6
Framework proposto

6.1.
Definição

Um *framework* é um conjunto de componentes (classes e interfaces) que funcionam juntos para solucionar um determinado problema de software. Segundo (JOHNSON & FOOTE, 1988), um *framework* é uma aplicação reutilizável e semicompleta que pode ser especializada para produzir aplicações personalizadas.

A principal finalidade de um framework é proporcionar a sua reutilização em outros projetos, diminuindo a complexidade na implementação de softwares. Além disso, um *framework* determina a arquitetura de sua aplicação, ou seja, define a estrutura geral, sua divisão em classes e objetos e, consequentemente, as responsabilidades entre si, assim como a forma de colaborarem e o fluxo de controle.

Um *framework* possui as seguintes características:

- É composto por múltiplas classes ou componentes, cada um provendo uma abstração de um conceito particular;
- Define como as abstrações trabalham juntas para resolver um problema;
- Seus componentes são reutilizáveis;
- Organiza padrões em alto nível.

O *framework* desenvolvido nesta Tese para a Categorização de Textos possui aproximadamente cento e vinte classes e interfaces e quatro mil linhas de código.

6.2. Ambiente de desenvolvimento

O ambiente de software neste trabalho foi implementado na linguagem de programação **C#**[26]**.** C# é uma linguagem de programação orientada a objetos criada pela empresa Microsoft e faz parte da plataforma de desenvolvimento ".NET". Embora existam mais de vinte linguagens de programação suportadas pela tecnologia ".NET", a linguagem C#, baseada na linguagem C++ e Java, é considerada a linguagem símbolo da plataforma ".NET".

Com uma ideia semelhante a da plataforma Java, o ambiente de desenvolvimento de aplicações ".NET" permite que *softwares* sejam desenvolvidos para qualquer sistema operacional que suporte o .NET *Framework*.

Portanto, constituem justificativas para a escolha deste Ambiente de Desenvolvimento e Linguagem de Programação a orientação a objetos que permite fácil reuso, bem como grande organização do código e possibilidade de portabilidade para diversos sistemas operacionais.

6.3. Objetivos

O *framework* proposto é capaz de:

- Empregar técnicas de Mineração de Textos para o fornecimento de ferramentas que auxiliem a execução da tarefa de Categorização de Textos.
- Fornecer tratamento linguístico específico à Língua Portuguesa do Brasil, isto é, que faça proveito das ricas informações semânticas presentes em qualquer linguagem natural;
- Automatizar as etapas necessárias para a realização de Categorização de Textos, otimizando a combinação das técnicas de pré-processamento textual e a escolha dos algoritmos de Aprendizado de Máquina e seus respectivos parâmetros.

[26] Pronuncia-se C Sharp.

6.4. Coleta

Nenhum mecanismo de coleta de dados foi implementado, pois a coleção de documentos utilizada nos estudos de caso foi obtida por meio do *corpus* CETENFolha (item 6.6).

6.5. Pré-Processamento

6.5.1. Tokenização

O processo de Tokenização (seção 3.2.1) empregado no *framework* segue a metodologia de geração de *tokens* (ver Figura 12) proposta em (KONCHADY, 2006). Esta metodologia, fortemente apoiada na utilização de um dicionário de palavras, obtém *tokens* com alto valor semântico, conforme resultados obtidos nos experimentos realizados no Estudo de Caso. Esta metodologia é composta dos seguintes passos:

1. Geração simples de *tokens:* baseada no conjunto de *tokens* delimitadores, como espaço e fim de linha.
2. Identificação de abreviações: realizada com auxílio de dicionários desenvolvidos para este propósito. Todas as abreviações encontradas são substituídas pelos seus termos não contraídos.
3. Identificação de palavras combinadas: muito comum em nomes próprios de organizações. Em geral, são palavras separadas por símbolos como o "&": "Casa & Vídeo".
4. Identificação de símbolos de Internet: geralmente, símbolos de internet atendem às normas regras estabelecidas no momento de criação do serviço. Desta forma, o uso de expressões regulares

auxilia na identificação de tais símbolos (JARGAS, 2006). Alguns exemplos de símbolos de Internet são IP, *e-mails* e *URLs*.

5. Identificação de números: este processo é realizado pela verificação de conteúdo numérico em meio aos *tokens*. Números na forma extensa, quando identificados, são convertidos em formato numérico.
6. Identificação de *tokens* multivocabulares: realizado também com forte apoio de um dicionário de termos. Busca reunir em um único *token* palavras que, quando utilizadas em conjunto, transmitem ideias diferentes ou incompletas quando utilizadas separadas. Exemplos comuns são "bolsa de valores", "casa da moeda".

