Uma Metodologia para Auxiliar na Seleção de Atributos Relevantes usados por Algoritmos de Aprendizado no Processo de Classificação de Textos

Claudia A. Martins[1,2] **Maria Carolina Monard**[1] **Edson T. Matsubara**[1]

[1] Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
13560-970, São Carlos, SP, Brazil
e-mail: {cam, mcmonard, edsontm}@icmc.sc.usp.br

[2] Universidade Federal de Mato Grosso
Instituto de Ciências Exatas e da Terra
Departamento de Ciência da Computação
78060-900, Cuiabá, MT, Brazil

Abstract

Existing learning algorithms expect their input to be presented in terms of constrained set of attributes. Thus, learning algorithms cannot be applied directly to the Text Mining task related to text classification which consists in automatically classifying text documents based on their contents.

In order to apply learning algorithms to text classification it is necessary to process the text documents into some form that is acceptable to the chosen algorithm. As every word in a document may be treated as an attribute, the selection of these attributes plays an important role on how well the learning algorithm can generalize.

This work proposes a methodology to select attributes from texts decomposed into words (stems) using the bag-of-words approach, considering the behavior of the learning algorithm used for text classification. The methodology is illustrated using three different algorithms on a set of documents.

Keywords: Text Mining, Preprocessing, Inductive Learning.

Resumo

Os algoritmos de aprendizado existentes utilizam como entrada um conjunto de exemplos descritos como vetores de atributos. Assim, os algoritmos de aprendizado não podem ser aplicados diretamente a tarefas de Mineração de Textos, relacionadas à classificação de textos, que consistem em classificar automaticamente documentos textuais baseado em seu conteúdo.

Na aplicação de algoritmos de aprendizado em classificação de textos é necessário transformar os documentos textuais em um formato aceito pelo algoritmo escolhido. Considerando que toda palavra em um documento pode ser tratado como um atributo, a seleção desses atributos tem uma função importante em quão bem um algoritmo de aprendizado consegue generalizar.

Neste trabalho é proposta uma metodologia para selecionar atributos de textos, decompostos em palavras (*stems*) usando a abordagem *bag-of-words*, considerando o comportamento do algoritmo de aprendizado usado na classificação de textos. A metodologia é ilustrada utilizando três diferentes algoritmos em um conjunto de documentos.

Palavras Chaves: Mineração de Textos, Pré-processamento, Aprendizado Indutivo.

1 Introdução

A tarefa de categorização automática de textos, ou documentos, geralmente emprega técnicas de Aprendizado de Máquina para induzir classificadores de um conjunto de textos rotulados. Entretanto, essa não é uma tarefa trivial devido, principalmente, a forma não estruturada dos textos e a alta dimensão do espaço de possíveis atributos. A transformação dos textos em um formato estruturado, de maneira que possam ser submetidos a algoritmos de Aprendizado de Máquina, tem uma influência fundamental em quão bem um algoritmo de aprendizado consegue generalizar [13]. Essa transformação consiste, basicamente, em identificar e selecionar os atributos a serem utilizados para representar os textos, bem como atribuir valores a esses atributos. Uma outra questão relacionada à tarefa de categorização de textos é a escolha do algoritmo de aprendizado.

Neste trabalho é proposta uma metodologia para execução de experimentos em um processo de Mineração de Textos, com o objetivo de selecionar os atributos que melhor representam os textos considerando a precisão dos classificadores induzidos por algoritmos de aprendizado. Para tanto, é realizado o pré-processamento dos documentos transformando-os em uma tabela atributo-valor e aplicando algumas técnicas para reduzir a dimensionalidade dessa tabela considerando, entre outras coisas, a precisão do classificador induzido pelo algoritmo de aprendizado. O pré-processamento dos documentos é realizado usando uma ferramenta computacional denominada PreTexT. A metodologia é ilustrada usando um corpus de documentos e três algoritmos de aprendizado, C4.5rules [12], CN2 [4] e SVMTorch II [5].

Este trabalho está organizado da seguinte forma: na Seção 2 são apresentados resumidamente as fases do processo de Mineração de Textos, apresentando uma visão geral do pré-processamento dos documentos baseada na técnica *bag-of-words* usada neste trabalho; na Seção 3 são descritas as características principais da ferramenta computacional PreTexT desenvolvida para realizar o pré-processamento de textos; na Seção 4 é descrita a metodologia proposta, ilustrada na Seção 5 na qual são mostrados os experimentos realizados usando a ferramenta PreTexT para selecionar os atributos do corpus de documentos utilizados, bem como os resultados obtidos utilizando diferentes algoritmos de aprendizado; finalmente, na Seção 6 são apresentadas as conclusões.

2 Mineração de Textos

Em Mineração de Textos — MT — algumas fases são essenciais e comuns a qualquer processo, as quais podem ser definidas como: (1) coleta de documentos (textos), (2) pré-processamento de textos; (3) extração do conhecimento e (4) avaliação e interpretação dos resultados.

