Uma solução flexível para a etapa de pré-processamento em mineração de textos

ADRIANO K. YAMADA[1], MARIA F. MOURA[2], SÉRGIO A. B. CRUZ[3], ROBERTO H. HIGA[4]

Nº 12611

RESUMO

A mineração de textos é uma área de PD&I cujo objetivo é a busca por padrões, tendências e regularidades em documentos textuais. O primeiro e mais importante passo do processo de mineração de textos compreende um conjunto de procedimentos para leitura da coleção de documentos, identificação e a seleção dos atributos estatisticamente mais significantes para representar a coleção por meio de uma matriz atributo-valor. O objetivo deste trabalho é apresentar uma solução flexível e expansível para a tarefa de pré-processamento em mineração de textos, capaz de atender às necessidades de diferentes projetos de pesquisa envolvendo esta temática.

ABSTRACT

Text mining is a RD&I area whose aim is to find patterns, trends and regularities in textual documents. The first and most important step in text mining is formed by a set of procedures for reading a document collection, identifying and selecting of the statistically most important attributes for representing this collection as an atributte-value matrix. The aim of this work is to present a flexible and expandable solution for preprocessing in text mining, capable to provide the needs of different research projects involving this subject.

1 Introdução

A mineração de textos é uma área de PD&I cujo objetivo é a busca por padrões, tendências e regularidades em textos escritos em língua natural. Normalmente refere-se ao processo de extrair conhecimento interessante de textos não estruturados; tendo sido inspirada na mineração de dados e, por vezes, considerada uma subárea da mesma. A mineração de dados corresponde à extração de conhecimento implícito, previamente desconhecido e potencialmente útil, ou a

[1]Estagiário de gradução Embrapa Informática Agropecuária

[2]Colaboradora – Pesquisadora Embrapa Informática Agropecuária

[3]Colaborador – Pesquisador Embrapa Informática Agropecuária

[4]Orientador – Pesquisador Embrapa Informática Agropecuária

busca por relações e padrões globais existentes em bases de dados. Enquanto a mineração de dados trabalha sobre bases de dados pré-organizadas, a mineração de textos trabalha sobre dados inerentemente não estruturados (Weiss et al, 2005).

Dessa forma, para tornar uma coleção de textos tratável por ferramentas de inferência, estatística ou lógica, o primeiro passo é transformar a coleção de textos em uma representação matemática, isto é, em uma representação estruturada. A representação, geralmente utilizada, é uma matriz, cujas linhas correspondem aos documentos e as colunas aos atributos (dita matriz atributo-valor) – ou a transposta dessa matriz (dita representação espaço vetorial). Os atributos são as palavras, ou composições de palavras, estatisticamente mais importantes na coleção de textos, bem como se pode ter alguns atributos que correspondam a alguma pré-classificação dos textos. O grau de importância do atributo é representado como o valor da célula de cada atributo em cada documento, que pode ser, por exemplo, sua frequência de ocorrência no documento. Ao conjunto de procedimentos que lê a coleção de textos, identifica e seleciona os atributos mais importantes e, finalmente, cria a matriz de representação, dá-se o nome de pré-processamento. Essa é a etapa mais importante do processo de mineração de textos, pois da escolha e seleção da coleção de textos, atributos e ponderações de importância dos mesmos, depende o sucesso ou o fracasso do processo; uma vez que qualquer análise de dados depende da qualidade dos dados, seus atributos observados e as medidas obtidas.