Além disso, para automatizar o processo de Identificação de *tokens* multivocabulares é realizado o cálculo do coeficiente de correlação entre os *tokens.* Essa medida indica a força e a direção do relacionamento linear entre duas variáveis aleatórias. Desta forma, *tokens* que apresentam alto grau de correlação são candidatos a *tokens* multivocabulares.

Bibliotecas que lidam com o processamento de linguagem natural contam com algoritmos de tokenização, como é o caso do Lucene e do OpenNLP com o *WhiteSpaceTokenizer* que divide o texto em *tokens* de acordo com os espaços em branco, ou seja, sequência de caracteres adjacentes entre os espaços em branco formam *tokens* (The Apache Software Foundation, 2011). Essa abordagem é simplista demais.

6.5.2. Análise/Remoção de *stopwords*

O processo de remoção de *stopwords* (item 3.2.2) possui o objetivo de reduzir a grande dimensionalidade de dados textuais através da remoção de palavras de baixo poder discriminatório.

O *corpus* utilizado nos estudos de casos é o CETENFolha (item 6.6). Essa coleção dispõe de três listas padrões de *stopwords*, elaboradas pela entidade

organizadora do *corpus*. Cada uma das listas possui cem, duzentos e trezentos termos considerados irrelevantes. A utilização de uma lista ou de outra deve ser baseada nos resultados obtidos.

Uma lista de aproximadamente cem *stopwords* da Língua Portuguesa é exibida na Tabela 10 e pode ser encontrada no endereço ao lado: "http://linguateca.di.uminho.pt/Paulo/stopwords/folha.MF100.txt".

Tabela 10 - Lista de cem *stopwords* utilizadas na etapa de Pré-processamento

a	da	em	já	nos	quando	Sua
à	das	entre	O RF D (	R	TX H	7D PE
DLQ G !	GH	HU D	PDL R	R QW H P	T XH P	7HP
DQ R	G HSR !	HVW £	PDL V	RV	U	7HU
D QR V	G HY H	HV WD (	PD V	RX	UL R	7R GR V
DR	GL D	HVW ¥ !	P H UF	S Dfl V	V¥ R	7U< V
DR V	GLV V !	HX	P H V P	S DU D	VH	8 P
D S HQ	GL]	IR L	PL O	S DX O R	V H JX	8 PD
DV	GR	I RO K D	PL OK (	S HO D	VH P	8 V
¢V	GRL V	I RU D !	PXL W !	S HO R	VH U	9DL
DW «	GR V	J R YH V	P XQ G	S H VV R	V HU £	
E UDV L	H	J U DQ	QD	S RG H	VH X	
FR P	«	K£	Q¥ R	SR U	V HX V	
F RP R	HO D	KRM H	QD V	S R UT X	V	
F RQW U	HOH	L VV R	QR	S UH V LC	V HE U !	

A existência dessas três listas de *stopwords*, criadas especificamente para o *corpus* utilizado nos estudos de casos, não justificou a adoção de outras listas pré-concebidas. No ***Snowball***[27]é possível encontrar *stoplists* para vários idiomas, incluindo o português.

Outro detalhe relacionado ao processo de remoção de *stopwords* é que a *stoplist* ideal deve ser construída levando em consideração o domínio do problema. Por exemplo, no caso de um sistema de Mineração de Textos para a área médica, termos como "exame" e "medicamento" poderiam ser incluídos na lista de *stopwords*, visto que, estas palavras, provavelmente, possuirão pouco poder discriminatório em relação a outros termos.

[27] Disponível em http://snowball.tartarus.org/

Portanto, assim que for obtida a coleção de documentos sob a qual será realizado o processo de Mineração de Textos, mesmo após a remoção das *stopwords* consideradas padrão, é interessante que seja analisada a distribuição das frequências de ocorrência dos termos que estão presentes na coleção. De acordo com (SALTON & BUCKLEY, 1988), termos com elevada frequência em muitos documentos constituem elementos de pouco poder discriminatório.

Portanto, além das três *stoplists*, o *framework* dispõe de mecanismo para análise de termos muito frequentes e com pouco caráter discriminatório, como citado acima. Esse mecanismo, simples, é baseado na seleção de termos com muita frequência nos documentos. Geralmente, esses termos são apresentados ao usuário para que ele dê decisão final sobre remover ou manter tais termos.

