Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Descrição do ambiente computacional TP-DISCOVER para mapear informações não estruturadas em uma tabela atributo-valor

Daniel de Faveri Honorato
Maria Carolina Monard

Nº 318

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Fevereiro/2008

Descrição do ambiente computacional TP-DISCOVER para mapear informações não estruturadas em uma tabela atributo-valor*

Daniel de Faveri Honorato
Maria Carolina Monard

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
Laboratório de Inteligência Computacional
Caixa Postal 668, 13560-970 - São Carlos, SP, Brasil
e-mail: {dfaverih@gmail.com, mcmonard@icmc.usp.br}

Resumo

Freqüentemente, o processo de Mineração de Textos, o qual auxilia na extração de conhecimento de conjunto de documentos textuais, requer que os dados encontrem-se armazenados em uma tabela atributo-valor. Neste trabalho é apresentada uma descrição detalhada e é exemplificado o funcionamento do ambiente computacional TP-DISCOVER, o qual foi desenvolvido para auxiliar na aplicação de uma metodologia de mapeamento de informações não estruturadas em informações estruturadas representadas em uma tabela atributo-valor.

Palavras-Chave: Pré-processamento de Texto, Aprendizado de Máquina, Mineração de Textos

*Trabalho desenvolvido com apoio do CNPq e da Fundação Parque Tecnológico Itaipu - FPTI.

Sumário

Sumário 3
Lista de Figuras 5
Lista de Abreviaturas 2

1 Introdução **2**

2 Ambiente Computacional TP-DISCOVER **4**
2.1 Pré-processamento 10
2.1.1 Padronização 11
2.1.2 Mapear Substituições 11
2.1.3 Corretor de Palavras 12
2.1.4 Lematização 13
2.1.5 Estatísticas do CFU 13
2.2 Extração de Terminologia 14
2.2.1 Correção de Etiquetas 15
2.2.2 Geração de listas de n-gramas 17
2.3 Identificação de Atributos e Construção da Tabela Atributo-valor 17

3 Biblioteca de Classes **22**
3.1 Interface com o Usuário 22
3.2 Diagramas de classes 23
3.2.1 Padronização 23
3.2.2 Gerar Termos Candidatos 24
3.2.3 Gerar Árvores 25
3.2.4 Dicionário 26

Referências **29**

A Descrição das Classes Implementadas na Linguagem Java **30**

B Descrição das Classes Implementadas na Linguagem Perl **34**

C Descrição dos *Scripts* Implementados na Linguagem Perl **36**

Lista de Figuras

2.1 Metodologia proposta 4
2.2 Módulos do ambiente 6
2.3 Arquitetura de interação entre os módulos do ambiente 6
2.4 Interface de abertura 7
2.5 Interface principal 8
2.6 Interface para a construção de um novo projeto 8
2.7 Arquivos correspondentes a cada documento do conjunto de documentos para processar 9
2.8 Estrutura de pastas geradas no novo projeto 9
2.9 Interface de padronização 11
2.10 Interface para aplicação de substituições 12
2.11 Interface do corretor de palavras 13
2.12 Interface do lematizador 13
2.13 Interface de estatísticas do CFU 14
2.14 Interface de etiquetação do CFU 15
2.15 Interface de correção de etiquetas 15
2.16 Interface de definição de máscaras 16
2.17 Interface de extração de terminologia 16
2.18 Interface de resultado da extração de terminologia 16
2.19 Interface de geração das listas de n-gramas 17
2.20 Interface de geração da lista de termos candidatos 18
2.21 Interface de definição de termos raíz 18
2.22 Interface de visibilização das árvores 19
2.23 Árvore gerada a partir do termo mucosa terço 19
2.24 Interface de construção manual do dicionário 20
2.25 Interface de construção automática do dicionário 21
2.26 Exemplo da tabela gerada pelo ambiente computacional 21

3.1 Arquitetura da interface com o usuário 22
3.2 Interface de padronização 24
3.3 Diagrama de classes de padronização 24
3.4 Diagrama de classes para gerar termos candidatos (Java) 25
3.5 Diagrama de classes para gerar termos candidatos (Perl) 25
3.6 Diagrama de classes para gerar árvores 26
3.7 Diagrama de classes para gerar árvores (Perl) 26
3.8 Diagrama de cltasses para construção da tabela 26
3.9 Diagrama de classes para construção da tabela (Perl) 26

Lista de Abreviaturas

CFU Conjunto de Frases Únicas

GUI *Graphic User Interface*

LABI *Laboratório de Bioinformática*

LABIC *Laboratório de Inteligência Computacional*

MD Mineração de Textos

NSP *N-gram Statistic Package*

TP-DISCOVER *Terms Patterns Discover*

XML *Extended Markup Language*

Capítulo 1

Introdução

O desenvolvimento e utilização de tecnologias para a aquisição e armazenamento de dados, nas mais diversas áreas do conhecimento, têm permitido o acúmulo de dados em uma velocidade maior que a capacidade humana possui para processá-los. Esses dados podem estar representados em diferentes formatos e, um dos formatos bastante utilizados é o formato textual não estruturado. Para que esses dados textuais brutos possam tornar-se úteis, é necessário que eles sejam representados de maneira apropriada, que sejam processados para extrair padrões tal que um modelo que represente o conhecimento embutido nesses dados seja construído. Uma das maneiras de alcançar esse objetivo é por meio da realização do processo de Mineração de Textos — MT — (Weiss et al., 2005). As tarefas de MT podem ser divididas em quatro fases: coleta de documentos, pré-processamento, extração de conhecimento e pós-processamento. Na fase de coleta de documentos são recuperados os documentos relevantes ao domínio da aplicação. A fase de pré-processamento é responsável por transformar os documentos para um formato adequado para serem submetidos aos algoritmos de extração de padrões. Uma das transfomações mais utilizadas para transformar textos em tabelas é a abordagem *bag-of-words*, a qual pode utilizar o formato de uma tabela atributo-valor, como representada na Tabela 1.1. Cada documento d_i é um exemplo da tabela e cada termo (palavra) t_j é um elemento do conjunto de atributos. Cada documento também pode ter associado à classe a qual pertence (C).

	t_1	t_2	$\dots$	t_M	C
d_1	a_{11}	a_{12}	$\dots$	a_{1M}	c_1
d_2	a_{21}	a_{22}	$\dots$	a_{2M}	c_2
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
d_N	a_{N1}	a_{N2}	$\dots$	a_{NM}	c_N

Tabela 1.1: Representação de documentos

Mais especificamente, na Tabela 1.1 estão representados N documentos (exemplos) compostos por M termos (atributos). Cada documento d_i é um vetor $d_i = (a_{i1}, a_{i2}, ...a_{iM})$, no qual o valor a_{ij} refere-se ao valor associado ao j-ésimo termo do documento i. O valor a_{ij}, do termo t_j no documento d_i, pode ser calculado utilizando diferentes medidas, tais como *boolean*, freqüência do termo no documento (*term frequency*) e freqüência do termo no documento em

relação ao número de vezes que aparece no conjunto de documentos.

Na fase de extração de conhecimento o objetivo é descobrir padrões úteis e desconhecidos presentes nos documentos. Por fim, é necessária a fase de pós-processamento para verificar se o objetivo foi alcançado ou se é necessário retornar a alguma fase anterior.

Conforme mencionado, o acúmulo de informações é cada vez maior nas diversas áreas do conhecimento. Essas informações podem estar representadas em diferentes formatos tais como tabelas no formato atributo-valor armazenadas em banco de dados ou em conjunto de textos com diferentes estruturações, tais como textos não estruturados, textos semi-estruturados e textos estruturados. Para que seja possível realizar a extração de padrões por meio da aplicação de técnicas de MT é necessário que as informações estejam representadas em um formato adequado. Geralmente, como mencionado, os algoritmos de MT realizam a extração de padrões a partir de uma representação atributo-valor, conforme foi apresentado na Tabela 1.1. Portanto, as informações que não estão nesse formato devem ser transformadas para o formato utilizado pelos algoritmos de MT.

Neste trabalho é apresentado o ambiente computacional TP-DISCOVER, o qual tem por objetivo implementar a metodologia de transformação de informações não estruturadas em informações estruturadas representadas em uma tabela atributo-valor. Essa metodologia é uma proposta do projeto de mestrado de Honorato (2008) e é descrita em maiores detalhes no relatório técnico de Honorato & Monard (2008).

Deve ser ressaltado que o ambiente computacional TP-DISCOVER faz parte de um projeto de pesquisa mais amplo, desenvolvido conjuntamente por pesquisadores do Laboratório de Bioinformática — LABI, da Universidade Estadual do Oeste do Paraná e o Laboratório de Inteligência Computacional — LABIC, da Universidade de São Paulo. Uma metodologia inicial foi desenvolvida no LABI (Honorato et al., 2005), na qual é necessária uma grande interação com especialista(s) do domínio. A metodologia implementada pelo TP-DISCOVER tem por objetivo diminuir, sempre que possível, a intervenção do(s) especialista(s) do domínio.

Este trabalho está organizado da seguinte maneira. No Capítulo 2 é descrita a arquitetura e é ilustrado o funcionamento do ambiente computacional e, no Capítulo 3 são descritos os principais diagramas de classes. Finalmente nos Apendices A, B e C são apresentadas as descrições das classes implementadas na linguagem Java, das classes implementadas na linguagem Perl e a descrição dos *scripts* implementados na linguagem Perl, respectivamente.

Capítulo 2

Ambiente Computacional TP-DISCOVER

O ambiente computacional *Terms Patterns Discover* — TP-DISCOVER — foi implementada para auxiliar na aplicação de uma metodologia desenvolvida para transformar informações médicas não estruturadas em informações estruturadas descritas em uma representação atributo-valor. A metodologia utiliza algoritmos híbridos da área de extração de terminologia para auxiliar na construção desses atributos. A metodologia, descrita em detalhes em Honorato & Monard (2008), é composta pelas seguintes cinco fases:

1. Pré-processamento;
2. Extração de terminologia;
3. Identificação de atributos;
4. Construção do dicionário; e
5. Construção da tabela atributo-valor, conforme ilustrado na Figura 2.1.

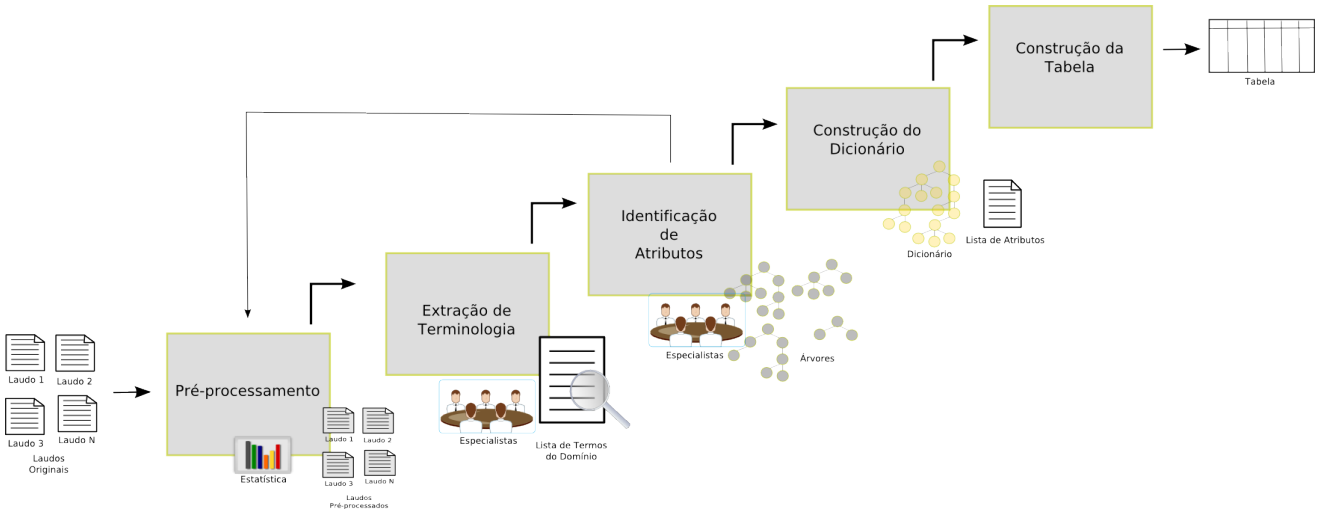

Figura 2.1: Metodologia proposta

Na fase de pré-processamento, o conjunto de documentos é preparado para a próxima fase, a de extração de terminologia (Pavel & Nolet, 2002), por meio de tarefas tais como toquenização, segmentação, construção de frases únicas, remoção de *stopwords*, lematização e etiquetação. Na fase de extração de terminologia são geradas, a partir da coleção de documentos etiquetados, listas de termos do domínio que possuem determinadas propriedades sintáticas. Os termos identificados são utilizados na terceira fase, na identificação de atributos, na qual, a partir de cada termo da lista de termos identificados são geradas árvores que mapeam o relacionamento dos termos identificados com os termos que aparecem no contexto (lado direito) desses termos. Nesses relacionamentos também são mapeados as freqüências que esses termos aparecem juntos. Depois de geradas as árvores são realizadas buscas sobre elas com o intuíto de identificar todos os ramos que possuem freqüência acima de um determinado limiar em relação ao número de documentos utilizados para gerar as árvores. Os ramos identificados são definidos como atributos que farão parte da tabela atributo-valor.