Para realizar esses procedimentos existem algumas ferramentas de domínio público que tem sido utilizadas por grupos de PD&I com bastante sucesso. A maioria dessas ferramentas permite que se trabalhe com textos em várias línguas, considerando-os como *bag of words*, e aplicando filtros tais como: limitar os atributos àqueles entre uma frequência mínima e uma máxima, nos textos (cortes de Luhn - (Luhn, 1958) e/ou na coleção de textos (cortes de Salton, (Salton et al., 1975)); eliminar uma lista pré-determinada de palavras (*stopwords*); remover inflexões de palavras (ferramentas PreText (Soares et al, 2008) e Lucene (McCandless et al., 2010), por exemplo), nesse caso, “intelig” corresponderia às palavras “inteligência” e “inteligente”; reconhecer expressões regulares nos textos (ferramenta NSP (Banerjee e Pedersen, 2003)); permitir desfazer-se de caracteres especiais, números, ou *tags* de linguagens de marcação. Porém essas ferramentas não apresentam todas essa funcionalidades, não geram um único formato de saída, deixam a desejar a presença de filtros específicos e, muitas vezes não são facilmente integráveis a outros sistemas. Ainda, os softwares R (R Development Core Team, 2011), Weka (Hall et al.,

2009) e similares, possuem pacotes integráveis, mas quase todos contemplam apenas a língua inglesa. Além disso, várias ferramentas são cedidas por universidades, como resultado de trabalhos de pesquisa, que, pela sua natureza, costumam não ser tão robustas e ter seu desenvolvimento descontinuado.

Devido a essa limitações, decidiu-se criar uma solução flexível, facilmente expansível, facilmente integrável a outros sistemas, que permita cobrir as necessidades de pré-processamento de textos em várias línguas e, que mantenha uma interface com outras ferramentas capazes de realizar uma análise sintática em um texto e gerar índices de frequência; além disso, essa solução deve permitir exportar os resultados em vários formatos utilizados por ferramentas de inferência, que também possam ser expandidos. Embora este objetivo seja um pouco ambicioso, pretende-se estimular um desenvolvimento colaborativo, num futuro próximo, disponibilizando a solução obtida como uma biblioteca de classes, sob licença GLP - *General Public License* (Licença Pública Geral), em um repositório de domínio público, por exemplo, o AgroLivre(http://www.agrolivre.gov.br/).

A solução desenvolvida é descrita no item material e métodos, bem como todas as soluções tecnológicas envolvidas. Em resultados e discussões são apresentados e discutidos alguns casos de uso da solução proposta. E, junto às conclusões o que se espera desenvolver em trabalhos futuros.

2 Material e Métodos

Nesta seção as técnicas de pré-processamento de textos mais utilizadas são brevemente descritas, pois são as previstas na primeira versão da eTMLib. A seguir descreve-se a arquitetura proposta para a biblioteca e sua relação com as tarefas de pré-processamento, bem como as soluções tecnológicas empregadas em sua implementação atual. Como a ferramenta Lucene apresenta muito das características desejadas para o desenvolvimento da eTMLib, apresenta-se uma breve descrição dos fatores que são determinantes no caso de se trocar a Lucene por outra ferramenta de base. E, por fim, as possibilidades de uso da eTMLib em linha de comando também são apresentadas.

2.1 Métodos de pré-processamento estatístico de textos

A tarefa de preparação de dados é sempre dependente dos métodos de inferência que se pretenda utilizar; e, o pré-processamento é a preparação dos dados

no processo de mineração de textos. Nesta primeira versão da eTMLib houve a preocupação em disponibilizar apenas os métodos estatísticos mais utilizados em mineração de textos, que também são os de execução mais rápida, e que fornecem um representação tabular da coleção de textos. Essa representação dos dados foi a escolhida por satisfazer ao maior número de ferramentas de extração de padrões (classificadores, análise exploratória de dados, análise estatística multivariada, etc).

Para a representação tabular, necessitam-se identificar atributos e seus valores em cada documento, obtendo uma matriz do tipo:

	a_1	...	a_m	c
d_1	i_{11}	0	i_{m1}	c_1
...	0	0	0	0
d_n	i_{n1}	0	i_{nm}	c_n

Onde,

$\{d_1,..d_n\}$: corresponde à coleção de n textos em análise;

$\{a_1,..a_m\}$: corresponde aos m atributos considerados importantes para o processo de extração de padrões na coleção de n documentos;

$\{c_1,...,c_n\}$: corresponde ao valor da classe associada a cada documento. Por exemplo, se os documentos se referissem a notícias de jornal sobre {esportes, agricultura, economia,saúde}, cada documento estaria associado a uma dessas classificações;

$\{i_{11},...,i1_m, i_{21}, \ldots, i_{mn}\}$: cada i_{ij} corresponde ao valor de importância de cada atributo j em cada documento i. A importância é determinada de acordo com o que se deseja medir, pode ser frequência no documento, frequência na coleção, etc.