A primeira fase do processo de MT, coleta de documentos, consiste na recuperação de documentos que são relevantes para o domínio da aplicação do conhecimento a ser extraído. Os documentos coletados devem ser transformados para um formato aceito pelos algoritmos de extração de conhecimento. Essa fase, denominada de pré-processamento de textos, cria uma estrutura que é freqüentemente representada como uma tabela

atributo-valor para a coleção de documentos. Essa fase, apresentada em maiores detalhes nas próximas seções, é computacionalmente cara e um cuidadoso pré-processamento é fundamental para o sucesso de todo o processo de MT.

Os documentos representados em um formato adequado podem ser submetidos a algoritmos de extração de conhecimento com o objetivo de descobrir padrões úteis e desconhecidos nos documentos. E, finalmente, a fase de avaliação verifica se o objetivo foi alcançado ou se algumas das etapas devem ser refeitas.

A seguir, é apresentado resumidamente duas das principais questões relacionadas à fase de pré-processamento: como representar os documentos e como diminuir a dimensionalidade do espaço de atributos.

2.1 Representação de Documentos

Dada uma coleção de documentos $D = \{d_1, d_2, \ldots d_n\}$ e um conjunto de categorias $C = \{c_1, c_2, \ldots c_z\}$ associadas com a coleção de documentos D, a tarefa de categorização de textos consiste em induzir um classificador que possa determinar se o documento d_i pertence ou não a categoria c_k, para $i = 1, 2, \ldots n$ e $k = 1, 2, \ldots z$. Os documentos podem ser descritos como vetores na forma $(\mathbf{d}_1, c_1), ..., (\mathbf{d}_n, c_z)$, no qual $\mathbf{d}_i$ é um vetor de alta dimensão representando os termos (palavras) que ocorrem no documento e c_k é a classe associada ao documento.

A identificação dos termos em um documento pode se referir às palavras presentes no texto (*bag of words*), ou em representações mais sofisticadas como frases ou sentenças. Entretanto, resultados experimentais mostraram que representações mais sofisticadas perdem em desempenho com relação a palavras simples [1, 6, 8]. De acordo com [8], a razão mais provável para explicar esses resultados é que, embora termos mais sofisticados tenham qualidade semântica superior, a qualidade estatística é inferior em relação a termos baseados em palavras simples. Assim, pesquisas utilizando representações simples e sofisticadas de documentos continuam ativas.

Cada termo t_j, para $j = 1, 2, \ldots, m$, será um elemento do conjunto de atributos da tabela atributo-valor. A atribuição de valores a cada um dos termos é baseada na freqüência que o termo aparece nos documentos. Dependendo do domínio, a representação binária pode ser adequada para atribuir valores aos atributos. Nesse caso, o valor 1 significa presença do termo j no documento i e o valor 0 ausência do termo. No entanto, a representação binária é muito simples e, geralmente, medidas estatísticas são empregadas levando em consideração a freqüência que um termo aparece no documento, bem como a freqüência que esse termo é encontrado em todos os documentos da coleção de documentos. Por exemplo, *term frequency* (tf) é uma medida que utiliza o número de ocorrências do termo t_j no documento d_i. Porém, quando termos com alta freqüência aparecem em toda (ou na maioria) dos documentos da coleção, esses termos não fornecem informação útil para diferenciar documentos. A medida *inverse document frequency* (idf) favorece termos que aparecem em poucos documentos da coleção. A medida idf, definida como $\log \frac{n}{x}$, varia inversamente ao número de documentos x que contém o termo t_j em uma coleção de documentos. Assim, pode-se definir a medida $tfidf$ combinando as medidas tf e idf. Essas medidas são apresentadas na Tabela 1.

Tabela 1: Definição das medidas

Medida	Fórmula	Comentário
tf	$\#(t_j, d_i)$	$\#(t_j, d_i)$ é o número de vezes que o termo t_j ocorre no documento d_i.
$tfidf$	$\#(t_j, d_i).\log \frac{n}{x}$	as medidas tf e idf são combinadas, na qual x representa o número de documentos em D em que o termo t_j ocorre pelo menos uma vez.
$tfidfn$	$\frac{tfidf(t_j,d_i)}{\sqrt{\sum_{s=1}^{m}(tfidf(t_s,d_i))^2}}$	um fator de normalização é utilizado na equação $tfidf$ para que documentos de tamanhos diversos sejam tratados com a mesma importância.