Depois de identificados os atributos, eles são mapeados em uma estrutura de dados do tipo árvore trie, que denominamos de dicionário de conhecimento, o qual é utilizado na última fase da metodologia, a de construção da tabela atributo-valor. Na construção da tabela são utilizados o conjunto de documentos, o dicionário e a lista de atributos identificados.

A tabela é constituída por atributos binários, ou seja, o valor dos atributos é zero ou um; ou presente ou ausente. O processo de preenchimento é realizado por meio de buscas no conjunto de documentos e no dicionário. Assim, durante o processamento do texto, se a seqüência de texto, extraída de um documento, "casar" com uma seqüência inserida no dicionário, é mapeado no atributo correspondente da tabela o valor presente, caso contrário, é mapeado o valor ausente.

O ambiente computacional possui dois módulos, sendo que o primeiro foi desenvolvido utilizando a linguagem Java[1] e a segundo foi desenvolvido utilizando a linguagem Perl (Schwartz et al., 1997)[2], conforme ilustrado na Figura 2.2.

O primeiro módulo é responsável por realizar a interação com o usuário, por meio de uma interface gráfica construída utilizando os recursos para desenvolvimento de interfaces gráficas da ferramenta NetBeans[3]. O segundo módulo é responsável por aplicar os algoritmos da metodologia desenvolvida. A interação entre os dois módulos é realizada por meio de arquivos no formato XML. Na Figura 2.3 é ilustrada a interação entre os módulos do ambiente implementado.

No primeiro módulo, os algoritmos foram implementados utilizando o conceito de camadas, as quais são: interface, negócio e persistência. A camada de interface, geralmente chamada de *Graphical User Interface* — GUI — é responsável pela interação do usuário com o sistema. A camada de negócio é responsável pela implementação da lógica de negócio da aplicação, ou seja, nessa camada estão todas os algoritmos e classes inerentes ao domínio da

[1] http://java.sun.com/
[2] http://perldoc.perl.org/
[3] http://www.netbeans.org/

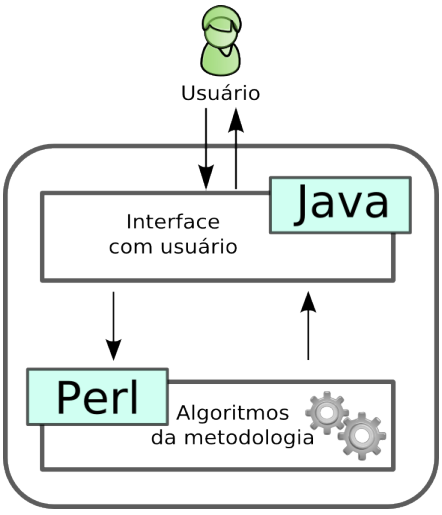

Figura 2.2: Módulos do ambiente

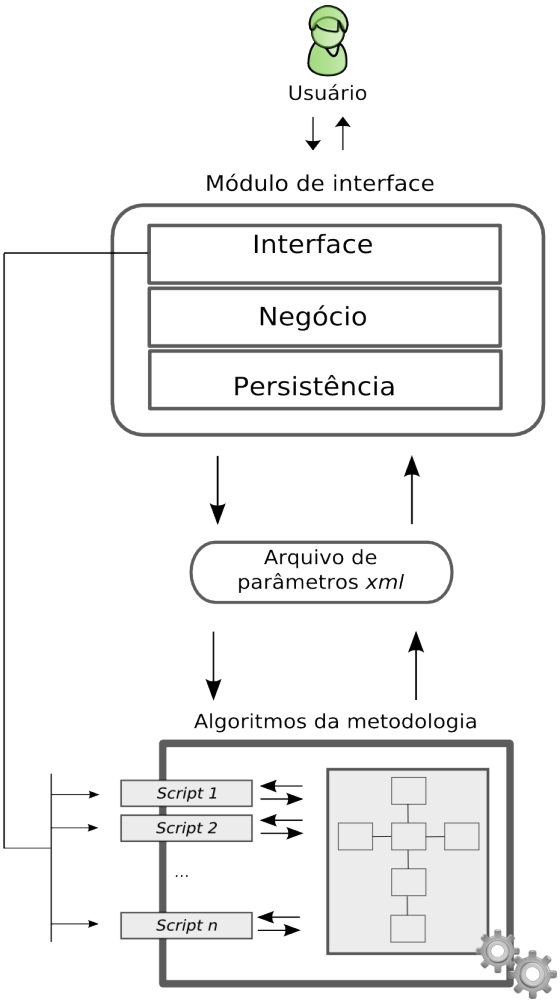

Figura 2.3: Arquitetura de interação entre os módulos do ambiente

aplicação, neste caso, aos algoritmos de interação com o usuário. A camada de persistência é responsável pela manipulação e inserção das informações fornecidas pelo usuário em um meio de armazenamento que possa ser recu-

perado posteriormente. Neste trabalho utilizamos arquivos no formato XML.

O segundo módulo é formado por *scripts*, os quais implementam os algoritmos da metodologia desenvolvida, e são invocados a partir do primeiro módulo. Também, o sistema utiliza diversos programas auxiliares tais como N-GRAM STATISTICS PACKAGE — NSP[4] (Banerjee & Pedersen, 2003), GRAPHVIZ[5] (Gansner & North, 2000), SENTER (Pardo, 2006), MXPOST (Ratnaparkhi, 1996) e TREETAGGER[6] (Schmid, 1994).

Para realizar a aplicação da metodologia é necessário que estejam disponíveis os documentos a partir dos quais serão aplicados os algoritmos e gerada a tabela atributo-valor. Esses documentos devem estar na forma textual em arquivos digitais no formato TXT. A interface de abertura do ambiente computacional é ilustrada na Figura 2.4 e a interface principal do ambiente computacional é ilustrada na Figura 2.5. Ela oferece quatro opções: Projeto, Pré-processamento, Gerar n-gramas e Árvores.

Figura 2.4: Interface de abertura

Utilizando a opção Projeto, é possível criar um projeto e carregá-lo posteriormente, uma vez que o sistema mantém as informações armazenadas para que possam ser recuperadas, permitindo, assim, aplicar a metodologia em períodos diferentes. Para aplicar a metodologia em um conjunto de documentos é necessário o usuário criar um novo projeto, conforme interface ilustrada na Figura 2.6.

O conjunto de documentos utilizado pelo ambiente computacional deve estar compactado em um arquivo no formato ZIP e o usuário deve indicar esse arquivo no ambiente. Na interface, depois de indicado esse arquivo, o sistema automaticamente preenche os campos restantes das abas Pasta 1 e Pasta 2 com os nomes das pastas que serão utilizadas como pastas auxiliares para armazenamento de arquivos temporários. Depois de definidas as informações o sistema gera automaticamente o arquivo *projeto.xml* no qual estão armazenadas informações sobre o projeto. Além desse arquivo XML, o sistema gera o arquivo *arquivoParaProcessar.xml* no qual são armazenados os nomes dos arquivos TXT que correspondem a cada documento do conjunto de documentos, conforme fragmento do arquivo *arquivosParaProcessar.xml* ilustrado na Figura 2.7.

[5] http://www.GraphViz.org/
[6] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

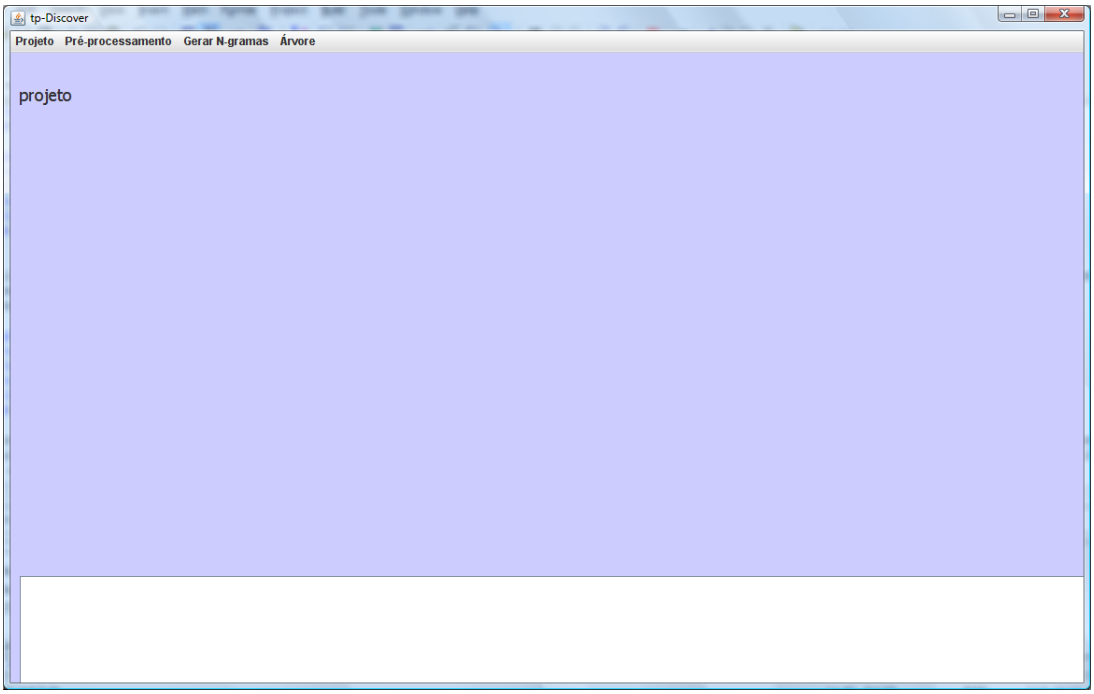

Figura 2.5: Interface principal

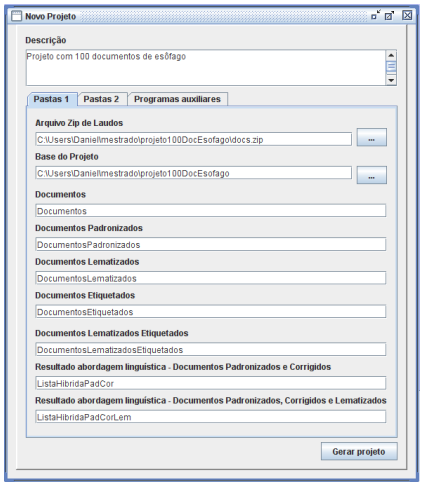

Figura 2.6: Interface para a construção de um novo projeto

No momento da geração de um novo projeto, o sistema cria automaticamente uma estrutura de pastas e arquivos conforme ilustrado na Figura 2.8. Essas pastas são utilizadas posteriormente para armazenar as informações que estarão sendo processadas durante a aplicação da metodologia. A seguir é descrito sucintamente a função de cada uma dessas pastas.