Os métodos estatísticos de pré-processamento previstos nesta versão são:

- **Sem uso algum de conhecimento externo:** supõe-se que não se sabe quais atributos devem ser procurados nos textos. Nesse caso, pode-se aplicar um tratamento básico aos textos, tal como desconsiderar *stopwords*, remover inflexões das palavras (*stemming*), desconsiderar números ou outros símbolos, compor palavras pela sequência em que aparecem no texto, etc.
- **Com uso de vocabulário controlado:** nesse caso, sabe-se de antemão quais atributos devem ser buscados nos textos e supõe-se que esta informação está disponível. Pode-se usar um conjunto de atributos pré-selecionados em algum processo anterior de pré-processamento, ou pré-processar uma ontologia ou um *thesaurus* para obter esses atributos.

- **Seleção de atributos:** a escolha dos atributos é realizada para a coleção, logo, a matriz gerada tende a ser esparsa, pois nem todos os atributos aparecem em todos os documento da coleção. Assim, os atributos precisam ser bem selecionados, para diminuir a questão de esparsividade e não sobrecarregar os métodos de extração de padrões no processamento de atributos potencialmente inúteis. Pode-se classificar a desconsideração de stopwords ou o uso de vocabulário controlado como seleção de atributos. Porém, na maioria das vezes está-se falando em cortes de atributos, como os propostos por Luhn, Salton, Pedersen, etc; que podem ser combinados entre si. Nessa versão foram implementados cortes por frequência do atributo em documentos e coleção e, também, a seleção de n-gramas proposta por Moura et al (2008).
- **Eliminação de redundâncias:** atributos redundantes podem ser encontrados de diversas formas. Nesta primeira implementação o único tipo de redundância considerado foi a repetição de ocorrências do mesmo atributo em diferentes n-gramas, como proposto por Moura et al (2008).
- **Sinônimos**: os sinônimos, para a eTMLib, estão sendo considerados como atributos que devem ser utilizados em lugar de alguns outros atributos que podem ter sido selecionados na coleção. Eles são trocados pelos atributos encontrados e sua frequência de ocorrência em cada documento é totalizada para todos só seus sinônimos.

Em expansões da biblioteca, poder-se-á fornecer outros métodos de extração e seleção de atributos, inclusive linguísticos ou híbridos, considerando contexto, etc.

2.2 A relação entre a eTMLib e a Lucene

Lucene é uma biblioteca de alta performance e escalável, escrita na linguagem Java para construção de aplicações de recuperação de informação, disponibilizada como código aberto pelo projeto Apache Lucene (http://lucene.apache.org/).

As funcionalidades de Lucene compreendem o ciclo completo de uma aplicação de recuperação de informação, desde a aquisição do conteúdo, a construção dos documentos e sua indexação até a utilização dos índices para realização e apresentação de buscas.

A eTMLib consiste em uma extensão de um conjunto de funcionalidades de Lucene para realização das tarefas associadas ao pré-processamento em mineração

de textos, compreendendo um conjunto de analisadores, utilizados para extrair palavras do texto, filtrar a pontuação e os acentos, proceder a normalização, remoção de palavras comuns (stopwords) e sua redução a sua raiz (stemming); e a funcionalidade de indexação relacionada para a extração de vetores de pares termo-frequência de ocorrência.