Para que o processo de seleção de *stopwords* relacionadas ao domínio ocorra sem intervenção humana, é preciso definir um valor de corte que será utilizado para remover automaticamente todos os termos que possuem frequência maior do que o valor definido. O valor adotado para eliminação de termos no *framework* é de oitenta por cento de frequência (SILVA A., 2007).

6.5.3. Processamento de Linguagem Natural

O uso de técnicas de Processamento de Linguagem Natural em Mineração de Textos tem o objetivo de identificar a real importância de cada termo em determinados contextos, possibilitando um ganho na qualidade dos resultados produzidos.

6.5.3.1. Identificação de Classes Gramaticais

Abordada desde a década de 50, a tarefa de *Part of Speech* (item 3.2.3.2) já foi testada por praticamente todos os métodos e algoritmos de Aprendizado de Máquina. Diversas outras tarefas de mais alto nível são dependentes desta tarefa base:

- Distinção de significados em aplicações de Recuperação de Informação;
- Solução de casos simples de ambiguidade;
- Facilita o entendimento/predição (*speech recognition*) de uma sentença:
    - Pronomes possessivos são seguidos por substantivos;
    - Pronomes pessoais são seguidos por verbos.
- Beneficia o processo de *text-to-speech*: fornecendo dados sobre a entonação com que uma determina palavra deve ser falada:
    - INsult(substantivo) / inSULT (verbo)
    - OBject (substantivo) / obJECT (verbo)

Entende-se por classe gramatical como a forma de classificação de um termo segundo seu significado dentro de uma sentença (CEGALLA, 2005). A Nomenclatura Gramatical Brasileira (NGB) divide os vocábulos em dez classes gramaticais, com a seguinte distribuição, ilustrada na Figura 29:

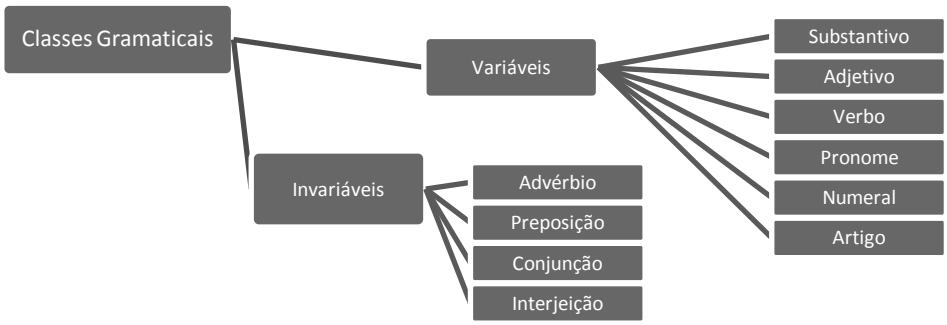

Figura 29 – Classes gramaticais segundo a NGB

Part of Speech Tagging ou Identificação de Classes Gramaticais é a primeira tarefa de Processamento de Linguagem Natural, baseada em Estatística, e possui como objetivo a identificação da classe gramatical correta de um vocábulo de acordo com o seu contexto, ou seja, a identificação de estruturas implícitas de um texto. A identificação da classe sintática de *tokens* é encarada na literatura como um típico problema de Classificação.

O problema a ser resolvido não é trivial, pois, muitas palavras possuem várias classes gramaticais, ou seja, quando fora de contexto, estas palavras

possuem ambiguidade em relação a como devem ser classificadas. Resolver as ambiguidades é a essência do problema de *Part of Speech Tagging*. Por exemplo, nas sentenças abaixo, exemplificadas na Tabela 11, uma mesma palavra pode assumir quatro classes gramaticais distintas:

The back door.	Adjetivo
On my back.	Substantivo
Win the voters back.	Advérbio
Promised to back the bill.	Verbo/Infinitivo

Tabela 11 - Exemplos ambíguos de identificação de classes gramaticais

O etiquetador mais conhecido para o Português brasileiro é o PALAVRAS, baseado em regras com *Constraint Grammar* de (BICK, 2000), e possui acurácia superior a 99%, mas não está disponível para ser embutido em aplicações.

No *framework*, uma das formas de seleção de termos será baseada nas classes gramaticais destes. Além disso, a heurística desenvolvida para a lematização necessita que verbos sejam identificados para que um léxico verbal construído nesta Tese (item 6.5.3.2) seja utilizado adequadamente. Portanto, é necessário implementar um Identificador de Classes Gramaticais no *framework*.