Documentos: Armazena o conjunto original de documentos;

DocumentosPadronizados: Armazena o conjunto de documentos após aplicação dos algoritmos de padronização;

DocumentosEtiquetados: Armazena o conjunto de documentos padronizados e etiquetados;

```
- <Arquivos number="100">
  - <enderecoArquivo>
      C:\Users\Daniel\mestrado\projeto100DocEsofago\Documentos\l73.txt
    </enderecoArquivo>
  - <enderecoArquivo>
      C:\Users\Daniel\mestrado\projeto100DocEsofago\Documentos\l94.txt
    </enderecoArquivo>
  - <enderecoArquivo>
      C:\Users\Daniel\mestrado\projeto100DocEsofago\Documentos\l130.txt
    </enderecoArquivo>
  - <enderecoArquivo>
      C:\Users\Daniel\mestrado\projeto100DocEsofago\Documentos\l187.txt
    </enderecoArquivo>
  - <enderecoArquivo>
      C:\Users\Daniel\mestrado\projeto100DocEsofago\Documentos\l204.txt
    </enderecoArquivo>
  - <enderecoArquivo>
      C:\Users\Daniel\mestrado\projeto100DocEsofago\Documentos\l222.txt
    </enderecoArquivo>
  - <enderecoArquivo>
      C:\Users\Daniel\mestrado\projeto100DocEsofago\Documentos\l279.txt
    </enderecoArquivo>
  - <enderecoArquivo>
      C:\Users\Daniel\mestrado\projeto100DocEsofago\Documentos\l292.txt
    </enderecoArquivo>
    ...
```


Figura 2.7: Arquivos correspondentes a cada documento do conjunto de documentos para processar

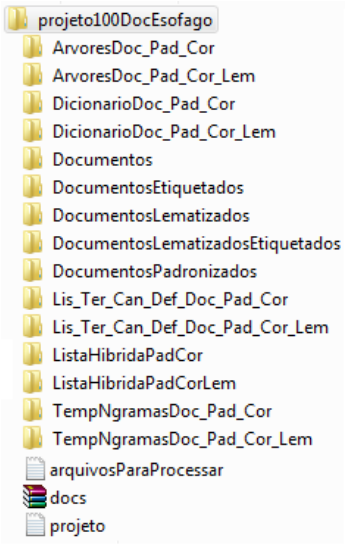

Figura 2.8: Estrutura de pastas geradas no novo projeto

DocumentosLematizados: Armazena o conjunto de documentos após a padronização e lematização;

DocumentosLematizadosEtiquetados: Armazena o conjunto de documentos padronizados, lematizados e etiquetados;

Lis_Ter_Can_Def_Doc_Pad_Cor: Armazena a lista de termos candidatos identificada a partir do conjunto de documentos padronizados e corrigidos

por meio do algoritmo de extração de terminologia, bem como a lista de termos definidos que são utilizadas para gerar as árvores utilizadas na identificação dos atributos;

Lis_Ter_Can_Def_Doc_Pad_Cor_Lem: Idem anterior para documentos lematizados;

ListaHibridaPadCor: Armazena as listas de termos identificadas no conjunto de documentos padronizados e corrigidos a partir de uma determinada máscara, por meio dos algoritmos de extração de terminologia;

ListaHibridaPadCorLem: Idem anterior para documentos lematizados;

TempNgramasDoc_Pad_Cor: Pasta temporária na qual são armazenadas as listas de n-gramas gerados a partir do conjunto de documentos padronizados e corrigidos;

TempNgramasDoc_Pad_Cor_Lem: Idem anterior para documentos lematizados;

ArvoresDoc_Pad_Cor: Nessa pasta é armazenado o arquivo XML contendo a árvore gerada a partir dos documentos padronizados e corrigidos. Também é utilizada para armazenar as figuras das árvores, se necessário;

ArvoresDoc_Pad_Cor_Lem: Idem anterior para documentos lematizados;

DicionarioDoc_Pad_Cor: Armazena o dicionário gerado por meio do conjunto de documentos padronizados e corrigidos. Armazena também a tabela atributo-valor gerada automaticamente;

DicionarioDoc_Pad_Cor_Lem: Idem anterior para documentos lematizados;

projeto.xml: Arquivo XML contendo informações sobre o projeto, tais como descrição sobre o projeto, informações sobre o endereço das pastas utilizadas no projeto entre outros;

arquivosParaProcessar.xml: Armazena os nomes dos arquivos correspondentes a cada documento do conjunto de documentos originais sobre os quais é aplicada a metodologia.

2.1 Pré-processamento

A primeira tarefa a ser realizada na aplicação da metodologia é realizar o pré-processamento dos documentos. Neste trabalho o pré-processamento compreende as seguintes cinco tarefas:

1. Padronização das informações;
2. Aplicação de substituições;
3. Correção ortográfica;

4. Aplicação de lematização; e

5. Estatísticas do CFU.

A seguir são apresentadas essas tarefas com mais detalhes.

2.1.1 Padronização

Na Figura 2.9 é ilustrada a interface de padronização de informações. Na

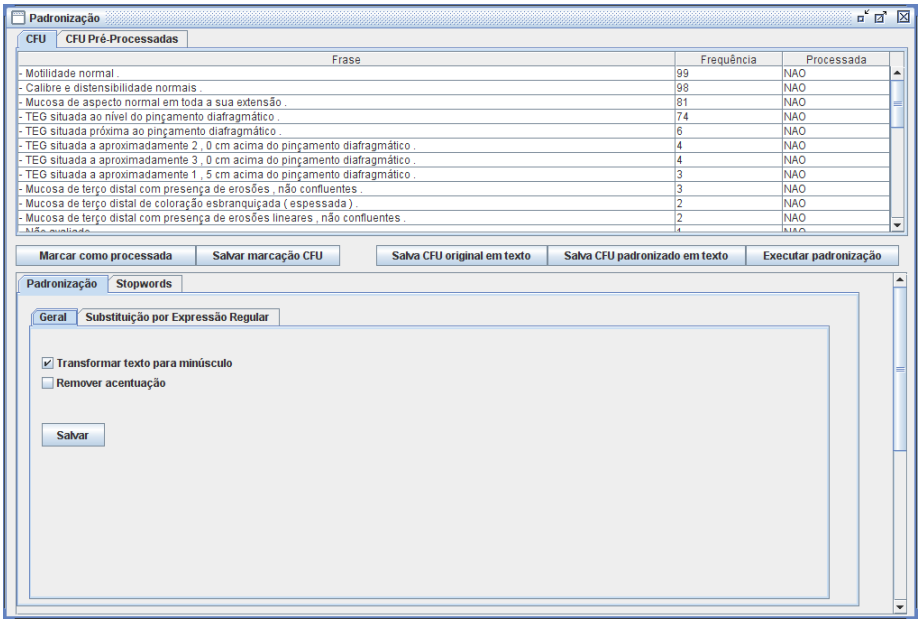

Figura 2.9: Interface de padronização

primeira vez que o usuário acessa essa interface, o sistema gera automaticamente um arquivo contendo o conjunto de frases únicas — CFU — identificadas no conjunto de documentos que está sendo processado e um arquivo de estatísticas desse CFU.

Nessa interface o usuário pode definir algumas modificações que deverão ser aplicadas sobre o conjunto de documentos tais como: informar se as informações deverão ser transformadas para minúsculo, informar a lista de *stopwords* (o usuário pode carregar um arquivo de *stopwords* a partir de um arquivo TXT ou inserir individualmente cada *stopword*) que deverão ser retiradas do texto e também informar, caso existam, as substituições de termos utilizando expressões regulares.

É importante destacar que as alterações que são configuradas nessa interface são aplicadas sobre todo o conjunto de documentos. Após serem aplicadas as alterações, o sistema gera um novo CFU e as estatísticas respectivas.

2.1.2 Mapear Substituições

O ambiente computacional permite que o usuário aplique substituições nos documentos. Diversos tipos de substituições são possíveis e as principais são:

- Substituição de palavras;
- Substituição de frases; e
- Substituição por expressão regular.

As substituições podem ser especificadas por meio da interface ilustrada na Figura 2.10.

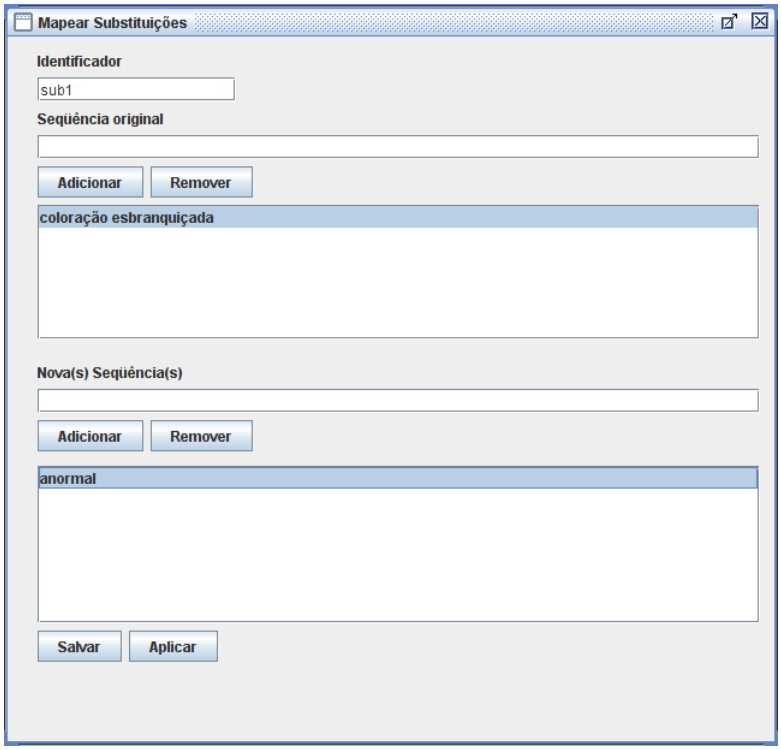

Figura 2.10: Interface para aplicação de substituições

Nessa interface, primeiramente o usuário define como entrada o nome da substituição e depois a substituição que deverá ser realizada. Na Figura 2.10 foi definida a substituição sub1 na qual será substituida a seqüência "coloração esbranquiçada" por "anormal". Após definir as substituições, as mesmas podem ser aplicadas sobre todo o conjunto de documentos padronizados.

2.1.3 Corretor de Palavras

O TP-DISCOVER permite que seja realizada a correção ortográfica nas informações contidas nos documentos. Duas opções são permitidas: manual ou automática. Na Figura 2.11 é ilustrada a interface utilizada para realizar a correção ortográfica, a qual pode ser utilizada tanto para realizar a correção manual como a automática.

No primeiro modo, modo manual, o usuário deve analisar manualmente o CFU padronizado, o qual pode ser acessado na interface ilustrada na Figura 2.9, e identificar as palavras incorretas. A medida que essas palavras são identificadas, elas podem ser inseridas nessa interface. Na caixa "Sugestão palavra" o usuário deve inserir a palavra a ser corrigida e na caixa "Palavra

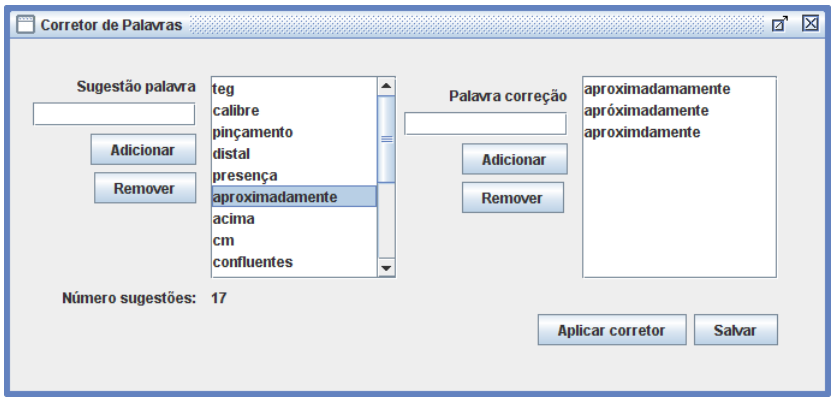

Figura 2.11: Interface do corretor de palavras

correção" deve ser inserida a palavra incorreta associada à palavra que foi inserida na caixa "Sugestão palavra". Depois de inseridas todas as palavras a serem corrigidas, o usuário pode aplicar a execução do algoritmo de correção de palavras, o qual irá substituir a palavra incorreta pela palavra correta no conjunto de documentos padronizados.

