

DIVOLI, A.; ATTWOOD, T. K. Bioie: extracting informative sentences from the biomedical literature. *Bioinformatics*, v. 21, n. 9, 2005.

DOMS, A.; SCHROEDER, M. Gopubmed: Exploring pubmed with the geneontology. *Nucleic Acid Research, 33 (Web Server Issue)*, p. 783–786, 2005.

EATON, A. D. Hubmed: a web-based biomedical literature search interface. *Nucleic Acids Res*, v. 34, p. 745–747, 2006.

FAGAN, J. L. The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *Journal of the American Society for Information Science*, v. 40, n. 2, p. 115–139, 1989.

FELDMAN, R.; SANGER, J. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. [S.l.]: Cambridge University Press, 2007.

FREITAG, D.; MCCALLUM, A. Information extraction with hmm structures learned by stochastic optimization. *Just Research*, 2000.

GONZALEZ, M.; LIMA, V. L. S. de; LIMA, J. V. de. Termos, relacionamentos e representatividade na indexação de texto para recuperação de informação. *Letras de Hoje*, 2006.

GRUBER, T. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 1993.

GUCKER, P. *Essential English grammar*. [S.l.]: Courier Dover Publications, 1966.

GöKER, A.; DAVIES, J. *Information Retrieval:Searching in the 21st Century*. [S.l.]: John Wiley and Sons, 2009.

HULTH, A. Improved automatic keyword extraction given more linguistic knowledge. Stockholm University, Sweden, 2003.

JACKSON, P.; MOULINIER, I. *Natural Language Processing for Online Applications: Text retrieval, extraction and categorization*. [S.l.]: John Benjamins Publishing Co, 2007.

JANNACH, D.; ZANKER, M.; FELFERNIG, A. *Recommender Systems: An Introduction*. [S.l.]: Cambridge University Press, 2010.

LIPSCOMB, C. E. Medical subject headings (mesh). *The Medical Library Association*, 2000.

LUI, Y. J. Extraction of significat phrases from text. *International Journal of Computer Science*, v. 2, n. 2, p. 101–109, 2007.

MARON, M. E.; KUHNS, J. *On relevance, probabilistic indexing and information retrieval*. Ohio, 1958.

MOENS, M.-F. *Automatic indexing and abstracting of document texts*. [S.l.]: Springer, 2000.

NCBI. *Entrez Programming Utilities Help*. National Center for Biotechnology Information (US), 2010. Disponível em: <http://www.ncbi.nlm.nih.gov/books/NBK25501/>. Acesso em: 16 nov. 2011.

ODLYZKO, A. M. Internet traffic growth: Sources and implications. *SPIE*, p. 1–15, 2003.

ORACLE. Java se 6 performance white paper. *White Paper*, 2010. Disponível em: <http://java.sun.com/performance/reference% -/whitepapers/6_performance-.html>. Acesso em: 16 nov. 2011.

PEREZ-IRATXETA, C.; BORK, P.; ANDRADE, M. A. Xplormed: a tool for exploring medline abstracts. *Computer Corner*, v. 26, n. 9, p. 573–575, 2001.

PORTER, M. F. An algorithm for suffix stripping. *Program*, v. 14, n. 3, p. 130–137, 1980.

ROSSE, C.; MEJINO, J. L. Subject medical headings (mesh). *Journal of biomedical informatics*, 2003.

SIADATY, M. S.; SHU, J.; KNAUS, W. A. Relemed: sentence-level search engine with relevance score for the medline database of biomedical articles. *BMC Med Inform Decis Mak*, v. 7, n. 1, 2007.

SMALHEISER, N. R.; ZHOU, W.; TORVIK, V. I. Anne o'tate: A tool to support user-driven summarization,drill-down and browsing of pubmed search results. *Journal of Biomedical Discovery and Collaboration*, v. 3, n. 2, 2008.

SMITH, L.; RINDFLESH, T.; WILBUR, W. J. Medpost: a part-of-speech tagger for biomedical text. *Bioinformatics*, v. 20, n. 14, 2004.

TIOBE. Tiobe programming community index for november 2011. *TIOBE Software*, 2011. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index-.html>. Acesso em: 16 nov. 2011.

TSAI, R. T.-H. et al. Various criteria in the evaluation of biomedical named entity recognition. *BMC Bioinformatics*, v. 7, n. 1, p. 92, 2006.

TURNEY, P. Learning to extract keyphrases from text. *NRC-CNRC, ERB-1057*, 1999.

TURPIN, A.; SCHOLER, F. User performance versus precision measures for simple search tasks. In: *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. [S.l.: s.n.], 2006.