Em (MARKHAM & RUS, 2005), é proposto um *BaseLine System* que, embora de simples formulação, obtém taxa de Acurácia superior a 90% (sob o ***Corpus Penn TreeBank***[28], da língua inglesa). A heurística deste *BaseLine System* é simples e segue as seguintes etapas:

1. Treinamento: construção de um dicionário que armazena para cada vocábulo do *corpus* a lista de todas as classes gramaticais que cada vocábulo pode assumir e a frequência com que cada uma dessas classes gramaticais ocorre para cada vocábulo.
2. Recall » atribuir a cada palavra de uma nova sentença a classe gramatical que mais ocorre para a mesma (consulta ao dicionário). Caso a palavra não tenha sido previamente adicionada ao dicionário, isto é, seja desconhecida, atribuir à classe de substantivo.

[28] Disponível em http://www.cis.upenn.edu/~treebank/

O *BaseLine System* implementado neste trabalho segue as mesmas heurísticas descritas acima. Entretanto, a taxa de acurácia obtida para a língua Portuguesa, em torno de 70%, leva a necessidade de outra alternativa: Aprendizagem de Máquina.

Para a solução desse problema baseada em Aprendizagem de Máquina, o primeiro passo deste processo foi a preparação dos *data sets*. Analisando o conjunto de treinamento, este foi dividido em dois subconjuntos:

- O primeiro conjunto é composto apenas por vocábulos que não possuem ambiguidade de classe gramatical, ou seja, possuem apenas uma classe gramatical em todo o corpus.
- O segundo conjunto é composto por vocábulos que possuem mais de uma classe gramatical atribuída aos mesmos.

Esta separação foi realizada para que as tags ausentes de ambiguidade não sejam enviadas para o treinamento da SVM, já que um simples *BaseLine* irá obter 100% de Acurácia sobre este conjunto, o que irá diminuir o tempo de treinamento e aumentar o desempenho dos resultados obtidos pela SVM.

A função *kernel* escolhida foi a Linear, que embora de menor complexidade, obtém resultados satisfatórios com muito menos tempo de treinamento. A otimização do parâmetro *C* adotou a heurística gulosa, assumindo, respectivamente, os seguintes valores: 0.1, 1, 10, 100.

A modelagem dos dados segue a metodologia de *sliding window* de três termos e é ilustrada na Tabela 12:

Tabela 12 - Modelagem dos dados baseada em *sliding window*

Termos que antecedem e sucedem o termo a ser predito	$word_{-3}$
	$word_{-2}$
	$word_{-1}$
	$word_{0}$
	$word_{+1}$
	$word_{+2}$
	$word_{+3}$

Classe dos termos que antecedem e sucedem o termo a ser predito (fornecida pelo BaseLine)	pos_{-3}
	pos_{-2}
	pos_{-1}
	pos_0
	pos_{+1}
	pos_{+2}
	pos_{+3}
Começa com letra maiúscula?	sim/não
Comprimento da palavra	inteiro

O software utilizado para o treinamento e teste da SVM foi o SVMTool, proposto por (GIMÉNEZ, J. & MÀRQUES, 2004). Possui como principal vantagem a escolha da direção de *tagging*, permitindo que a mesma possa ser feita da esquerda para direita (padrão), ou da direita para esquerda, ou para ambas as direções. É divido nos quatro módulos citados abaixo:

- SVMT-learner [treinamento dos classificadores SVM];
- SVMT-tagger [POS-tagging de uma entrada qualquer];
- SVMT-evaluator [avaliação dos resultados de *tagging*];
- SVMT API [API para uso do SVMT-tagger treinado].

Outras vantagens deste software são o tempo de treinamento muito inferior a outros semelhantes (WEKA, por exemplo) e a classificação multiclasses automática (método de redução da SVM em problemas binários: "um contra todos").

O resultado geral obtido é de 91% para a taxa de acurácia. Para a necessidade da heurística de lematização, que necessita apenas identificar se um termo é, ou não, um verbo, o resultado obtido foi de 95%. No último caso, apenas uma classe gramatical foi considerada: verbo ou não verbo.

6.5.3.2. Lematização

Lematização ou *stemming* (item 3.2.3.4) é o processo de reduzir ao radical original palavras derivadas ou flexionadas deste. O principal objetivo da utilização

de um processo de *stemming* é reduzir a grande dimensionalidade das aplicações de Mineração de Textos, pois, com a remoção de prefixos e sufixos de palavras derivadas de um mesmo radical, e que, antes, seriam consideradas como *tokens* distintos, obtém-se um único *token* para a representação de todas elas.