No modo de correção automática, o ambiente computacional gera automaticamente, utilizando um algoritmo implementado como parte deste trabalho, uma lista de sugestões de correções baseado em cálculo de medida de distância entre cada termo do texto. O usuário deve então analisar essa lista e remover sugestões incorretas e pode, também, inserir manualmente outras correções.

Em ambos os casos, modo manual ou modo automático, para aplicar o corretor de palavras, o usuário deve armazenar a lista de correções, opção "Salvar" da Figura 2.11. Depois de salvar a lista de correções, pode ser aplicado o algoritmo de correção no conjunto de documentos por meio da opção "Aplicar corretor", na Figura 2.11.

2.1.4 Lematização

A aplicação da lematização é realizada sobre o conjunto de documentos padronizados e corrigidos por meio da interface ilustrada na Figura 2.12.

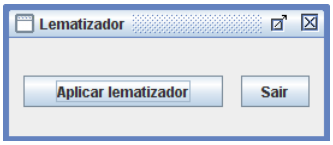

Figura 2.12: Interface do lematizador

Após a aplicação do algoritmo de lematização, o usuário pode verificar a nova lista de frases únicas gerada (CFU) a partir dos documentos lematizados.

2.1.5 Estatísticas do CFU

O ambiente computacional possibilita que sejam visualizadas estatísticas do conjunto de frases únicas geradas em cada passo da aplicação da metodolo-

gia. Essas estatísticas podem ser visualizadas por meio da interface ilustrada na Figura 2.13, na qual são mostradas as estatísticas do CFU padronizado, corrigido e lematizado.

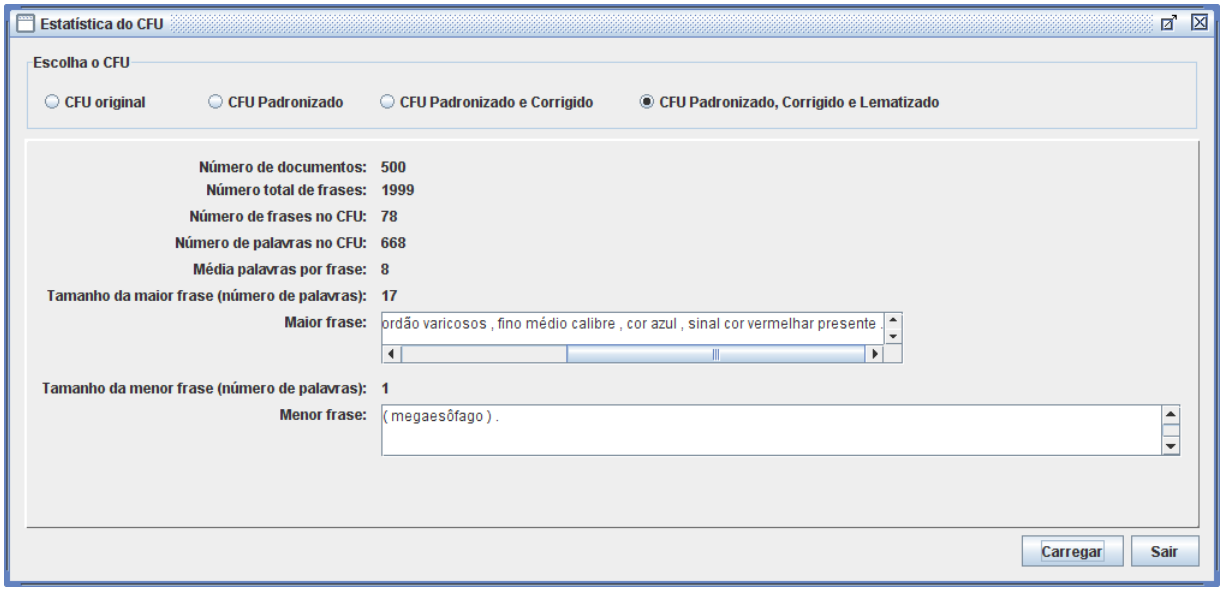

Figura 2.13: Interface de estatísticas do CFU

2.2 Extração de Terminologia

Na aplicação da metodologia é realizada a extração de terminologia do domínio utilizando uma abordagem híbrida (Teline et al., 2003). Na abordagem adotada neste trabalho, primeiramente é realizada a extração de termos do domínio que atendem a uma determinada máscara sintática e depois são analisadas as freqüências desses termos. Para realizar a extração de terminologia é necessário que os documentos estejam etiquetados[7].

O ambiente computacional TP-DISCOVER possibilita que sejam etiquetados os documentos utilizando o etiquetador MXPOST. Esse etiquetador foi desenvolvido para língua geral. Portanto, para o domínio médico, objeto de estudo deste trabalho, é necessário realizar a correção de algumas etiquetas, uma vez que algumas etiquetas são associadas de maneira incorreta. Para isso, inicialmente o usuário deve etiquetar o CFU gerado do conjunto de documentos a partir do qual serão extraídas as unidades terminológicas. Essa tarefa é realizada automaticamente pelo MXPOST por meio da interface ilustrada na Figura 2.14, na qual é possível selecionar a qual conjunto de frases únicas (CFU) deverá ser aplicada a etiquetação.

Após, deve ser analisado, manualmente, o CFU etiquetado e as etiquetas consideradas incorretas devem ser identificadas e corrigidas pelo usuário.

[7]Consiste em associar a cada termo do documento sua classe gramatical tais como substantivo, artigo e verbo, entre outros.

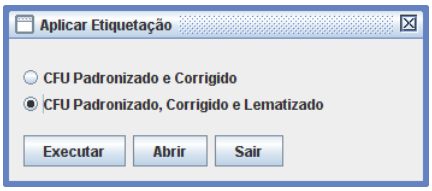

Figura 2.14: Interface de etiquetação do CFU

2.2.1 Correção de Etiquetas

As etiquetas incorretas identificadas pelo usuário no CFU devem ser inseridas na interface ilustrada na Figura 2.15, juntamente com a etiqueta correta correspondente.

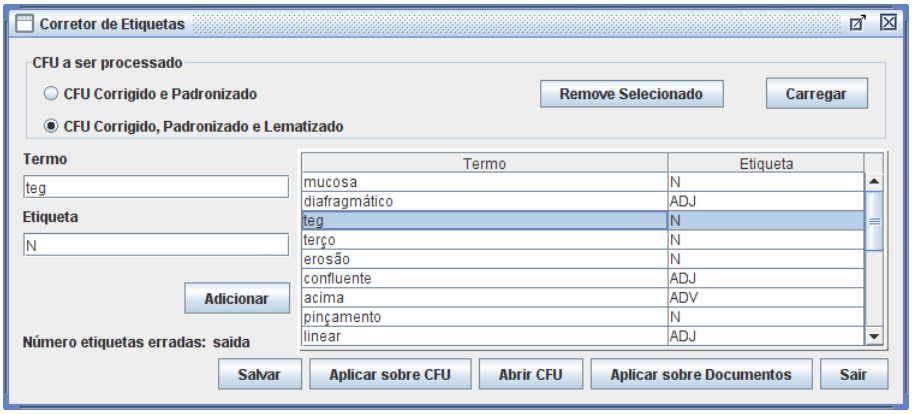

Figura 2.15: Interface de correção de etiquetas

Nessa interface, primeiramente o usuário informa qual conjunto está sendo tratado, ou seja, no conjunto de documentos padronizados e corrigidos ou no conjunto de documentos padronizados, corrigidos e lematizados. Depois o usuário pode corrigir as etiquetas incorretas fornecidas pelo MXPOST. Após analisar o CFU e inserir as etiquetas corretas, pode ser aplicada a correção de etiquetas sobre o CFU e analisar novamente esse CFU para verificar se todas as etiquetas estão corretas. Após realizar esse processo, o conjunto de documentos está pronto para que sejam extraídas as unidades terminológicas por meio dos algorimtos de extração de terminologia.

Conforme mencionado, a extração de terminologia utilizando a abordagem híbrida proposta neste trabalho, necessita que sejam definidas as máscaras a partir das quais serão extraídas as unidades terminológicas do conjunto de documentos etiquetados. Na interface ilustrada na Figura 2.16 é possível definir as máscaras que serão aplicadas na extração de terminologia.

As máscaras utilizadas podem ser: N (substantivo), ADJ (adjetivo), VERB (verbo), LOCU (locução), ADV (advérbio), PREP+ART (preposição + artigo), tais como qualquer combinação das mesmas. Após definidas as máscaras deve ser aplicado o algoritmo de extração de terminologia (Honorato & Monard, 2008), o qual irá identificar, em todo o conjunto de documentos selecionado, todos os termos que casam com as máscaras que foram definidas. Na Figura 2.17 é ilustrada a interface utilizada para a extração de terminologia.

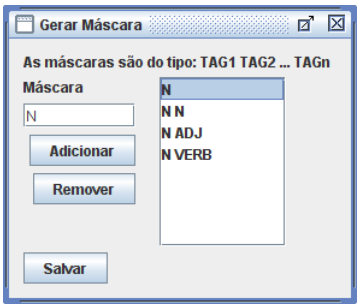

Figura 2.16: Interface de definição de máscaras

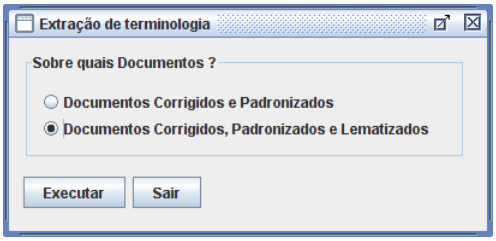

Figura 2.17: Interface de extração de terminologia

Nessa interface, o usuário deve selecionar sobre quais documentos será realizada a extração de terminologia. Após realizada a extração de terminologia o usuário pode visualizar, por meio da interface ilustrada na Figura 2.18, as unidades terminológicas que foram extraídas a partir de cada máscara definida — N nessa figura.

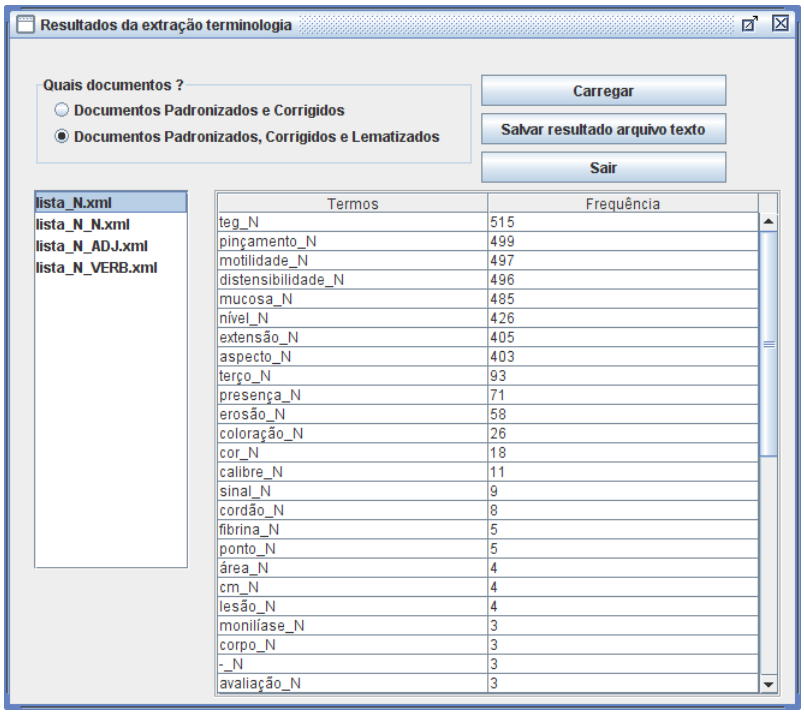

Figura 2.18: Interface de resultado da extração de terminologia

Ou seja, nessa interface o usuário deve selecionar quais documentos estão sendo processados e escolher qual máscara quer visualizar. Depois de selecio-

nada a máscara desejada o sistema retorna as unidades terminológicas que casaram com a máscara e a freqüência do termo no conjunto de documentos.