Uma outra questão a ser considerada quando se utiliza as medidas tf e idf está relacionada a documentos que possuem um número muito diferente de termos. Em muitas situações, documentos pequenos são representados por poucos termos, enquanto que documentos maiores, geralmente, são representados por muitos termos. Quando uma grande quantidade de termos é usada na representação de documentos, a probabilidade do termo pertencer a um documento é alta e, assim, documentos maiores tem melhores chances de serem relevantes do que documentos menores. Normalmente, todos os documentos relevantes deveriam ser tratados com a mesma importância independente do seu tamanho. Um fator de normalização, nesse caso, deve ser incorporado para igualar o tamanho de vetores dos documentos. A medida $tfidfn$ — Tabela 1 — utiliza o fator de normalização para documentos de tamanhos diversos.

2.2 Redução da Dimensionalidade dos Atributos

Na criação da tabela atributo-valor utilizadas por algoritmos de Aprendizado de Máquina, cada termo que aparece no documento pode ser um elemento do conjunto de atributos que descreve o documento. Assim, a dimensionalidade do conjunto de atributos é um problema que deve ser tratado. Vários métodos podem ser utilizados a fim de reduzir a quantidade de atributos visando uma melhor representatividade e melhor desempenho do processo de MT. Entre outros, a transformação de cada termo para o radical que o originou, por meio de algoritmos de *stemming*, é um método amplamente utilizado e difundido.

Algoritmos de *stemming*, basicamente, consistem em uma normalização lingüística, na qual as formas variantes de um termo são reduzidas a uma forma comum denominada *stem*. A conseqüência da aplicação de algoritmos de *stemming* consiste na remoção de prefixos ou sufixos de um termo, ou mesmo na transformação de um verbo para sua forma no infinitivo. Portanto, um algoritmo de *stemming* é fortemente dependente do idioma no qual os documentos estão escritos. Um dos algoritmos de *stemming* mais conhecidos é o algoritmo do Porter que remove sufixos de termos em inglês [10]. O algoritmo tem sido amplamente usado, referenciado e adaptado nos últimos 20 anos. Diversas implementações do algoritmo estão disponibilizadas na *Web*, entre elas a página oficial escrita e mantida pelo autor para distribuição do seu algoritmo (`http://www.tartarus.org/~martin/PorterStemmer`).

A aplicação de algoritmos de *stemming* aos termos dos documentos reduz significativamente a quantidade de possíveis atributos que possam representar os documentos. Porém, na maioria das vezes, essa redução

não é suficiente e outras formas para reduzir a dimensionalidade é necessária. A Lei de Zipf descreve uma maneira de descobrir termos considerados pouco representativos em uma determinada coleção de documentos. A lei, formulada por George Kingsley Zipf professor de lingüística de Harvard (1902-1950), declara que a freqüência de ocorrência de algum evento está relacionada a uma função de ordenação. Zipf mostrou que uma das características das linguagens humanas, populações das cidades e muitos outros fenômenos humanos e naturais, seguem uma distribuição similar, a qual denominou de "*Principle of Least Effort*" [17].

Existem diversas maneiras de enunciar a Lei de Zipf para uma coleção de documentos. A mais simples é procedimental: pegar todos os termos na coleção e contar o número de vezes que cada termo aparece. Se o histograma resultante for ordenado de forma decrescente, ou seja, o termo que ocorre mais freqüentemente aparece primeiro, então, a forma da curva é a "curva de Zipf", para aquela coleção de documentos. Se a curva de Zipf for plotada em uma escala logarítmica, ela aparece como uma reta com inclinação -1, embora [**?**] mostram que a curva de Zipf é dependente da linguagem e do estilo. A Lei de Zipf em documentos de linguagem natural pode ser aplicada não apenas aos termos mas, também, a frases e sentenças da linguagem. Na realidade, a lei de Zipf é uma observação empírica que se aplica em diversos domínios, e segue a distribuição $p_1 = \frac{c}{1}, p_2 = \frac{c}{2}, \ldots, p_n = \frac{c}{n}$, na qual $c = \frac{1}{H_n}$ e $H_n = \sum_{i=1}^{n} \frac{1}{i}$ [7]. Ou seja, considerando uma coleção de documentos escritos em linguagem natural, foi observado que o j-ésimo termo mais comum ocorre com freqüência inversamente proporcional a j.

Enquanto Zipf verificou sua lei utilizando jornais escritos em inglês, Luhn [9] usou a lei como uma hipótese nula para especificar dois pontos de corte, os quais denominou de superior e inferior, para excluir termos não relevantes. Os termos que excedem o corte superior são os mais freqüentes e são considerados comuns por aparecer em qualquer tipo de documento, como as preposições, conjunções e artigos. Já os termos abaixo do corte inferior são considerados raros e, portanto, não contribuem significativamente na discriminação dos documentos. Assim, Luhn propôs uma técnica para encontrar termos relevantes, assumindo que os termos mais significativos para discriminar o conteúdo do documento estão em um pico imaginário posicionado no meio dos dois pontos de corte. Porém, uma certa arbitrariedade está envolvida na determinação dos pontos de corte, bem como na curva imaginária, os quais são estabelecidos por tentativa e erro [14]. Como a Lei de Zipf, a técnica não é restrita apenas a termos mas, também, pode ser aplicada a *stem* ou sentenças dos documentos. A seguir, é descrita uma ferramenta computacional, por nós implementada, que utiliza os conceitos apresentados.