Apesar dos benefícios acima, algoritmos de *stemming* também possuem desvantagens. Casos de *overstemming* e *understemming* são comuns em todos os algoritmos propostos (SILVA A. A., 2007). Todos os algoritmos de *stemming* são baseados em regras, e uma das principais causas de erro são os verbos irregulares, que geralmente são exceções às regras.

Para minimizar esse problema, um léxico de aproximadamente 10 mil verbos e suas conjugações foi obtido em parceria com o Departamento de Engenharia de Software da PUC-Rio para que, quando um verbo fosse identificado, a redução à forma canônica desse verbo fosse realizada por consulta ao léxico, ignorando os algoritmos de *stemming* nesses casos.

A utilização dessa abordagem minimizou os erros de *overstemming* e *understemming* em 40% para os termos verbais.

Para outras classes gramaticais, serão aplicados os algoritmos de *stemming*. A seleção dos algoritmos será realizada automaticamente baseada nos resultados obtidos na Categorização. Inicialmente, os algoritmos de *stemming* implementados no *framework* foram: Stemmer S, Método de Porter (adaptado ao Português) e Método de Lovins. Em seguida foi incluído também o RSLP Stemmer (Removedor de Sufixos da Língua Portuguesa), criado por (ORENGO & HUYCK, 2001), que constitui um dos mais modernos algoritmos de *stemming* concebidos para a língua Portuguesa.

6.5.3.3. Thesaurus

Outra técnica empregada é a utilização de um dicionário de relações hierárquicas entre termos, o dicionário *Thesaurus* (item 3.3.2), para substituir termos que possuam valores semânticos semelhantes ou relacionados, mas, são grafados de maneira distinta, sejam estes sinônimos ou exemplos de sinonímia. A

utilização deste dicionário evita equívocos ao realizar o cálculo de frequência destes termos. Para exemplificar, pode-se supor que em determinado documento sobre animais, encontramos dois termos de grande frequência: cão e cachorro. Porém, como possuem grafias distintas, estes termos serão computados separadamente, ainda que pelo fato de transmitirem a mesma ideia, deveriam ser considerados como somente um termo e ter frequências e outras métricas somadas. Outro benefício proporcionado por estes dicionários é relação entre termos. Retornando ao exemplo acima, com a ajuda do dicionário, verifica-se que cachorro e cão possuem valores semânticos aproximados e pertencem a uma categoria superior: a de animais de estimação que também pertence à categoria de animais. O dicionário *Thesaurus,* construído durante a execução deste trabalho, e que foi utilizado neste estudo de caso possui cerca de 13 mil termos preferenciais e 17 mil termos não preferenciais, todos classificados em dezenas de categorias. A estrutura de dados utilizada na construção deste dicionário é ilustrada na Figura 30.

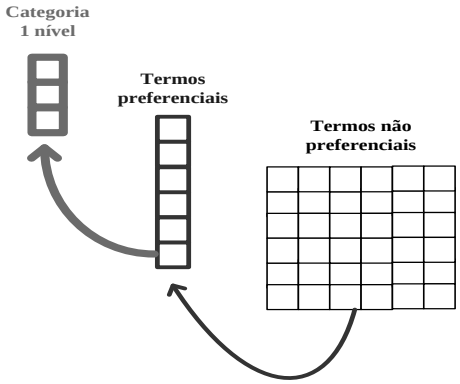

Figura 30 - Estrutura do dicionário Thesaurus utilizado no Sistema de MT

No contexto de Thesaurus, conforme citado em (BARROS, 2010), há também a WordNet.BR com aproximadamente 44 mil termos que possuem relações de antonímia e sinonímia. Está disponível por meio do Thesaurus Eletrônico para o Português do Brasil[29] (TeP). A base de dados desse dicionário pode ser utilizada para estudos linguísticos.

Esta base foi também incorporada pelo *framework* desenvolvido nesta Tese.

[29] Disponível em http://www.nilc.icmc.usp.br/tep2/index.htm

Cerca de trinta e dois por cento dos *tokens* foram consultados e ou substituídos por termos do dicionário. Além disso, com a utilização deste dicionário, houve redução média de sete por cento no número de *tokens* encontrados (*tokens* multivocabulares, etapa 6, item 6.5.1).

6.5.4. Redução de características

Para uma representação adequada dos textos faz-se necessária a seleção dos termos mais representativos, pois em virtude do tamanho destes e de sua quantidade de *tokens*, os mesmos podem ser intratáveis pelos algoritmos de classificação. Este processo é conhecido como redução de características.