2.2.2 Geração de listas de n-gramas

O ambiente computacional permite que o usuário execute a geração de listas de n-gramas, as quais são utilizadas pela metodologia para a construção das árvores em uma etapa posterior. Na interface ilustrada na Figura 2.19 é possível realizar essa tarefa.

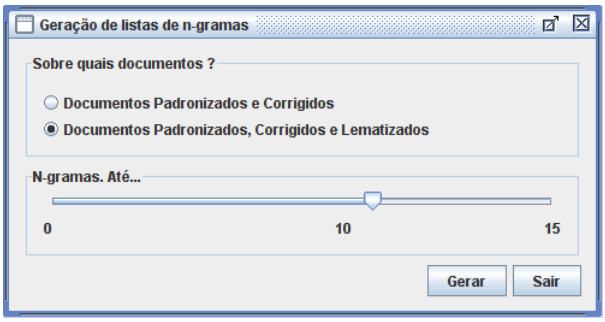

Figura 2.19: Interface de geração das listas de n-gramas

Primeiramente, o usuário deve selecionar sobre qual conjunto de documentos devem ser extraídas as listas de n-gramas. Depois, é necessário selecionar o número de n-gramas. Por exemplo, se o usuário selecionar 3, o ambiente irá gerar até 3-gramas: uma lista de unigramas, outra lista de bigramas e uma terceira lista de trigramas. Essas listas são utilizadas na próxima etapa de geração de árvores a partir das quais são identificados os atributos da tabela atributo-valor.

2.3 Identificação de Atributos e Construção da Tabela Atributo-valor

Antes da geração das árvores, é necessário definir quais serão os termos (unidades terminológicas) que serão utilizados como raíz das árvores. Neste trabalho foram implementadas algumas heurísticas que auxiliam na identificação desses termos. As heurísticas implementadas são aplicadas sobre duas listas de n-gramas: uma lista de unigrama e outra lista de bigrama. A tarefa de identificação de termos é realizada utilizando a interface ilustrada na Figura 2.20.

Nessa interface, primeiramente deve ser selecionado o conjunto de documentos que está sendo trabalhado. Depois deve ser selecionada a lista de unigramas e a lista de bigramas sobre as quais serão aplicadas as heurísticas. Também devem ser especificados os parâmetros Alfa e Theta, explicados em detalhes em Honorato & Monard (2008), utilizados na aplicação das heurísticas. Depois de realizar a aplicação das heurísticas sobre as duas listas de termos, pode ser realizada ainda uma análise manual para definir a lista de

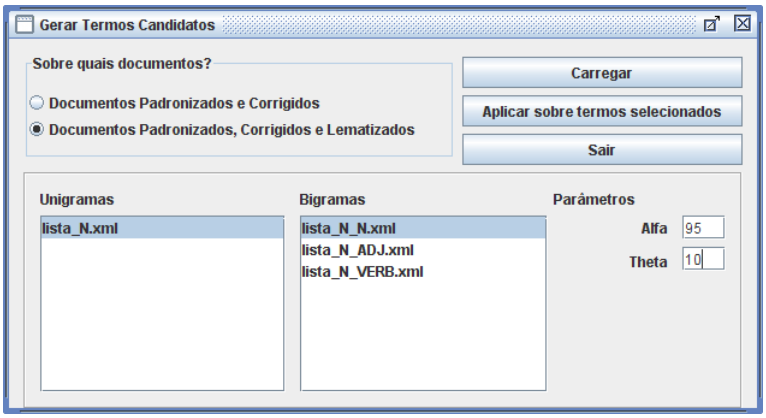

Figura 2.20: Interface de geração da lista de termos candidatos

termos final de termos que será utilizada para geração da árvore. Essa análise é realizada na interface ilustrada na Figura 2.21.

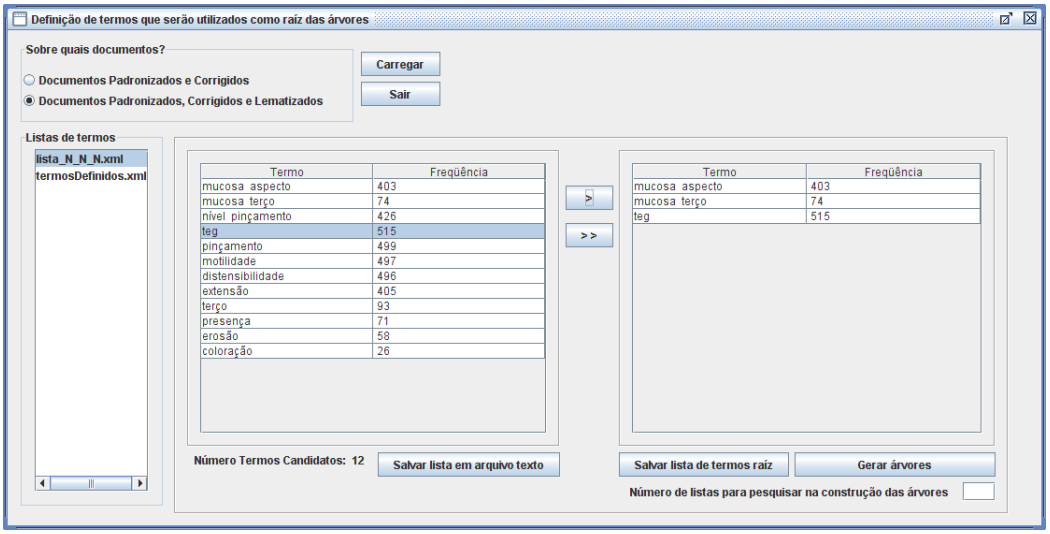

Figura 2.21: Interface de definição de termos raíz

Nessa interface, primeiramente deve ser selecionado o conjunto de documentos que está sendo trabalhado. Depois, o usuário seleciona o conjunto de termos candidatos extraídos por meio da aplicação de heurísticas o qual está armazenado, no exemplo da Figura 2.21, no arquivo *listaN_NN.xml*. O nome dese arquivo indica que está armazenada uma lista de termos que foi gerada por meio da aplicação das heurísticas sobre uma lista de unigramas N e outra lista de bigramas NN. O usuário pode selecionar os termos manualmente com a ajuda de um especialista do domínio ou pode selecionar todos termos identificados pelo algoritmo. Após selecionar os termos, eles devem ser armazenados e o usuário deve definir qual o número de listas para pesquisar na construção da árvore. Ou seja, se o usuário definir 3, o ambiente irá utilizar as listas de unigramas, bigramas e trigramas para gerar a árvore[8]. O valor padrão pode ser extraído da média de palavras por frase, a partir do arquivo de estatísticas do CFU. O usuário pode redefinir esse valor e executar

[8]Essas listas devem ser geradas a partir da interface apresentada na Seção 2.2.2

a geração de árvores. Para cada termo definido o ambiente computacional irá gerar uma árvore a qual será utilizada para a identificação de atributos posteriormente. Caso estejam ocorrendo interações com o especialista para a identificação manual dos atributos, as árvores geradas podem ser visibilizadas por meio da interface ilustrada na Figura 2.22.

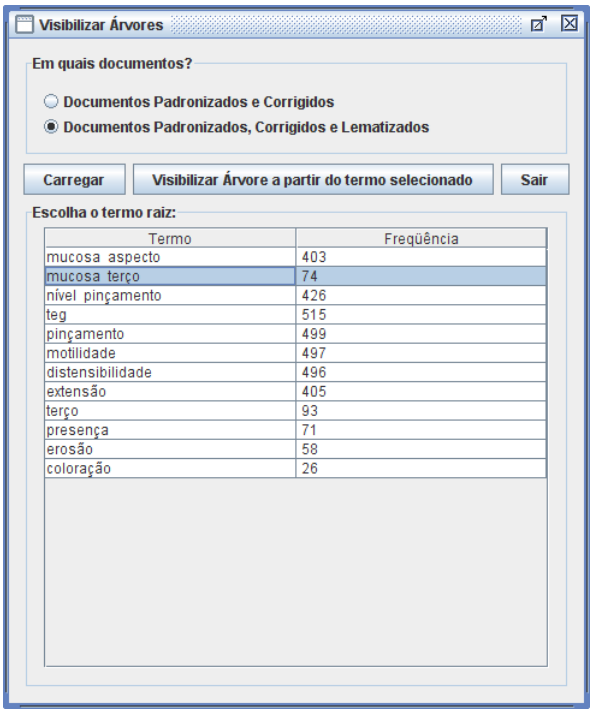

Figura 2.22: Interface de visibilização das árvores

Nessa interface, o usuário primeiramente deve selecionar o conjunto de documentos que está sendo tratado e carregar a lista de termos que foram definidos para a geração das árvores. Para gerar a figura de uma árvore a partir de um termo (unidade terminológica) deve ser selecionado o termo e requisitado ao ambiente a geração da árvore. O ambiente computacional irá gerar a árvore, conforme ilustrada na Figura 2.23, a partir do termo selecionado, o termo `mucosa terço` no exemplo na Figura 2.23.

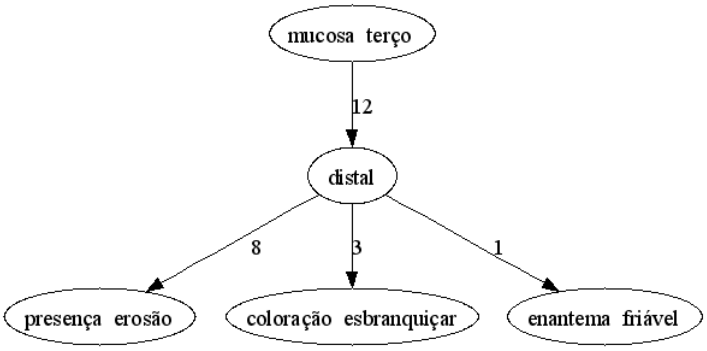

Figura 2.23: Árvore gerada a partir do termo mucosa terço

A opção de visibilizar a árvore é muito útil quando está sendo aplicada a metodologia com a ajuda de um especialista do domínio.

Após a geração das árvores, o usuário pode executar a geração de atributos e preenchimento da tabela atributo-valor. Para esse fim, existem dois modos de construir o dicionário de atributos: modo manual e modo automático.

O primeiro é realizado com a ajuda de um especialista, o qual irá auxiliar na definição dos atributos por meio da análise das árvores. Após identificados os atributos, eles devem ser inseridos utilizando a interface ilustrada na Figura 2.24. Nessa interface, primeiramente o usuário deve selecionar qual conjunto

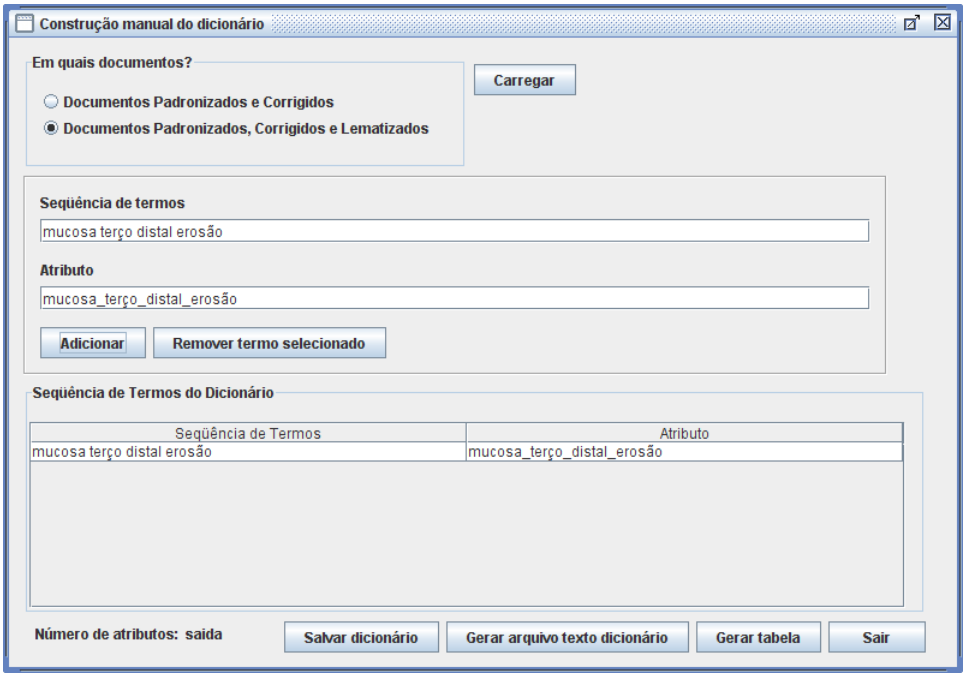

Figura 2.24: Interface de construção manual do dicionário

de documentos está sendo tratado. Depois, é iniciada a inserção de atributos. Os atributos são formados por uma seqüência de termos identificados nos ramos das árvores. Automaticamente o sistema define o nome do atributo que será armazenado na tabela a partir da seqüência definida pelo usuário. Após gerar os atributos, na mesma interface, o usuário pode executar a geração da tabela atributo-valor. Essa opção faz com que o ambiente computacional automaticamente crie a tabela atributo-valor formada pelos atributos identificados e preencha a tabela. O preenchimento é realizado da seguinte maneira. Se uma determinada seqüência utilizada para gerar o atributo for identificada no texto do documento, o atributo correspondente é preenchido com o valor 1, de presente. Depois de processar o documento, os atributos que não foram identificados são preenchidos com o valor 0, de ausente.