3 A Ferramenta PreTexT

PreTexT é uma ferramenta computacional implementada na linguagem Perl [16] usando o paradigma de orientação a objetos. A ferramenta foi desenvolvida com o objetivo de realizar de forma automática o pré-processamento de uma coleção de documentos escritos em três idiomas distintos: português, espanhol e inglês. A implementação da ferramenta é baseada no algoritmo de *stemming* do Porter para a língua inglesa,

o qual foi adaptado para a língua portuguesa e espanhola. A ferramenta também inclui facilidades para reduzir a dimensionalidade do conjunto de atributos usando a Lei de Zipf e os cortes Luhn.

Resumidamente, dentre as características gerais da ferramenta PreTexT, podem ser destacadas algumas, tais como: (i) extrair *stems* de palavras em português, espanhol e inglês; (ii) ignorar palavras que não são consideradas significativas usando uma lista de *stopwords*; (iii) criar arquivos intermediários que contém as freqüências dos *stems* de cada um dos documentos, a freqüência dos *stems* na coleção de documentos e a freqüência das palavras que originam cada um desses *stems*; (iv) utilizar qualquer das quatro medidas definidas na Seção 2.1 para atribuir o valor associado a cada *stem* na coleção de documentos; (v) aplicar a Lei de Zipf e cortes de Luhn; (vi) trabalhar com termos simples ou compostos — 1, 2 e 3-*grams*; (vii) gerar gráficos; (viii) criar a tabela atributo-valor utilizando *stems*.

A lista de *stopwords* padrão de PreTexT contém termos gerais tais como artigos, conjunções, preposições, pronomes e alguns advérbios. Essa lista encontra-se armazenada em um arquivo. O usuário pode utilizar somente essa lista de *stopwords* padrão da ferramenta bem como pode criar outros arquivos contendo listas adicionais de *stopwords* específicas do domínio. A ferramenta está preparada para considerar conjuntos de arquivos contendo *stopwords*. Para realizar automaticamente os cortes de Luhn, PreTexT tem uma opção para utilizar somente os *stems* que estão no intervalo de freqüência $(\overline{x} - ks; \overline{x} + ks)$ no qual $\overline{x}$ é a média da freqüência dos *stems*, s é o desvio-padrão e k é uma constante definida pelo usuário. Uma outra opção permite ao usuário definir livremente os pontos de corte superior e inferior.

A ferramenta, ilustrada na Figura 1, consiste de dois módulos principais: **Stem.pl** e **Report.pl**. O primeiro módulo é responsável pela transformação de termos nos *stems* correspondentes. A entrada para esse módulo pode ser uma palavra, um documento ou uma coleção de documentos. Na Figura 1 está ilustrado este último caso, no qual a coleção de documentos é identificada pelo nome de um diretório, embaixo do qual encontra-se um conjunto de arquivos tal que cada arquivo contém um dos documentos da coleção. Além disso, o usuário deve especificar o idioma dos documentos, *i.e.* português, inglês ou espanhol e, se for o caso, a lista de *stopwords* por ele definida, a qual será adicionada à lista de *stopwords* padrão da ferramenta. A saída consiste de vários arquivos intermediários descritos resumidamente no item (iii) da descrição das características gerais da ferramenta. Esses arquivos contém informações úteis para o usuário e também são utilizados pelo módulo **Report.pl**.

O módulo **Report.pl** tem como entrada os arquivos intermediários, gerados pelo **Stem.pl**, e um arquivo no qual são especificados os parâmetros de execução. Nesse arquivo é definida qual medida utilizar, os pontos de corte mínimo e máximo de Luhn, bem como a quantidade de *grams* a considerar (termos simples ou compostos). Os valores *default* da ferramenta são: a medida tf, sem corte (todos os *stems* são considerados) e 1, 2 e 3-*grams* (*stems* simples e compostos). A saída do módulo **Report.pl** consiste dos arquivos de dados `.data` e `.names` no formato utilizado pelo DISCOVER[1], além de diversos gráficos que mostram a freqüência

[1] A ferramenta PreTexT será integrada futuramente ao ambiente DISCOVER, um projeto de pesquisa em desenvolvimento no Laboratório de Inteligência Computacional, LABIC - http://labic.icmc.usp.br, para planejamento e execução de experimentos relacionados com o uso de sistemas de aprendizado no processo de Mineração de Dados e de Mineração de Textos [3, 11].