A redução de características pode ser do tipo:

- Global: consiste na escolha dos termos mais importantes para a coleção.
- Local: consiste na escolha de um conjunto de termos para cada categoria.

A natureza da redução é realizada por meio da seleção das características dos documentos da coleção, ou seja, as características selecionadas são um subconjunto das características originais. Esta redução geralmente é realizada através da seleção das melhores características, de acordo com algum critério. A grande dificuldade dessa abordagem é selecionar um subconjunto de termos que possa fazer uma discriminação adequada entre as várias categorias, e ao mesmo tempo, ser pequeno o suficiente para que possa ser utilizado pelo classificador. As técnicas mais utilizadas para este fim são baseadas na frequência de palavras (item 4.3.1.3), como TF/IDF, Ganho de informação e de Escore de Relevância (WIENER, 1995).

A escolha do conjunto de características possui enorme impacto no processo de Categorização. O *framework* é capaz de selecionar os termos mais importantes de cada documento segundo as duas abordagens. A escolha de uma ou outra abordagem é determinada pelo resultado obtido no processo de otimização da Categorização de Textos.

6.5.5. Indexação

Indexação é fase responsável por criar estruturas de dados denominadas índices, capazes de permitir que uma consulta seja realizada sem que seja necessário analisar toda uma base de dados (MANNING, RAGHAVAN, & SCHÜTZE, 2007).

Atualmente, a técnica mais utilizada para a indexação textual é a de **índices invertidos**[30] (KONCHADY, 2006). Essa técnica é suportada no *framework* proposto por meio da utilização da biblioteca **Lucene**[31]. O processo de tokenização é realizado com a heurística desenvolvida no *framework* (item 6.5.1).

6.5.6. Classificadores implementados

O *framework* dispõe de três classificadores para realizar a atividade de classificação: *k-NN*, SVM e Classificador Bayesiano.

A implementação do algoritmo *k-NN* permite que as seguintes métricas possam ser usadas para calcular a distância entre cada instância: Distância Euclidiana e Cálculo do Cosseno.

A implementação do SVM é baseada na versão implementada por (JOACHIMS, 1998) chamada de SVM^{light}, disponível em seu próprio endereço na Internet[32].

A implementação do Classificador Bayesiano é baseada na versão disponível em Java no software Weka. A classe *NaiveBayes* foi reescrita para C#.

---

[30] Recebem essa denominação por inverter a hierarquia da informação. No lugar de uma lista de documentos contendo termos, tem-se uma lista de termos referenciando documentos.

[31] Disponível em http://lucenenet.apache.org/

[32] http://svmlight.joachims.org

6.5.7.
Técnicas de combinação de classificadores

O método de escolha por voto majoritário está implementado como premissa para toda operação de classificação que utiliza como classificador o SVM, pois este é um classificador binário.

Além disso, também foi implementada a votação em que pesos são atribuídos aos classificadores (votação com pesos); ou seja, o voto de cada classificador é relativo ao seu peso, para a decisão final. O peso é proporcional ao índice de acerto de cada classificador na categoria.

Para categorias em que um determinado classificador é muito eficiente, utiliza-se um mecanismo de seleção dinâmica de classificadores denominado *Best By Class*.

Além disso, o uso de *bagging* é natural, pois diversas heurísticas de pré-processamento são combinadas, produzindo alterações na forma de representação dos documentos.

6.6.
Corpus

A primeira etapa (ver 3.1) do processo de Mineração de Textos é a coleta. Essa etapa é responsável por obter os documentos que farão parte da coleção de documentos.

O processo de Categorização de Textos constitui um paradigma de Aprendizagem de Máquina supervisionado, portanto necessita que ao apresentar a instância de treinamento, esta além de conter os atributos que a representem, deve conter também a resposta desejada (categoria) para a categorização da mesma.

Na língua inglesa, diversos *corpora etiquetados são* utilizados como *benchmark* de Categorização de Textos. Temos como exemplo o *corpus* **Reuters-215783**[33] que é o mais utilizado em todo o mundo nesta atividade. Na língua

[33] Disponível em http://www.daviddlewis.com/resources/testcollections/reuters21578/

portuguesa do Brasil, o *corpus* que predomina nos estudos sobre Categorização de Textos é o CETENFolha, conforme mostrado nos trabalhos relacionados.