No segundo modo de identificar os atributos, modo automático, o usuário deve acessar a interface ilustrada na Figura 2.25 e selecionar qual conjunto de documentos está sendo tratado.

Depois, o usuário deve definir qual o limiar de criação dos atributos. Esse limiar refere-se à porcentagem relacionada à freqüência com que um determinado ramo da árvore aparece em relação ao número de laudos. Por exemplo, a partir de um conjunto de 100 laudos e se for definido o limiar 5, o ambiente computacional irá selecionar todos os ramos que possuem freqüência maior

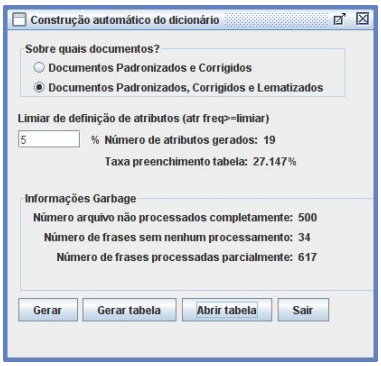

Figura 2.25: Interface de construção automática do dicionário

ou igual a 5 para serem atributos da tabela atributo-valor. Depois de definidos os atributos, pode ser gerada automaticamente a tabela atributo-valor. Depois de gerada a tabela o ambiente computacional fornece a taxa de preenchimento da tabela, ou seja, a porcentagem de células que foram preenchidas como presentes. A Figura 2.26 ilustra um fragmento de uma tabela gerada pelo ambiente.

```
arquivo;teg_situar_nivel_pinçamento_diafragmático; ...
173.txt.seg;1;1;0;1;0;0;0;1;0;1
194.txt.seg;1;1;0;1;0;0;0;1;0;1
1130.txt.seg;0;1;0;1;0;0;0;1;1;1
1187.txt.seg;1;1;0;1;0;0;0;1;0;1
1204.txt.seg;0;1;0;0;0;0;0;1;1;1
1222.txt.seg;0;1;0;0;0;0;1;1;1;1
1279.txt.seg;0;1;1;0;0;0;1;1;1;1
1292.txt.seg;1;1;0;1;0;0;0;1;0;1
1297.txt.seg;1;1;0;1;0;0;0;1;0;1
1430.txt.seg;1;1;0;1;0;0;0;1;0;1
1452.txt.seg;1;1;1;0;0;0;1;1;0;1
1473.txt.seg;1;1;0;1;0;0;0;1;0;1
1535.txt.seg;1;1;1;0;0;0;1;1;0;1
1547.txt.seg;0;1;0;0;0;0;1;1;1;1
1569.txt.seg;1;1;0;1;0;0;0;1;0;1
1599.txt.seg;1;1;0;1;0;0;0;1;0;1
1610.txt.seg;1;1;0;1;0;0;0;1;0;1
1724.txt.seg;1;1;0;0;0;0;1;1;0;1
1731.txt.seg;1;1;0;1;0;0;0;1;0;1
1743.txt.seg;1;1;0;1;0;0;0;1;0;1
1748.txt.seg;1;1;0;1;0;0;0;1;0;1
1771.txt.seg;1;1;0;1;0;0;0;1;0;1
1800.txt.seg;0;1;0;1;0;0;0;1;0;1
1840.txt.seg;0;1;1;0;0;0;1;1;1;1
1871.txt.seg;0;1;1;0;1;1;1;1;0;1
...
```

Figura 2.26: Exemplo da tabela gerada pelo ambiente computacional

Nessa tabela, a primeira linha contém os atributos identificados que participam da tabela atributo-valor. As linhas restantes contém o nome do arquivo do documento processado, seguido dos valores dos atributos identificados. Na geração da tabela atributo-valor também é gerado um arquivo, o qual denominados de *garbage*, no qual são armazenadas as frases dos documentos que não sofreram nenhum processamento e as frases que foram processadas parcialmente.

Finalmente, o ambiente computacional calcula o número dessas frases e exibe o resultado para o usuário. É bom ressaltar que essa é uma informação importante, uma vez que indica a porcentagem de frases que estão sendo processadas completamente, parcialmente ou não estão sofrendo nenhum processamento, em relação ao número total de frases do conjunto de documentos.

Capítulo 3

Biblioteca de Classes

Conforme mencionado, o ambiente computacional TP-DISCOVER foi desenvolvido utilizando as linguagens Java e Perl. Na primeira foi desenvolvida a interface de interação com o usuário e na segunda foram desenvolvidos os algoritmos que implementam a metodologia. A seguir são apresentados os principais diagramas de classes implementados para o desenvolvimento do ambiente computacional.

3.1 Interface com o Usuário

A interface de interação com o usuário foi desenvolvida na linguagem Java utilizando o conceito de três camadas, as quais são denominadas camada de interface, camada de negócio e camada de persistência. Na Figura 3.1 é ilustrado o esquema que representa esse conceito.

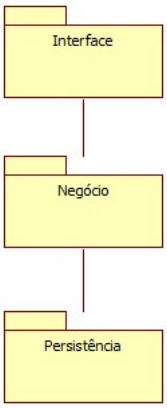

Figura 3.1: Arquitetura da interface com o usuário

Na camada de interface estão armazenadas todas as classes responsáveis aos formulários de interação com o usuário. A camada de negócio contém as classes responsáveis pelo tratamento das informações que são passadas pela camada de interface e requisições para a camada de persistência. A camada de persistência é responsável pelo armazenamento e recuperação de informações. Neste trabalho, a camada de persistência é responsável pela manipulação de arquivos no formato XML.

A interação entre essas camadas é realizada da seguinte maneira. Qualquer requisição do usuário em uma determinada interface da camada de interface é passada para a camada de negócio, a qual é responsável por manipular essa informação e fazer a requisição na camada de persistência. Esta, por sua vez, realiza a operação requisitada e retorna as informações para a classe chamadora da camada de negócio, a qual, por sua vez, retorna as informações para a classe chamadora da camada de interface.

No ambiente computacional TP-DISCOVER podem ser identificados cinco tipos de arquivos no módulo de interface desenvolvida na linguagem Java, os quais representam as classes implementadas nas três camadas. O cinco tipos de arquivos são os correspondentes a:

1. classe de interface: essas classes estão armazenadas na camada de interface e possuem o prefixo *Tfrm*. Elas são responsáveis pelas interfaces de interação com o usuário;

2. classe de entidade: são responsáveis pelo armazenamento temporário de informações e possuem o prefixo *TE*.

3. classe de conjunto: armazenam conjuntos de informações, por exemplo, conjunto de frases únicas, e possuem o prefixo *TConjunto*.

4. classe de negócio: são responsáveis pela comunicação com a camada de interface e persistência e possuem o prefixo *TN*.

5. classe de persistência: responsável pelo armazenamento e recuperação de informações de arquivos XML.

3.2 Diagramas de classes

A seguir são apresentados os principais diagramas de classes implementados neste trabalho. Uma descrição das classes implementadas na linguagem Java e na linguagem Perl, assim como uma descrição dos *scripts* implementados na linguagem Perl são apresentados nos anexos A, B e C, respectivamente.

3.2.1 Padronização

A interface de padronização, ilustrada na Figura 3.2, é responsável pela aplicação de padronizações tais como, toquenização, segmentação, remoção de *stopwords*, transformação para minúsculo e substituição por expressão regular. Na Figura 3.3 é ilustrado o diagrama de classes utilizado para realizar as operações de padronização.

Nesse diagrama, a classe *TfrmPrePadronizar* representa a interface que o usuário está interagindo. Depois de informar as padronizações necessárias, as informações são passadas para as classes inferiores e, essas classes, são responsáveis por interagir com a classe de persistência. Nesta última, as informações são armazenadas nos arquivos XML. Quando o usuário aplica a execução das padronizações, o ambiente computacional realiza uma chamada

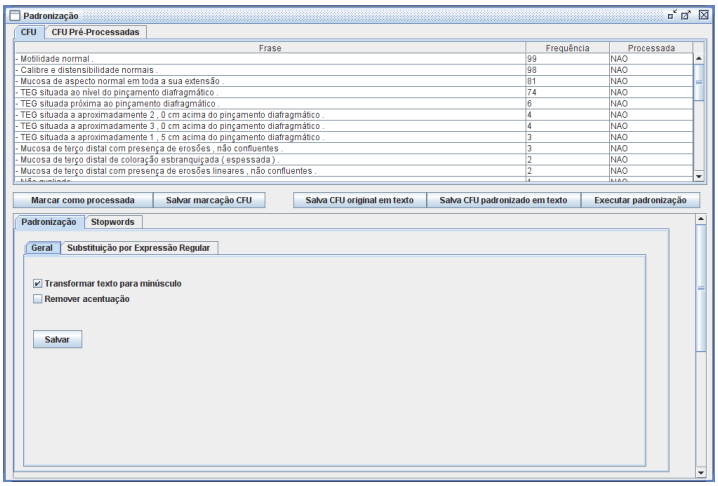

Figura 3.2: Interface de padronização

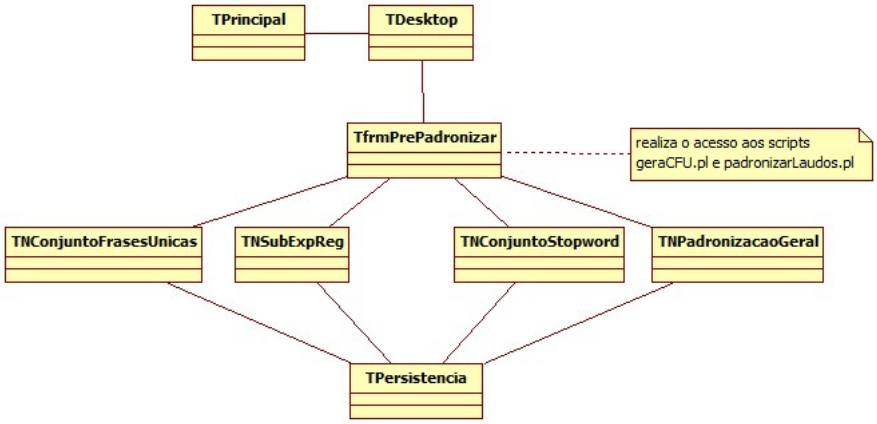

Figura 3.3: Diagrama de classes de padronização

externa, que faz a ligação com a linguagem Perl, a qual tem por objetivo aplicar as padronizações sobre o conjunto de documentos e gerar novamente o conjunto de frases únicas.