2.3 Arquitetura da eTMLib

A eTMLib é constituída por (i) um conjunto de extensões de analisadores e filtros de Lucene que, em conjunto, realizam a análise dos termos ou atributos presentes nos documentos analisados e contabilizam sua frequência de ocorrência; (ii) algoritmos para seleção do conjunto de atributos estatisticamente mais representativos do conjunto de documentos analisado (vocabulário) e (iii) um conjunto de funções para exportar esses dados na forma de uma matriz atributo-valor, em diferentes formatos apropriados para utilização com diversas outras ferramentas (CSV, libSVM (Chang e Lin, 2011) e Discover (Soares et al, 2008)). Suas funcionalidades compreendem aquelas mais comumente utilizadas para a realização de pré-processamento para mineração de textos:

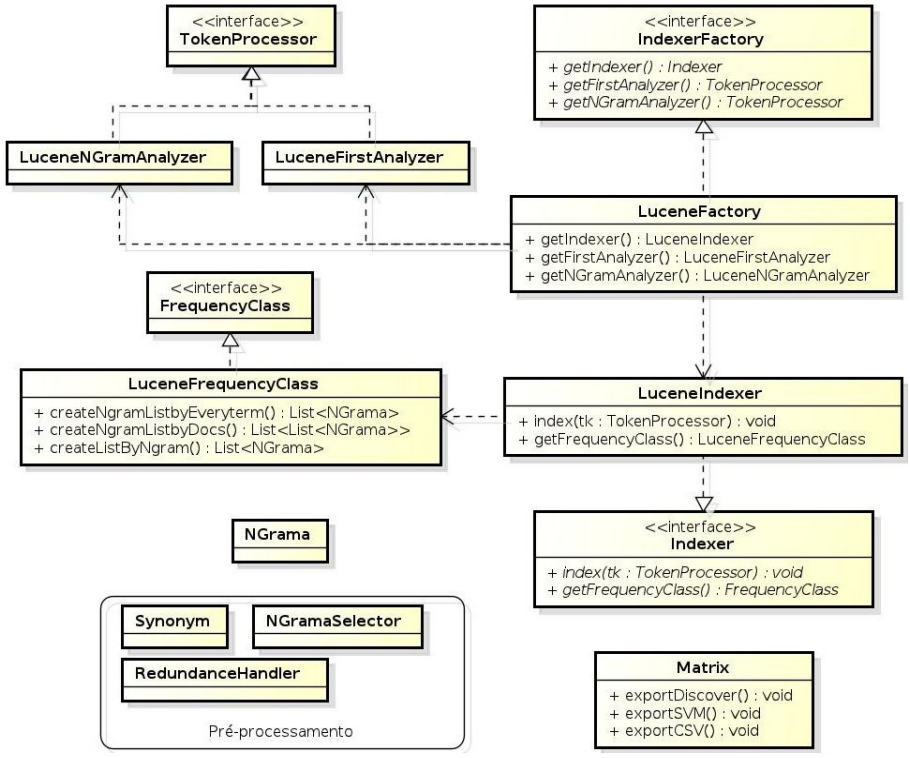

Figura 2 – Diagrama de Classes envolvidas na indexação

O código da aplicação cliente (usuária da biblioteca eTMLib) deve ser elaborado com base em interfaces que abstraem as funcionalidades envolvidas na indexação. Desta forma a aplicação cliente não sofrerá qualquer modificação caso a ferramenta de indexação Lucene seja substituída. A Figura 2 apresenta algumas das principais interfaces e classes implementadas na eTMLib envolvidas no procedimento de indexação.

A interface *IndexerFactory* define um conjunto de métodos de acordo com o padrão de projeto *Factory Method* (Gamma et al, 1995). Desta forma a implementação final para as interfaces *TokenProcessor*, *Indexer* e *FrequencyClass* depende da ferramenta escolhida para indexação. Neste trabalho estas interfaces são implementadas com base na ferramenta Lucene. As funcionalidades de filtragem são suportadas por classes que implementam a interface *TokenProcessor*. As classes *LuceneNGramAnalyzer* e *LuceneFirstAnalyzer* suportam as funcionalidades de pré-processamento e permitem a filtragem de palavras por *stemming* e remoção de *stopwords*. Além destas funcionalidades a classe *LuceneNGramAnalyzer* permite o tratamento de n-gramas. A classe *NGramaSelector possibilita o corte de n-gramas com base em critério proposto por Moura et al (2008). A remoção por redundância de atributos nos n-gramas é realizado pela classe RedundanceHandler.* A classe *Synonym* gerência atributos sinônimos, possibilitando o agrupamento e tratamento em conjunto destes atributos.