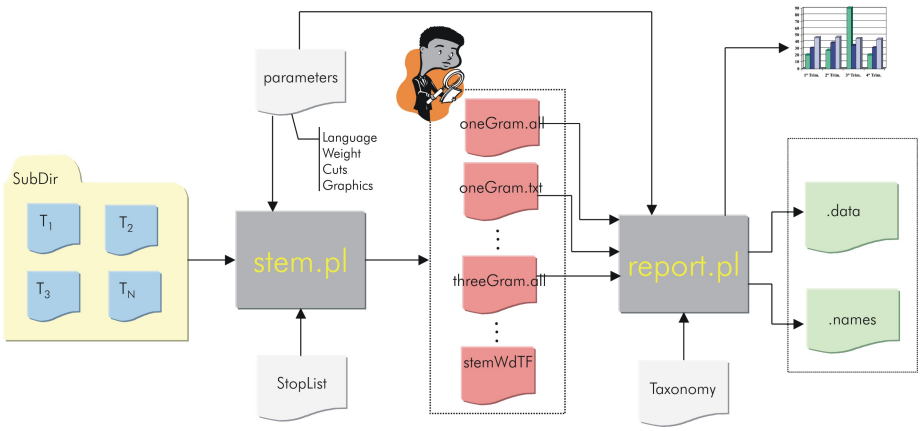

Figura 1: A ferramenta PreTexT

dos *stems* na coleção de documentos.

4 Metodologia Proposta para Execução de Experimentos

A metodologia proposta é um processo iterativo e interativo, envolvendo desde a fase de pré-processamento dos textos até a escolha do melhor classificador induzido por algoritmos de aprendizado.

Basicamente, a metodologia consiste em: (i) submeter os documentos ao módulo **Stem.pl** da ferramenta PreTexT para encontrar os *stems* dos termos e as freqüências respectivas; (ii) gerar uma ou mais tabelas atributo-valor com o módulo **Report.pl** do PreTexT, usando diversas medidas de atribuição de valores aos atributos, eliminando os *stems* com freqüência abaixo de um dado limiar; (iii) encontrar pontos de corte mínimo e máximo usando como referência a quantidade de exemplos da classe minoritária e o desvio padrão da média de freqüência; (iv) submeter as tabelas atributo-valor a algoritmos de aprendizado medindo os erros do classificador induzido utilizando, por exemplo, *10 fold cross-validation*; (v) analisar os erros encontrados pelo classificador.

É proposto um limiar de freqüência, item (ii), para que *stems* com freqüência abaixo desse limiar sejam descartados. Foi definido o valor de 10% da quantidade de exemplos da classe minoritária. Já no item (iii), a proposta é usar num primeiro momento o número de exemplos pertencentes a classe minoritária como referência ao corte mínimo. A idéia de usar esse valor considera a possibilidade de existir um atributo que discrimina perfeitamente uma classe. No caso extremo, esse atributo apareceria somente uma vez em todos os documentos dessa classe, e seu valor mínimo é dado pelo número de documentos na classe minoritária. A partir dos resultados dos experimentos, pode-se definir novos valores para o corte mínimo. Para encontrar o ponto de corte máximo, a nossa proposta é utilizar como referência um ou dois desvios padrão da média de freqüência dos *stems*.

Para ilustrar a metodologia proposta, foram realizados vários experimentos usando uma coleção de documentos escritos em inglês. Essa coleção de documentos, fornecida pelo grupo de pesquisadores do ISISTAN[2],

[2] http://www.exa.unicen.edu.ar

contém 332 documentos classificados em quatro classes: Biomedical, Goats, Music e Sheeps. Cada documento está armazenado em um arquivo texto, extensão `txt`. O tamanho total dessa coleção de documentos é 641,1 KB, cujo tamanho médio dos documentos é 217,47 KB ± 298,33 KB. Na Tabela 2 são mostrados as distribuições desses documentos em cada uma das quatro classes.

Tabela 2: Número de documentos em cada classe

Biomedical	Goats	Music	Sheeps	Total
136 (40,96%)	70 (21,08%)	61 (18,38%)	65 (19,58%)	332 (100%)

Os algoritmos de aprendizado utilizados foram o C4.5rules, CN2 e o SVMTorch II. C4.5rules e CN2 são algoritmos de aprendizado simbólico que induzem regras de decisão as quais descrevem um contexto específico associado com uma classe. Apesar dos dois algoritmos induzirem regras de decisão, o *bias* indutivo de cada um dos dois algoritmos é muito diferente [2].

SVMs são técnicas de aprendizado baseadas na Teoria de Aprendizado Estatístico proposta por [15]. Essa técnica mapeia os dados de entrada para um espaço abstrato de alta dimensão, onde os exemplos podem ser eficientemente separados por um hiperplano. O SVM incorpora este conceito usando funções denominadas *Kernels*. Essas funções permitem o acesso a espaços complexos de maneira simplificada e computacionalmente eficientes. O hiperplano ótimo nesse espaço é definido como aquele que maximiza a margem de separação entre dados pertencentes a diferentes classes. A principal vantagem dos SVMs é sua precisão e robustez em dados com alta dimensionalidade. Entretanto, diferentemente de algoritmos de aprendizado simbólico, classificadores induzidos utilizando SVMs não são diretamente interpretáveis pelos usuários.