O princípio de interação entre as classes apresentado para a interface de padronização é aplicado para todas as interfaces restantes do ambiente computacional TP-DISCOVER. Ou seja, quando existe alguma informação a ser armazenada ou recuperada, a classe da interface se comunica com a classe de negócio, a qual, por sua vez, se comunica com a classe de persistência. Se for uma tarefa de armazenamento de informações, a classe de persistência retorna um número indicando se foi realizada com sucesso ou não. Caso seja para recuperar alguma informação, a classe de persistência retorna o conjunto de informações para a camada de negócio, a qual, por sua vez, retorna para a classe de interface que é exibida para o usuário.

3.2.2 Gerar Termos Candidatos

Antes da geração de termos candidatos a serem utilizados como termos raíz das árvores, é realizada a seleção do conjunto de documentos e das listas de

unigramas e bigramas que serão utilizados para a aplicação das heurísticas propostas neste trabalho (Honorato & Monard, 2008)[1]. As classes utilizadas para selecionar essas listas são ilustradas na Figura 3.4.

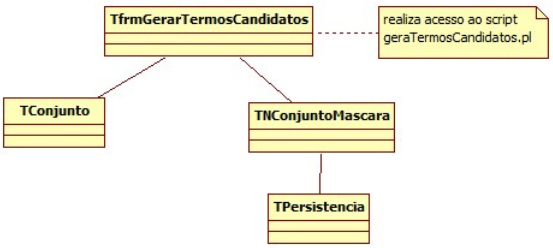

Figura 3.4: Diagrama de classes para gerar termos candidatos (Java)

Depois do usuário selecionar as listas de unigramas e bigramas a serem utilizados, a classe *TfrmGerarTermosCandidatos* realiza uma chamada externa para o *script geraTermosCandidatos.pl*, o qual, por sua vez, acessa as classes ilustradas no diagrama apresentado na Figura 3.5.

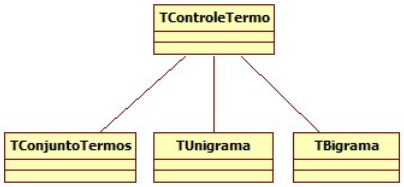

Figura 3.5: Diagrama de classes para gerar termos candidatos (Perl)

Na identificação de termos candidatos, a classe *TControleTermo* recebe as listas de unigramas e bigramas, além dos parâmetros Alfa e Theta, e realiza a aplicação de heurísticas sobre essas listas. Como resultado, tem-se a lista de termos candidatos que poderão ser utilizados na geração das árvores.

3.2.3 Gerar Árvores

Depois de definidos os termos (unidades terminológicas) que serão utilizados como raíz das árvores, a classe *TfrmDefineTermos*, ilustrada na Figura 3.6 realiza uma chamada externa para o *script geraArvore.pl* passando também o número de listas de n-gramas que deverão ser pesquisadas.

No *script*, as classes ilustradas na Figura 3.7 são utilizadas para a geração das árvores. A classe *TControle* carrega as listas de n-gramas que foram definidas e passa para a classe *TArvore*, a qual é responsável por realizar as pesquisas nessas listas e identificar os n-gramas de interesse que serão utilizadas pelo *script geraArvore.pl* na construção da árvore.

[1]As heurísticas propostas consideram listas de unigramas e bigramas, as quais, se necessário, podem ser estendidas incrementando o n-grama para $n > 2$

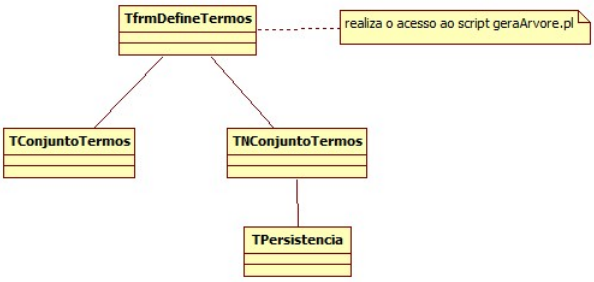

Figura 3.6: Diagrama de classes para gerar árvores

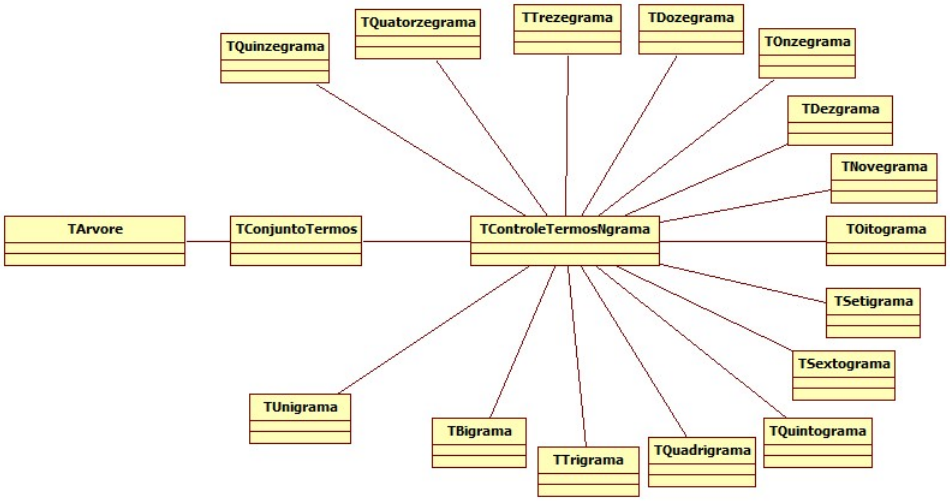

Figura 3.7: Diagrama de classes para gerar árvores (Perl)

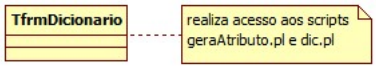

Figura 3.8: Diagrama de cltasses para construção da tabela

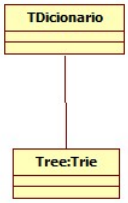

Figura 3.9: Diagrama de classes para construção da tabela (Perl)

3.2.4 Dicionário

A geração dos atributos que irão compor a tabela atributo-valor é realizada pelo *script geraAtributos.pl*, o qual é acessado por meio da classe *TfrmDicionario*, ilustrada na Figura 3.8.

Depois, por meio da mesma interface, pode ser executado o *script dic.pl*,

o qual é responsável pela construção da tabela atributo-valor. Esse *script* faz uso das classes ilustradas na Figura 3.9.

Na classe *TDicionario* estão implementados os métodos que fazem o casamento de padrões entre o texto contido no conjunto de documentos e o dicionáro implementado utilizando a estrutura de dados árvore trie.

Referências Bibliográficas

Banerjee, S. & Pedersen, T. (2003). The design, implementation, and use of the Ngram Statistic Package. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pag. 370–381, Cidade do México, México. Citado na página 7.

Gansner, E. R. & North, S. C. (2000). An open graph visualization system and its applications to software engineering. *Software — Practice and Experience*, 30(11):1203–1233. Citado na página 7.

Honorato, D. D. F. (2008). Metodologia de transformação de laudos médicos não estruturados e estruturados em uma representação atributo-valor. Dissertação de Mestrado (a ser defendida), ICMC-USP. Citado na página 3.

Honorato, D. D. F., Lee, H. D., Monard, M. C., Wu, F. C., Machado, R. B., Neto, A. P., & Ferrero, C. A. (2005). Uma metodologia para auxiliar no processo de construção de bases de dados. In *Anais do V Encontro Nacional de Inteligência, XXV Congresso da Sociedade Brasileira de Computação*, pag. 593–601, Porto Alegre, RS, Brasil. Citado na página 3.

Honorato, D. D. F. & Monard, M. C. (2008). Descrição de uma metodologia de mapeamento de informações não estruturadas em uma representação atributo-valor. Relatório técnico 317, ICMC-USP, São Carlos, SP, Brasil. Citado nas páginas 3, 4, 15, 17, e 25.

Pardo, T. A. S. (2006). Senter: Um segmentador sentencial automático para o português do brasil. Relatório técnico, ICMC-USP, São Carlos, SP. Citado na página 7.

Pavel, S. & Nolet, D. (2002). *Handbook of terminology*. Minister of Public Works and Government Services Canada, Québec, Canadá. `http://www.bureaudelatraduction.gc.ca/pwgsc_internet/en/publications/documents/handbook.pdf`. Citado na página 5.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In Brill, E. & Church, K., editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pag. 133–142. Nova Jérsei, EUA. Citado na página 7.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, Manchester, UK. unknown. Citado na página 7.

Schwartz, R., Christiansen, T., & Pyle, L. W. (1997). *Learning Perl*. California. Citado na página 5.

Teline, M. F., de B. Almeida, G. M., & Aluísio, S. M. (2003). Extração manual e automática de terminologia: comparando abordagens e critérios. In *Proceedings of the 16th Brazilian Symposium on Computer Graphics and Image Processing*, volume 1, pag. 1–12, São Carlos, Brasil. Citado na página 14.

Weiss, S. M., Indurkhya, N., & Zhang, T. u. (2005). *Text mining : predictive methods for analyzing unstructured information;*. Springer, New York, NY. Citado na página 2.

Apêndice A

Descrição das Classes Implementadas na Linguagem Java

A seguir é apresentada sucintamente cada uma das classes do ambiente computacional TP-DISCOVER implementadas na linguagem Java.

TPrincipal: Classe principal do ambiente computacional;

TDesktopFrame: Interface responsável por realizar a chamada às demais classes de interface do ambiente computacional;

TProjeto: Interface responsável pela criação de um novo projeto;

TfrmPrePadronizar: Classe da interface responsável por realizar a padronização dos documentos;

TfrmMapearSubstituicoes: Interface responsável pelo mapeamento de substituições;

TfrmCorretorPalavras: Classe responsável pela interface de correção ortográfica;

TfrmLematizar: Classe da interface de lematização;

TfrmEtiquetadorCFU: Classe responsável pela interface de etiquetação do CFU;

TfrmEtiquetadorDocumentos: Classe responsável pela interface de etiquetação de documentos;

TfrmCorretorEtiquetas: Classe responsável pela interface de correção de etiquetas;

TfrmEstatistica: Classe da interface de estatísticas dos CFU;

TfrmGeraNgramaEstatistica: Classe responsável pela interface de geração de listas de n-gramas com suas respectivas freqüências;

TfrmGeraMascara: Classe responsável pela interface de geração de máscaras;

TfrmGeraNgramaHibrida: Classe responsável pela interface de geração de n-gramas usando a abordagem híbrida;

TfrmGerarTermosCandidatos: Classe responsável pela geração de termos candidatos por meio da aplicação de heurísticas;

TfrmDefineTermos: Classe responsável pela definição de termos que serão utilizados como raíz das árvores;

TfrmVisibilizarArvores: Classe responsável pela interface de visibilização de árvores;

TfrmDicionario: Classe da interface do dicionário manual;

TfrmDicionarioAutomatico: Classe responsável pela interface do dicionário automático;

TConjunto: Classe base responsável pelo armazenamento de um conjunto de objetos genéricos;

TConjuntoEtiquetaCorreta: Classe responsável pelo armazenamento de uma lista de objetos do tipo *TEEtiquetaCorreta*;

TConjuntoFrasesUnicas: Classe utilizada para armazenar a lista de objetos do tipo *TEFraseUnica*, que representa uma determinada frase única;

TConjuntoMascara: Classe responsável pelo armazenamento de um conjunto de máscaras, no qual cada máscara é representado pelo objeto *TEMascara*;

TConjuntoNGramas: Classe na qual são armazenadas as listas de n-gramas;

TConjuntoSeqTermos: Classe responsável por armazenar as listas de seqüências de termos utilizada na construção do dicionário. Cada seqüência é armazenada em um objeto do tipo *TESeqTermos*;

TConjuntoStopword: Classe responsável pelo armazenamento do conjunto de *stopwords*. Cada elemento da lista é um objeto do tipo *TEStopword*;

TConjuntoSubExpReg: Classe responsável por armazenar as substituições inseridas na interface de padronização. Cada elemento do conjunto é um objeto do tipo *TESubExpReg*;

TConjuntoSubstituicao: Classe responsável por armazenar o conjunto de substituições. Cada elemento do conjunto é um objeto do tipo *TESubstituicao*;