A indexação é realizada por classes que implementam a interface *Indexer*. Após a realização da indexação a classificação dos atributos por frequência pode ser acessada a partir das classes que implementam a interface *FrequencyClass*. A classe *LuceneFrequencyClass* implementa esta interface e fornece os dados necessários para a construção da matriz de documentos x atributos. A partir deste ponto a matriz pode ser exportada para formatos de arquivos que possibilitem o seu processamento por ferramentas para identificação de padrões.

Além destas classes e interfaces envolvidas diretamente no processo de indexação a biblioteca é complementada por um conjunto de classes que possibilitam o ajuste dos parâmetros dos processos de pré-processamento, filtragem e análise dos resultados.

A eTMLib está disponível tanto na forma de biblioteca para incorporação a aplicações standalone quanto na forma de um comando executável.

2.4 Modo de uso

A eTMLib poder ser utilizada tanto como uma biblioteca para incorporação em uma aplicação específica quanto de forma independente por meio de linha de comando. Ela pode ser executada da seguinte forma:

java -jar eTMLib.jar <opções> <diretório do corpus>

onde:

-i <língua(s)>: 1 – inglês, 2 -português. Mais de uma língua pode ser especificada, ex: -i 1,2. Default: 1.

-f <filtro(s) utilizados>: 1 – stopwords+corte, 2 – stopwords+stemming+corte, 3 – sotpwords+stemming+corte+n-gramas. Especifica a sequencia de tarefas de pré-processamento a ser realizada. Default = 1.

-l <min>:<max>: Especifica os valores mínimo e máximo a serem utilizados na realização de cortes de Luhn. Default: 1:1000000.

-d <min>:<max>: Especifica os valores de df mínimo e máximo a serem utilizados. Default: 1:|D|, onde |D| é....

-n <min>:<max>: 1 – apenas unigramas, 2 – unigramas+bigramas, 3 – unigramas+bigramas+trigramas Especifica o conjunto de n-gramas desejados. Default: 1:3.

-t <índices>: 1: TF, 2 – DF, 3 - TF-IDF, 4 – Boolean. Especifica a semântica dos valores na matriz atributo-valor de saída. Default: 1.

-g <group file>: Se especificado, produz uma matriz atributo-valor onde as linhas correspondem a uma sumarização dos valores encontrados no pré-processamento por documento. <group file> deve ser um arquivo textual contendo em cada linha um conjunto de identificações de documentos que deverão ser assumidos como definindo grupos. Cada linha é precedida pela especificação do grupo. A especificação do grupo e da lista de documentos são separados por (:) e os documentos por (,). Ex: grupo1: doc1, doc2, doc3, …

-m <sumarizacao>: 1 – mediana, 2 – média. Se a opção -g é especificada, esta opção especifica a forma de sumarização das frequências. Default: 1.

-s <arquivo de sinônimos>: Se especificado, substitui a ocorrência de termos sinônimos antes de gerar a matriz atributo-valor. <arquivo de sinônimos> deve ser um arquivo textual contendo em cada linha um conjunto de identificações de termos e seus sinônimos. Cada linha é precedida da especificação do termo e a lista de seus sinônimos. A especificação do termos e a lista de sinônimos são separados por (:) e os sinônimos por (,).

-v <arquivo de VC de entrada>: Lista de termos (vocabulário controlado) a ser processada. O arquivo de VC deve ser um arquivo textual com cada termo em uma linha.

-z <arquivo de stopwords de entrada>: Arquivo textual contendo lista de termos stopwords, um termo por linha.

-p <prefixo para especificação do(s) arquivo(s) de saída>: Especifica um prefixo para

composição dos nomes do(s) arquivo(s) de saída. Default: out.

-o <formato(s)>: 1 – CSV, 2 – Discover, 3 – SVMLib. Esta opção especifica o formato de saída para a matriz atributo-valor. Mais de um formato de arquivo de saída pode ser especificado. Default: 1.