A seguir, a metodologia proposta é ilustrada utilizando a coleção de documentos e os três algoritmos descritos nesta seção.

5 Resultados Experimentais

A coleção de documentos foi fornecida à ferramenta PreTexT, no primeiro passo, especificando como corte mínimo o valor 6, o qual representa 10% da classe minoritária Music com 61 exemplos. Foram encontrados um total de 1284 *stems* com média de freqüência 77,2 ± 85,3. Utilizando tanto os gráficos quanto as tabelas de freqüência de *stems* gerados por PreTexT, foi possível observar a distribuição das freqüências dos *stems* na coleção de documentos, Figura 2.

O próximo passo consiste em determinar alguns valores para aplicar os cortes mínimo e máximo, executar PreTexT para gerar a tabela atributo-valor correspondente, *i.e.* os arquivos `.data` e `.names` no formato do DISCOVER, e observar o erro cometido pelo classificador induzido utilizando essa tabela atributo-valor, de forma a ajustar convenientemente os valores desses cortes.

Na Tabela 3 são mostrados os resultados obtidos em cinco experimentos realizados. Em cada um desses experimentos foram utilizadas duas das quatro medidas implementadas na ferramenta: as medidas tf e $tfidfn$

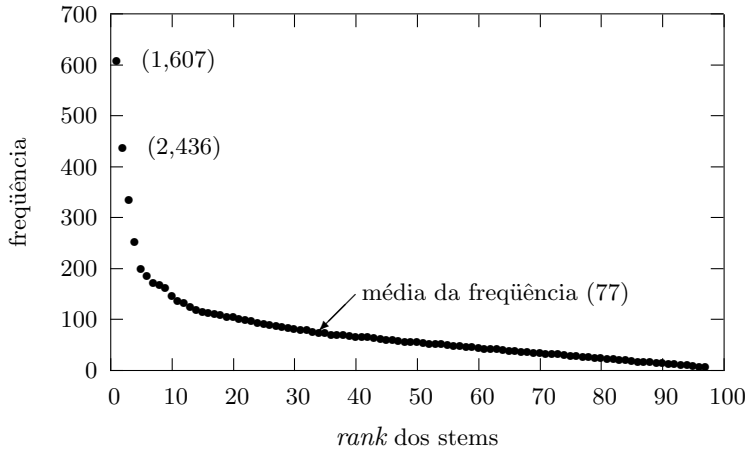

Figura 2: Freqüência de *stems*

(Tabela 1) utilizando os algoritmos C4.5rules (identificado por C45r) e CN2. Nessa tabela, Exp identifica o experimento; Min e Max são, respectivamente, os valores mínimo e máximo utilizados para realizar os cortes de Luhn; # Atr é a quantidade de atributos (*stems*) da tabela atributo-valor construída; Medida identifica a medida utilizada no experimento; A_Erro é o erro aparente dos classificadores induzidos por C4.5rules e CN2, *i.e.* quando toda a coleção de documentos é utilizada para treinar e testar; Erro_10foldCV é o erro e o desvio padrão do classificador induzido calculado usando 10 $fold\ cross\ validation$; # Regras é o número de regras do conjunto de regras que constitui o classificador final e # Atr_Regras é o número de atributos distintos presentes nesse conjunto de regras.

Tabela 3: Resultados experimentais - C4.5rules e CN2

Exp	Min	Max	# Atr	Medida	A_erro		Erro_10foldCV		# Regras		# Atr_R	
					C45r	CN2	C45r	CN2	C45r	CN2	C45r	CN2
				tf	$7,2\%$	$2,7\%$	$6,7\% \pm 1,7\%$	$7,6\% \pm 1,5\%$	5	33	4	54
E_1	6		1284	$tfidfn$	$8,1\%$	$2,7\%$	$13,0\% \pm 1,5\%$	$13,0\% \pm 1,7\%$	8	41	8	69
				tf	$20,5\%$	$3,6\%$	$34,9\% \pm 1,8\%$	$29,8\% \pm 1,9\%$	15	71	18	118
E_2	6	163	1276	$tfidfn$	$20,5\%$	$3,6\%$	$34,1\% \pm 2,1\%$	$27,1\% \pm 1,5\%$	15	71	18	118
				tf	$7,2\%$	$7,5\%$	$12,1\% \pm 1,6\%$	$9,0\% \pm 1,5\%$	9	22	10	29
E_3	61		60	$tfidfn$	$7,8\%$	$10,1\%$	$12,1\% \pm 1,2\%$	$14,1\% \pm 1,7\%$	11	22	17	29
				tf	$8,1\%$	$5,4\%$	$9,3\% \pm 1,6\%$	$9,6\% \pm 1,1\%$	7	24	9	42
E_4	30		172	$tfidfn$	$8,1\%$	$7,5\%$	$16,3\% \pm 1,5\%$	$13,2\% \pm 1,5\%$	10	28	12	48
				tf	$6,3\%$	$4,2\%$	$8,1\% \pm 0,9\%$	$8,2\% \pm 1,5\%$	9	28	10	45
E_5	15		446	$tfidfn$	$8,7\%$	$6,0\%$	$13,9\% \pm 1,2\%$	$11,7\% \pm 1,5\%$	9	32	12	53