TConjuntoTermos: Classe responsável por armazenar o conjunto de termos candidatos e termos raíz que serão utilizados na construção das árvores;

TConjuntoTermosErrados: Classe responsável pelo armazenamento da lista de termos utilizados para correção ortográfica. Cada elemento da lista é um objeto da classe *TTermosErrados*;

TConjuntoTermosMascara: Classe responsável por armazenar o conjunto de termos identificados a partir de uma determinada máscara. Cada elemento do conjunto é um objeto do tipo *TETermosMascara*;

TEBigrama: Classe entidade de bigramas;

TEEstatisticaCFU: Classe entidade responsável pelo armazenamento de informações de estatística de CFU;

TEEtiquetaCorreta: Classe entidade responsável pelo armazenamento de etiquetas corretas, usada na correção de etiquetas;

TEFraseUnica: Classe entidade responsável pelo armazenamento de uma determinada frase única;

TEMascara: Classe entidade responsável por armazenar uma máscara;

TEPadronizacaoGeral: Classe entidade responsável por armazenar as informações de padronização relacionadas à transformação para minúsculo e remoção de acentuação;

TEProjeto: Classe entidade responsável pelo armazenamento de informações do projeto tais como endereços das pastas e arquivos;

TESeqTermos: Classe utilizada como classe entidade para armazenar termos que são inseridos no dicionário;

TEStopword: Classe entidade responsável pelo armazenamento de uma *stopword*;

TESubExpReg: Classe entidade responsável por armazenar informações de padronização relacionadas à substituições por expressões regulares;

TESubstituicao: Classe entidade responsável pelo armazenamento de informações de substituições;

TETermo: Classe entidade responsável pelo armazenamento de termos que são identificados e definidos como raíz;

TETermosMascara: Classe entidade responsável pelo armazenamento de termos identificados por meio da aplicação de máscaras;

TETrigrama: Classe entidade responsável pelo armazenamento de trigramas;

TEUnigrama: Classe entidade responsável pelo armazenamento de unigramas;

TNConjuntoEtiquetaCorreta: Classe de negócio do conjunto de correção de etiquetas;

TNConjuntoFrasesUnicas: Classe de negócio do conjunto de frases únicas;

TNConjuntoMascara: Classe de negócio do conjunto de máscara;

TNConjuntoNGramas: Classe de negócio do conjunto de n-gramas;

TNConjuntoSeqTermos: Classe de negócio do conjunto de seqüência de termos utilizados pelo dicionário;

TNConjuntoStopword: Classe de negócio do conjunto de *stopwords*;

TNConjuntoSubstituicao: Classe de negócio do conjunto de substituições;

TNConjuntoTermos: Classe de negócio do conjunto de termos;

TNConjuntoTermosErrados: Classe de negócio do conjunto de termos errados;

TNConjuntoTermosMascara: Classe de negócio do conjunto de termos identificados por meio da aplicação de máscaras;

TNEstatisticaCFU: Classe de negócio de estatísticas do CFU;

TNPadronizacaoGeral: Classe de negócio da padronização geral;

TNProjeto: Classe de negócio do projeto;

TNSubExpReg: Classe de negócio de substituição de expressões regulares;

TPersistencia: Classe responsável pelo armazenamento e recuperação de informações de arquivos no formato XML. Quando é realizada a recuperação de informações, elas são passadas para a camada de negócio; quando é realizado o armazenamento, a classe de persistência recebe as informações e armazena nos arquivos XML.

Apêndice B

Descrição das Classes Implementadas na Linguagem Perl

A seguir são definidas as principais classes utilizadas pelos *scripts*.

TDocumento: Classe responsável pela manipulação de documentos do tipo texto;

TProjeto: Classe responsável pelo armazenamento e recuperação de informações de projeto;

TStopword: Classe responsável pela aplicação do algoritmo de remoção de *stopwords*;

TNormalize: Classe responsável pela transformação de informações do documentos para minúsculo;

TConjuntoFrasesUnicas: Classe responsável pela geração de conjunto de frases únicas a partir de um conjunto de documentos e extração de estatísticas do CFU;

TTokenizer: Classe responsável pela toquenização das informações dos documentos;

TSentencer: Classe responsável pela aplicação da ferramenta de sentenciamento das informações nos conjuntos de documentos;

TUnigrama: Armazena lista de unigramas;

TBigrama: Armazena lista de bigramas;

TTermosErrados: Classe responsável por armazenar lista de termos errados e termo de sugestão para correção ortográfica;

TConjuntoTermosErrados: Classe responsável pela armazenamento da lista de objetos da classe *TTermosErrados*;

TConjuntoTermos: Classe responsável pelo armazenamento de lista de termos correspondentes à objetos do tipo *TUnigrama*, *TBigrama*, etc.

TControleTermo: Classe responsável pela aplicação das heurísticas sobre listas de unigrama e bigramas;

TControleTermoNgrama: Classe responsável pelo preenchimento de objetos do tipo *TConjuntoTermos*;

TArvore: Classe auxiliar utilizada pelo *script* geraArvore.pl para extração de informações das listas de n-gramas e pesquisa nas listas de n-gramas;

TDicionario: Classe responsável pela inserção de seqüências de termos correspondentes aos atributos na árvore trie e pelo algoritmo de construção da tabela atributo-valor. Também é responsável pelo casamento de padrões entre o dicionário e o conjunto de documentos;

XML::Simple: Classe responsável pela abertura de arquivos XML;

Text::Levenhtein: Classe responsável pelo cálculo da distância entre termos no *script* de correção ortográfica;

String::Tokenizer: Classe auxiliar de toquenização utilizada pela classe *TTokenizer*;

Tree::Trie: Classe que implementa a estrutura de dados árvore trie e utilizada para armazenar as seqüências de termos que são utilizadas no casamento de padrões que resulta no preenchimento da tabela atributo-valor.

Apêndice C

Descrição dos *Scripts* Implementados na Linguagem Perl

Os principais *scripts* implementados na linguagem Perl são apresentados a seguir.

geraCFU.pl: *Script* responsável pela geração do primeiro conjunto de frases únicas e pela toquenização e segmentação do conjunto de documentos original. Utiliza como classes auxiliares *TProjeto*, *TDocumento* e *TConjuntoFrasesUnicas*, *TTokenizer*, *TSentencer* e *XML::Simple*;
Entrada: Lista de arquivos para processar;
Saída: Conjunto de frases únicas;

padronizaLaudos.pl: *Script* responsável por aplicar a padronização sobre o conjunto de documentos original. Utiliza as classes auxiliares *TProjeto*, *XML::Simple*, *TStopword*, *TNormalize*, *TConjuntoFrasesUnicas* e *TDocumento*;
Entrada: Arquivos gerados pela interface de padronização e conjunto de documentos original;
Saída: Conjunto de documentos padronizados;

substituicao.pl: *Script* responsável pela aplicação de substituição sobre o conjunto de documentos. Esse *script* transforma as informações de substituição definidas pelo usuário para o formato adequado utilizado pela classe Text::RewriteRules[1] e aplica as substituições. Utiliza as classes auxiliares *TDocumento*, *TProjeto* e *TConjuntoFrasesUnicas*.
Entrada: Lista de substituições definidas pelo usuário e documentos padronizados;
Saída: Nova lista de documentos com as substituições aplicadas;

corretorPalavras.pl: *Script* responsável pela identificação de sugestões de correções automaticamente. Utiliza como classes auxiliares *TDocumento*, *TProjeto*, *TUnigrama*, *Text::Levenhtein*, *TTermosErrados* e *TConjuntoTermosErrados*;

[1] `http://search.cpan.org/~ambs/Text-RewriteRules-0.10/lib/Text/RewriteRules.pm`

Entrada: Lista de unigramas geradas a partir do conjunto de documentos padronizados;
Saída: Lista de sugestões de correções;

aplicaCorretorPalavras.pl: *Script* responsável por aplicar as correções dos termos com base no arquivo de correções definido pelo usuário. Utiliza como classes auxiliares *TDocumento*, *TProjeto*, *TUnigrama*, *TTermosErrados* e *TConjuntoFrasesUnicas*.
Entrada: Arquivo contendo lista de correção e o conjunto de documentos padronizado;
Saída: Conjunto de documentos padronizados e corrigidos;

lematiza.pl: *Script* responsável por aplicar a lematização sobre o conjunto de documentos padronizados e corrigidos. A principal função desse *script* é transformar todos os arquivos do conjunto de documentos para o formato utilizado pelo lematizador TREETAGGER e, depois de lematizados, retornar ao formato original. Utiliza como classes auxiliares *TProjeto*, *TDocumento* e *TConjuntoFrasesUnicas*.
Entrada: Conjunto de documentos padronizados e corrigidos;
Saída: Conjuntos de documentos lematizados;

etiquetaDocumentosMXPOST.pl: *Script* responsável por executar a etiquetação no conjunto de documentos utilizando a ferramenta MXPOST. Utiliza como classes auxiliares *TDocumento* e *TProjeto*.
Entrada: Documentos definidos para serem etiquetados;
Saída: Documentos etiquetados;

etiquetaCFUMXPOST.pl: *Script* responsável por executar a etiquetação no conjunto de frases únicas utilizando a ferramenta MXPOST. Utiliza como classes auxiliares *TDocumento* e *TProjeto*.
Entrada: Conjunto de frases únicas definido para ser etiquetado;
Saída: Conjunto de frases únicas etiquetado;

aplicaCorretorEtiquetas.pl: *Script* responsável pela aplicação da correção de etiquetas no CFU. Utiliza como classes auxiliares *TDocumento* e *TProjeto*.
Entrada: Lista de termos com etiquetas corretas e o CFU;
Saída: CFU com etiquetas corretas associadas aos termos;

aplicaCorretorEtiquetasDocumentos.pl: *Script* responsável pela correção de etiquetas no conjunto de documentos. Utiliza como classes auxiliares *TDocumento* e *TProjeto*.
Entrada: Lista de termos com etiquetas corretas e o conjunto de documentos;
Saída: Conjunto de documentos com termos etiquetados corretamente;

casamentoMascara.pl: *Script* responsável por realizar a extração de termos do conjunto de documentos a partir de um conjunto de máscaras. Utiliza como classes auxiliares *TProjeto* e *TDocumento*.
Entrada: Lista de máscaras e conjunto de documentos etiquetados;
Saída: Lista de termos extraídos correspondentes às máscaras definidas;

geraTermosCandidatos: *Script* responsável pela geração de termos candidatos a partir de uma lista de unigramas, uma lista de bigramas e os parâmetros Alfa e Theta. Utiliza como classes auxiliares *TUnigrama*, *TBigrama*, *TConjuntoTermos* e *TControleTermo*.
Entrada: Arquivo de unigrama e arquivo de bigrama gerados a partir do conjunto de documentos etiquetados e parâmetros Alfa e Theta definidos pelo usuário;
Saída: lista de termos candidatos identificada por meio da aplicação de heurísticas;

geraArvore.pl: *Script* responsável pela geração da árvore a partir dos termos definidos pelo usuário como termos raíz. Utiliza como classes auxiliares *TDocumento*, *TControleTermoUnigrama*, *TControleTermo*, *TArvore* e *TProjeto*.
Entrada: Lista de termos raíz e número correspondente ao total de listas de n-gramas a serem pesquisadas;
Saída: Árvores correspondentes aos termos raíz de entrada;

geraAtributos.pl: *Script* responsável pela geração de atributos, a partir das árvores geradas, os quais irão fazer parte da tabela atributo-valor. Utiliza como classes auxiliares *TDocumento* e *TProjeto*
Entrada: Árvores geradas e número correspondente à poda que deverá ser aplicada na árvore no momento da identificação dos atributos;
Saída: Lista de atributos;

dic.pl: *Script* responsável pela construção e preenchimento da tabela atributo-valor. Utiliza como classe auxiliares *TDicionario*, *TProjeto* e *TDocumentos*;
Entrada: Lista de atributos (dicionário) e conjunto de documentos;
Saída: Tabela atributo-valor e arquivo contendo informações não processadas (*garbage*);