CETENFolha, Corpus de Extratos de Textos Eletrônicos NILC/Folha de S. Paulo, é um corpus de cerca de vinte e quatro milhões de palavras em Língua Portuguesa do Brasil, criado pelo projeto de Processamento Computacional do Português com base nos textos do jornal Folha de S. Paulo. Este corpus possui aproximadamente 341 mil porções de textos, classificados por semestre e caderno do jornal do qual provém. Informações adicionais sobre o corpus CETENFolha são apresentadas na Tabela 13.

Tabela 13 - Informações adicionais sobre o CETENFolha

Quantidade de palavras	24 milhões
Idioma	Português (Brasil)
Origem	Textos publicados no Jornal Folha de S. Paulo
Período de Publicação	1994
Quantidade de edições	365
Classificado?	Pelo tipo de caderno a que pertence a notícia
Etiquetado?	Não

A escolha desse *corpus* dispensa a execução da etapa de coleta de dados. Para o estudo de casos desta Tese, os documentos que serviram como base do experimento foram os textos jornalísticos obtidos do *corpus* CETENFolha. Desta forma, foram selecionados documentos textuais deste *corpus*, ou seja, notícias, que possuem mais de sessenta termos para o estudo de casos. Será utilizado como categoria dos textos obtidos desse corpus o caderno de onde cada um provém.

Cada texto deste arquivo possui uma quantidade considerável de metadados associados, como identificador do texto, título, o próprio texto, semestre, caderno no jornal e autor da notícia. A categoria a que cada notícia pertence está definida em *"cad"*. Todas as outras informações fornecidas pelos metadados foram ignoradas e não fazem parte do processo de Categorização.

Na Figura 31 é possível visualizar um exemplo de texto jornalístico, em seu formato original, presente neste corpus. As *tags* “<ext>” “</ext>” delimitam o início e fim de cada notícia ou documento; as *tags* “<t>” e “</t>” delimitam o título da notícia; as *tags* “<a>” e “” delimitam o autor da mesma; cada

parágrafo é identificado pelas *tags* “ ” e “ ”; e cada sentença da notícia fica compreendida entre as *tags* “<s>” e “</s>”.

```
<ext id=1 cad="Opinião" sec="opi" sem="94a">
 <s>
  <t> PT no governo </t>
 </s>
 <s>
  <a> Gilberto Dimenstein </a>
 </s>
 <p>
  <s> BRASÍLIA Pesquisa Datafolha publicada hoje revela um dado supreendente:
recusando uma postura radical, a esmagadora maioria (77%) dos eleitores quer o PT
participando do Governo Fernando Henrique Cardoso . </s>
  <s> Tem sentido -- aliás, muitíssimo sentido . </s>
 </p>
 <p>
  <s> Muito mais do que nos tempos na ditadura, a solidez do PT está, agora,
ameaçada . </s>
  <s> Nem Lula nem o partido ainda encontraram um discurso para se diferenciar . </s>
  <s> Eles se dizem oposição, mas ainda não informaram o que vão combater . </s>
  <s> Muitas das prioridades do novo governo coincidem com as prioridades do PT . </s>
 </p>
</ext>
```

Figura 31 - Exemplo de documento do corpus CETENFolha

Removendo as *tags* e excluindo as informações adicionais de metadados, o que será utilizado no processo de categorização é transcrito abaixo:

> PT no governo
>
> BRASÍLIA Pesquisa Datafolha publicada hoje revela um dado supreendente: recusando uma postura radical, a esmagadora maioria (77%) dos eleitores quer o PT participando do Governo Fernando Henrique Cardoso .
>
> Tem sentido -- aliás, muitíssimo sentido .
>
> Muito mais agora do que nos tempos da ditadura, a solidez do PT está, agora, ameaçada .
>
> Nem Lula nem o partido ainda encontraram um discurso para se diferenciar .
>
> Eles se dizem oposição, mas ainda não informaram o que vão combater .
>
> Muitas das prioridades do novo governo coincidem com as prioridades do PT .

Os cadernos de notícias (categorias) selecionados foram os de esporte, imóveis, informática, política e turismo. Optou-se por esses cadernos para que os resultados obtidos pudessem ser comparados com outros resultados obtidos, como em (CAMARGO, 2007) e (LINDEN, 2008).

6.7.
Assistência Inteligente

O *framework* dispõe de uma base de conhecimentos sobre Categorização de Textos baseada em (BORKO & BERNICK, 1963), (SEBASTIANI, 2002) (SHOLOM, INDURKHYA, ZHANG, & DAMERAU, 2005), (KONCHADY, 2006) e (KUDO & MATSUMOTO, 2004) que relatam as principais heurísticas utilizadas neste paradigma. (JOACHIMS, 1998) provê informações sobre a utilização de SVM e as técnicas de pré-processamento adequadas para Categorização de Textos com SVM. (BAEZA-YATES & BERTIER, 1999) relata as abordagens de pré-processamento para Recuperação de Informação, como os modelos de representação de documentos e as métricas utilizadas para representar a importância de cada termo. E os trabalhos relacionados no item 1.3 abordam diversas técnicas utilizadas para Categorização de Textos em Português.