-w <arquivo de saída VC>: Grava arquivo de vocabulário controlado obtido pelo processamento.

-y <arquivo de saída de stopwords>: Gera arquivo de stopwords da coleção.

-a: Gera arquivo de sinônimos a partir de arquivo de vocabulário controlado especificado na opção -v, sendo que o nome do arquivo gerado é sinymoOnt.txt.

3 RESULTADOS E DISCUSSÃO

A eTMLib tem sido utilizada no projeto Prosgen (código SEG 03.09.01.025.00.00), que objetiva a análise e priorização de genes candidatos utilizando informações oriundas do banco de dados de artigos científicos denominado Pubmed (NCBIa, 2011) e técnicas de mineração de textos na para apoiar a interpretação biológica de genes candidatos. A estratégia de priorização proposta baseia-se no trabalho desenvolvido por Yu et al. (2010), cujo primeiro passo é a construção de uma coleção de documentos, em formato XML, contendo descrições funcionais de genes encontrados no genoma do organismo em estudo. Para cada gene é criado um documento contendo os títulos e resumos de artigos relacionados, recuperados, via *ftp*, do sítio (NCBIb, 2011), utilizando o conjunto de programas *Entrez Programming Utilities*. O passo seguinte consiste em realizar o pré-processamento desses documentos, com o objetivo de obter uma representação matricial em um espaço n-dimensional, formado pelo conjunto de atributos identificados dos próprios textos, e então aplicar algum processo de extração de padrões.

Como exemplo de pré-processamento, neste artigo, utilizamos uma coleção de artigos relacionados a genes do organismo B.taurus (*Bos taurus*), obtido do Pubmed (NCBIa, 2011), com 7414 artigos em inglês. Para obter os atributos e, consequentemente, a representação matricial foram utilizados quatro tipos de filtragem. A primeira filtragem consiste em eliminar apenas as *stopwords* da língua inglesa (artigos, interjeições, pronomes, etc). O segundo filtro foi definido por Salton em 1979 (Salton et al., 1979), onde são mantidos apenas os atributos que ocorrem em 1 a 10 % dos documentos da coleção de textos. Com estes dois filtros ter-se-iam 55855 atributos para representar os 7414 artigos; como ilustrado na Tabela 1. Para o terceiro filtro é necessário que se tenha um vocabulário controlado, extraído de uma

ontologia ou *thesaurus*, e que seja pré-processado antes dos textos para constituir uma base de vocabulário controlado. Quando se tem o vocabulário controlado pode-se interseccionar o conjunto de atributos encontrados na coleção com o conjunto do vocabulário controlado e então obter atributos, de fato, representativos da coleção de textos. Veja na Tabela 1 que após esse filtro o número de atributos caiu para 1695, ou seja, 3% do total inicial e, coma certeza de que são atributos que interessam na coleção de textos. Por fim, o último filtro foi aplicado aos resultados sem vocabulário controlado e com vocabulário controlado; nos dois casos o número de atributos é reduzido em apenas 32; porém proporcionalmente o vocabulário controlado tem uma melhor representatividade. Este último filtro é um tratamento de sinônimos, que ajuda a reduzir a dimensão final dos atributos bem como a tratar redundâncias semânticas sem usar algoritmos de processamento de língua natural. Os sinônimos são definidos como uma lista de atributos com um único equivalente, que substitui todas as ocorrências dos demais atributos presentes no conjunto de atributos encontrado. Tanto o vocabulário controlado como os sinônimos foram extraídos da Gene Ontology – GO (2012), que possui 212322 termos e 22700 sinônimos.

Filtros aplicados	Sem Vocabulário Controlado	Vocabulário da *GO*
Stopwords da língua inglesa	1050760	1050760
74 ≤ df ≤ 741	55855	55855
vocabulário controlado	55855	1695
sinônimos	55823	1663

Tabela 1: Número de atributos encontrados para a coleção *BosTaurus*, com 7414 artigos em inglês. A GO possui 212322 termos e 22700 sinônimos.