No primeiro experimento foram executados os algoritmos com todos os atributos, excluindo apenas os atributos com freqüência menor que 6. Pode ser observado que o erro encontrado por C4.5rules usando a medida tf foi muito bom considerando que o erro da classe majoritária é 59,04%.

O segundo experimento ilustra a busca por um valor de corte Max apropriado. O valor escolhido foi 163. Esse valor está relacionado com a média 77,2 da freqüência dos *stems* mais um desvio padrão. Esse experimento não teve um bom desempenho pois, apesar de ter retirado apenas 8 atributos, pode ser observado que foram retirados atributos relevantes para discriminar as classes, pois o erro incrementou muito. Por exemplo, para a medida tf o erro incrementou aproximadamente 5 vezes (de 6,7% para 34,0%) e quatro vezes (7,6% para 29,8%) para C4.5rules e CN2 respectivamente.

No experimento E_3 foi utilizado como ponto de corte Min o número de exemplos pertencentes a classe

minoritária. Pode ser observado que esse valor não é apropriado, visto que o erro aumenta, com relação ao experimento E_1, para ambos algoritmos. Assim, para os experimentos E_4 e E_5 definiu-se Min com o valor 30 e 15 respectivamente representando, aproximadamente, 50 e 25% do valor anterior. Percebe-se que apesar do erro ter diminuído com relação ao experimento E_3, o erro obtido no experimento E_1 para a medida tf é menor. Levando em consideração somente o erro de classificação, pode-se concluir que os resultados do experimento E_1 são os melhores. Com relação aos modelos induzidos, é possível observar que o mais simples é o induzido por C4.5rules, já que esse classificador consegue resultado semelhante com um conjunto de 5 regras e 4 atributos diferentes, enquanto que CN2 necessita de um conjunto de 33 regras que utilizam 54 atributos diferentes. Na Figura 3 são mostrados os erros dos classificadores induzidos nos experimentos E_1, E_3, E_4 e E_5.

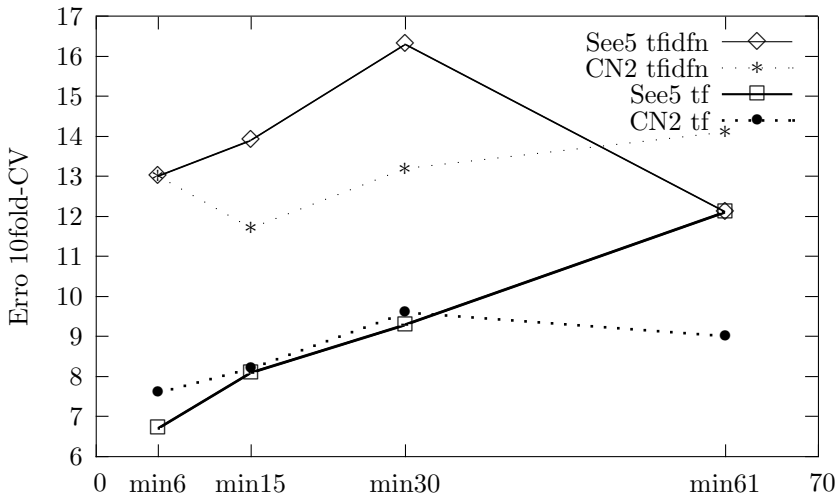

Figura 3: Erro dos classificadores

Os dados utilizados nos experimentos E_1 e E_5, com os quais foram obtidos os menores erros de classificação, foram submetidos ao algoritmo SVMTorch. Os resultados obtidos são mostrados na Tabela 4, na qual 10fold_errorL representa os erros obtidos usando 10 *fold cross validation* e o *Kernel* Linear; 10fold_errorG representa os erros obtidos usando 10 *fold cross validation* e o *Kernel* Gaussiano com desvio padrão 10.