A construção dessa base de conhecimento permite ao *framework* decidir as melhores opções de pré-processamento e ajustes de parâmetros dentre as disponíveis na ferramenta, reduzindo desta forma, o espaço de busca por uma solução ótima. Contudo, há novas abordagens às tarefas de pré-processamento desenvolvidas nesta Tese e que, portanto, nunca foram avaliadas anteriormente.

Foi elaborado um fluxo das etapas que devem ou podem ser seguidas no processo de Categorização de Textos para cada um dos algoritmos de classificação. Além disso, os valores desejáveis para a variação dos parâmetros de cada algoritmo de classificação são especificados.

Por exemplo, para o algoritmo *k-NN*, com base na literatura, os valores do parâmetro *k* devem variar entre um e quinze (SEBASTIANI, 2002). Para evitar a ocorrência de empate e ser necessária uma heurística adicional para a seleção da categoria predita nesses casos, opta-se por utilizar valores ímpares para *k*, portanto *k* deve variar sempre em *+2* ou *-2* a partir de seu valor inicial no intervalo especificado. Além disso, quatro tipos de medidas são utilizados para calcular a distância entre uma instância e seus vizinhos: Euclidiana, Cosseno, Jaccard e

Manhattan. Essas medidas são especificadas como funções na ordem de execução preferida.

KNN	
Parâmetros	Tipo
Parâmetro k	Numérico
Distância	Função

Valor mínimo	Valor máximo	Valor inicial	Incremento	Decremento
1	15	5	+2	-2

Função	Definição	Ordem execução
1. Euclidiana	...	1
2. Cosseno	...	2
3. Jaccard	...	3
4. Manhattan	...	4

Figura 32 - Exemplo de modelagem de execução para o *k*-NN

O planejamento das ações é modelado como um uma lista de tarefas que possui precedência de ações. A Tabela 14 exibe a modelagem do plano de algumas tarefas do *framework*. A precedência de ações delimita em que momento uma tarefa pode ser executada:

- Tarefas que possuem uma ou mais tarefas como precedentes podem ser executadas com os resultados distintos de cada uma das tarefas precedentes. Por exemplo, a tarefa *bb* pode ser executada com a saída do processamento das tarefas *b* e *c*. Isso constitui dois caminhos de execução para a tarefa *bb*.

Tabela 14 - Planejamento de ações

Id	Etapa	Tarefa	Antecedente	Tipo Entrada	Tipo Saída
a	*Pré-Processamento*	*Início de etapa*	*N/D*	*N/D*	*N/D*
b	Pré-Processamento	Tokenização	a	*A0*	*A1*
c	Pré-Processamento	Remoção de stopwords	b	*A1*	*A1*
d	Pré-Processamento	Remoção de stopwords (domínio)	c	*A1*	*A1*
z	*Pré-Processamento*	*Fim de etapa*	*N/D*	*N/D*	*N/D*
aa	*PLN*	*Início de etapa*	*N/D*	*N/D*	*N/D*
bb	PLN	POS tagging (verbo/não verbo)	b ou c	*A1*	*A2*
cc	PLN	POS tagging (completo)	b ou c	*A1*	*A2*

dd	PLN	Lematização verbal	bb	*A2*	*A1*
ee	PLN	Lematização PORTER	bb	*A2*	*A1*
ff	PLN	Lematização PORTER	bb	*A2*	*A1*
gg	PLN	Lematização LOVINS	bb	*A2*	*A1*
hh	PLN	Lematização RSLP	bb	*A2*	*A1*
...	...	...	...	...	...
zz	*PLN*	*Fim de etapa*	*N/D*	*N/D*	*N/D*

Um construtor de ações foi desenvolvido no *framework* para que o planejamento de ações sempre seja válido. Para isso, as informações sobre precedência e tipos de entrada e saída são utilizadas. Os tipos de entrada e saída definem o formato dos dados que são processados pelas tarefas. Por exemplo, o tipo de entrada *A0* constitui um conjunto de cadeias de caracteres (documento textual). Desta forma, observando a Tabela 14 é possível concluir que somente a tarefa *b* está apta a trabalhar com a informação neste formato.