4 CONCLUSÃO

Os objetivos propostos foram atingidos, pois tem-se uma arquitetura flexível, que permite a evolução funcional da biblioteca e uma integração a novas aplicações muito simples, por meio da biblioteca, do reuso das classes ou por linha de comando. O fato de ser executável por linha de comando permite que ela possa ser utilizada por ferramentas de *workflow* sem dificuldades. Exportar vários formatos de dados permite que seus resultados possam ser utilizados por várias ferramentas de indução estatística ou lógica. O exemplo tabulado mostra que a implementação da eTMLib está robusta, em relação ao dimensionamento da coleção e do conjunto de atributos, bem como sua flexibilidade em tratar diferentes parametrizações de execução.

O próximo passo é disponibilizá-la em domínio público, sob licença GPL - GNU *General Public License*, para se tenha desenvolvimento colaborativo de outros grupos que trabalham com mineração de textos.

5 REFERÊNCIAS

- BANERJEE, S. e PEDERSEN, T. (2003). The design, implementation, and use of the ngram statistics package. Em Proceedings of 4th Conference on Intelligent Text Processing and Computational Linguistics - CICLing, páginas 370–381.
- CHANG, CC.; LIN, CJ. (2011) LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
- EO. Environment Ontology. In: http://www.gramene.org/, acessado em maio de 2012
- GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. Design patterns: elements of reusable object-oriented software. Reading: Addison-Wesley Publishing Company, 1995. 395p. (Addison-Wesley Professional Computing Series).
- GO; The Gene Ontology. In: www.geneontology.org, acessado em maio de 2012.
- acessado em maio de 2011.
- Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten IH. (2009) The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.
- KONCHADY, M. Building Search Applications – Lucene, LingPipe and Gate. Mustru Publishing, Virginia. 2008.
- LUHN, H. P. (1958). The automatic creation of literature abstracts. Journal of Research and Development - IBM, 2(2):159–165.
- McCANDLESS, M.; HATCHER, E.; GOSPODNETIĆ, Lucene in Action. Second Edition. Manning Publications Co. 2010.
- MOURA, M. F.; NOGUEIRA, B. M.; CONRADO, M. S.; SANTOS, F. F.; Rezende, S. O. Making Good Choices of Non-Redundant N-gram Words. In: INTERNATIONAL WORKSHOP ON DATA MINING AND ARTIFICIAL INTELLIGENCE, 1.; INTERNATIONAL CONFERENCE ON COMPUTER AND

INFORMATION TECHNOLOGY, 11., 2008, Khulna, Bangladesh. Khulna, Bangladesh: IEEE Faculty of Electrical and Electronic Engineering and Khulna University of Engineering and Technology, 2008. v. 1. p. 64-71.

- NCBIa; PubMed. In: http://www.ncbi.nlm.nih.gov/sites/entrez?db=PubMed&itool=toolbar, acessado em novembro de 2011.
- NCBIc; Entrez Gene. In: ftp://ftp.ncbi.nih.gov/gene/, acessado em novembro de 2011.
- R DEVELOPMENT CORE TEAM (2011). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.
- SALTON, G., YANG, C. S., e YU, C. T. (1975). A theory of term importance in automatic text analysis. Journal of the American Association Science, 1(26):33–44.
- SOARES, M. V. B., PRATI, R. C. MONARD, M. C. PreTexT II: Descrição da Reestruturação da Ferramenta de Pré-Processamento de Textos. Relatório Técnico, ICMC/USP, São Carlos, n. 333, 2008. In: http://www.icmc.usp.br/~biblio/BIBLIOTECA/rel_tec/RT_333.pdf.
- WEISS, S. M., INDURKHYA, N., ZHANG, T., e DAMERAU, F. J. (2005). Text Mining – Predictive Methods for Analizing Unstructured Information. Springer Science+Business Media, Inc.
- YU, S.; TRANCHEVENT, L.C.; MOOR, B.D.; MOREAU, Y. Gene priorization and clustering by multi-view text mining, v. 11:28, 2010, http://biomedcentral.com/1471-2105/11/28.