Tabela 4: Resultados experimentais - SVMTorch

Exp	Min	Max	# Atr	Medida	10fold_errorL	10fold_errorG
E_1	6		1284	tf	$13,5\% \pm 6,5\%$	$17,8\% \pm 5,8\%$
				$tfidfn$	$26,8\% \pm 7,0\%$	$16,3\% \pm 4,1\%$
E_5	15		446	tf	$12,9\% \pm 4,9\%$	$15,0\% \pm 4,4\%$
				$tfidfn$	$23,0\% \pm 9,4\%$	$15,4\% \pm 6,5\%$

Os erros obtidos com SVMTorch foram maiores do que os obtidos com C4.5rules e CN2. Isso mostra mais uma vez a importância de realizar experiências para decidir qual algoritmo de aprendizado é melhor para um determinado conjunto de dados. Como mencionado anteriormente, a técnica de SVMs é considerada muito boa para dados com alta dimensionalidade. Ainda que na maioria dos casos a precisão de SVM é superior à de algoritmos simbólicos, esse resultado não se verifica sempre, como pode ser observado nos experimentos realizados neste trabalho. É interessante observar que na maioria dos experimentos realizados,

a medida tf obteve melhores resultados que a medida $tfidfn$. Este é um resultado que também consideramos particular para este conjunto de dados e indutores utilizados, já que a medida $tfidfn$ não está diretamente correlacionada com a medida tf.

É importante salientar que o uso de um indutor simbólico permite ao usuário/especialista verificar na primeira execução da ferramenta, ou posteriormente, os atributos (*stems*) utilizados pelas regras induzidas. Após, utilizando as informações fornecidas nos arquivos gerados pelo módulo **Stem.pl** da ferramenta PreTexT, é possível verificar se esses *stems* correspondem a palavras consideradas relevantes. Caso contrário, essas palavras podem ser colocadas nas listas de *stopwords* do usuário executando novamente a ferramenta.

6 Conclusão

Diversas formas de representação de documentos podem ser adotadas em um processo de MT. A representação de documentos usando a abordagem *bag-of-words* é uma das mais utilizadas devido a sua simplicidade e seu bom desempenho comparado a representações mais sofisticadas, tais como frases, clusters de palavras ou grafos conceituais. No entanto, a abordagem *bag-of-words* é um processo estatístico, o qual utiliza a freqüência que as palavras ocorrem no documento, não envolvendo semântica relacionada as palavras.

Entretanto, quando a abordagem *bag-of-words* é utilizada na representação dos documentos, uma questão primordial consiste em determinar quais e como serão os atributos que discriminem bem os documentos, visto que a quantidade de possíveis atributos é muito grande. Uma outra questão está relacionada com a escolha do algoritmo de aprendizado a ser utilizado para extrair um bom classificador. Assim, essas questões mostram que a tarefa de MT não é uma tarefa trivial.

Neste trabalho foi apresentada uma ferramenta computacional cujo objetivo consiste em auxiliar o usuário no pré-processamento de dados textuais, usando algoritmos de aprendizado. PreTexT possui muitos recursos e facilidades que auxiliam no pré-processamento de documentos utilizando a técnica de *stemming*. Também, foi apresentada uma metodologia para execução de experimentos em MT usando diversos algoritmos de aprendizado e uma mesma coleção de documentos, permitindo assim escolher o melhor classificador para essa coleção analisando os resultados obtidos. A fim de ilustrar o uso e da ferramenta PreTexT e a metodologia de execução de experimentos foram mostradas algumas experiências realizadas com uma coleção de documentos e três algoritmos de aprendizado.

Agradecimentos

O auxílio financeiro concedido pela CAPES e FAPESP, Brasil.

Referências

[1] C. Apté, F. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. *Information Systems*, 12(3):233–251, 1994. http://citeseer.nj.nec.com/apte94automated.html.

[2] J. A. Baranauskas and M. C. Monard. An unified overview of six supervised symbolic machine learning inducers. Technical Report 103, ICMC-USP, 2000. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_103.ps.zip.

[3] G. E. A. P. A. Batista. Pré-processamento de dados em aprendizado de máquina supervisionado, 2003. Tese de Doutorado, ICMC-USP.

[4] P. Clark and R. Boswell. The cn2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[5] R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[6] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM 98)*, 1998. http://research.microsoft.com/~sdumais/cikm98.pdf.

[7] D. E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, 1973.

[8] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50, June 1992.

[9] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.

[10] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[11] R. C. Prati. O *framework* de integração do sistema DISCOVER, abril 2003. Dissertação de Mestrado, ICMC-USP.

[12] J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, CA, 1988.

[13] F. Sebastiani. Machine learning in automated text categorisation. *ACM Computing Surveys*, 34(1):1–47, March 2002. http://faure.iei.pi.cnr.it/~fabrizio/Publications/ACMCS02.pdf.

[14] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979. http://citeseer.nj.nec.com/vanrijsbergen79information.html.

[15] V. N. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. theory of probability and its applications. (16):262–280, 1971.

[16] L. Wall, T. Christiansen, and R. L. Schwartz. *Programming in PERL*. O'Reilly,Inc, 1996.

[17] G. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, 1949.