Quanto mais simples, melhor! Categorização de Textos baseada na Navalha de Occam

Renato M. Silva, Akebo Yamakami
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP)
Campinas, São Paulo, Brasil
{renatoms, akebo}@dt.fee.unicamp.br

Tiago A. Almeida
Departamento de Computação (DComp)
Universidade Federal de São Carlos (UFSCar)
Sorocaba, São Paulo, Brasil
talmeida@ufscar.br

***Resumo*—Categorização de textos é um problema que tem recebido muita atenção nos últimos anos devido ao aumento no volume de informações textuais. O processo manual de categorizar documentos de texto é cansativo, tedioso, demorado e muitas vezes impraticável quando há um alto volume de dados. Existe uma grande demanda para que esse processo seja realizado de maneira automática por métodos computacionais. Embora vários métodos já tenham sido propostos, muitos sofrem com o problema da maldição da dimensionalidade ou apresentam alto custo computacional, inviabilizando seu uso em cenários reais. Diante disso, este trabalho apresenta dois métodos de categorização de textos baseados no princípio da descrição mais simples. Experimentos realizados com bases de dados reais, públicas e de grande porte indicam que os métodos propostos possuem baixo custo computacional e alta eficácia, apresentando resultados muitas vezes superiores aos métodos *naïve* Bayes e máquinas de vetores de suporte, considerados atualmente o estado-da-arte em categorização de textos. Além disso, os métodos propostos são robustos na prevenção de *overfitting*, fáceis de implementar e naturalmente adaptados ao aprendizado incremental, favorecendo seu emprego em cenários reais e *online*.**

I. Introdução

O volume de informações digitais armazenadas em forma textual vem crescendo desenfreadamente. Atualmente, livros, jornais de notícias, revistas, artigos científicos, entre outros, são geralmente publicados em meio digital, fato que não ocorria há alguns anos. Até mesmo documentos antigos estão sendo digitalizados com o intuito de facilitar o acesso e controle. Somado a isso, existem outros inúmero serviços, tais como *e-mail*, mensagens de texto compartilhadas através de celulares e redes sociais e comentários em sites e blogs, que contribuem de forma significativa na quantidade de informação textual que precisa ser analisada com o intuito de prover acesso, segurança, organização e demais tipos de ações que colaborem positivamente para a experiência do usuário.

Quando se tem uma grande quantidade de documentos disponíveis, é necessário agrupá-los em categorias para facilitar a análise e extração de conhecimento, bem como a busca e o manuseio. Os humanos tem uma grande facilidade para realizar esse trabalho. Ao ler uma notícia, um livro ou um artigo científico, uma pessoa normalmente consegue saber que tipo de assunto é abordado no documento e colocá-lo em um grupo adequado. Porém, quando o número de documentos é muito grande, a categorização dos mesmos por meio de trabalho humano se torna inviável, pois requer muito tempo e uma quantidade muito grande de mão-de-obra.

Uma das formas de automatizar o processo de categorização de documentos é por meio de algoritmos de classificação de texto. Esse tipo de algoritmo recebe um documento e o classifica em alguma categoria predefinida. Vários métodos de aprendizado de máquina foram aplicados em tarefas de categorização de textos, tais como k-vizinhos mais próximos [1], [2], *naïve* Bayes [3], Rocchio [4], [2], máquinas de vetores de suporte (do inglês, *Support Vector Machines* – SVM) [5], árvores de decisão [3], [6] e redes neurais artificiais [7], [8]. Contudo, a maioria desses métodos sofrem de pelo menos um dos seguintes problemas bem conhecidos [9], [10], [11]:

1) o problema da maldição da dimensionalidade;
2) o alto custo computacional necessário para o treinamento, o que impede a aplicação desses métodos em problemas de aprendizado *online*.

Vários trabalhos de pesquisa sobre categorização de textos indicam claramente que os melhores métodos de classificação atualmente disponíveis são o SVM e o *naïve* Bayes [5], [12], [13], [14]. Dado que a dimensionalidade do espaço de atributos é geralmente muito alta, em geral, o SVM com *kernel* linear obtém os melhores resultados, se comparado às outras funções de *kernel* disponíveis [15], [16], [13].

Idealmente, um bom classificador de texto deveria oferecer um aprendizado rápido, tal como o *naïve* Bayes, e robustez contra o super-ajustamento dos dados (*overfitting*), tal como o SVM. Porém, enquanto que, em algumas aplicações, o desempenho do *naïve* Bayes decai quando a dimensionalidade aumenta, o SVM frequentemente apresenta custo computacional de treinamento muito alto, principalmente se o número de classes do problema é grande e se o espaço de atributos tem alta dimensão.

Diante disso, neste trabalho são propostos dois métodos de categorização de textos baseados no princípio da descrição mais simples (do inglês, *Minimum Description Length* – MDL) [17], [18]. Os ideais teóricos desse princípio tem suas raízes na complexidade de Kolmogorov e no princípio da navalha de Occam. Desse modo, em problemas de seleção de modelos, os menos complexos e que se ajustem bem aos

dados são preferíveis. Consequentemente, métodos baseados no princípio MDL naturalmente evitam *overfitting*, embora tenham um baixo custo computacional. Portanto, métodos de classificação baseados no princípio MDL tem um processo de treinamento similar ao do *naïve* Bayes, mas oferecem robustez na prevenção do problema de *overfitting*. Outra vantagem dos métodos baseados no princípio MDL é que eles são naturalmente adaptados para serem aplicados em problemas multiclasses, sem a necessidade de qualquer truque, tais como a técnica um-contra-todos e outras variantes [19]. Além disso, eles podem ser usados em problemas de classificação *online*, dado que seu aprendizado pode ser atualizado de forma incremental.

Especificamente, neste trabalho são apresentados os métodos de categorização de textos MDL-CF e MDL-DFS. O primeiro é formado pela combinação do MDL com a técnica de fatores de confidência (do inglês, *Confidence Factors* – CF) [10], [20], e o segundo é formado pela combinação do MDL com a técnica de seleção de atributos distintivos (do inglês, *Distinguishing Feature Selector* – DFS) [21]. Ambos foram avaliados na categorização de textos usando bases de dados tradicionais, grandes e públicas e os resultados obtidos foram comparados com os desempenhos do SVM linear e *naïve* Bayes.

O restante deste trabalho está estruturado da seguinte forma: na Seção II são apresentados os conceitos básicos sobre o princípio da descrição mais simples. Na Seção III são descritos os dois métodos de categorização de textos propostos neste trabalho. As configurações adotadas nos experimentos são detalhadas na Seção IV. Na Seção V são apresentados os resultados. Por fim, conclusões e direções para trabalhos futuros são descritas na Seção VI.

II. Princípio da descrição mais simples

O princípio da descrição mais simples (MDL) possui a prerrogativa de que em um problema de seleção de modelos que tenha dois ou mais candidatos, deve-se escolher aquele que possui o menor tamanho de descrição [17], [18]. Isso significa que modelos menos complexos são preferíveis [22], [23]. Essa ideia é uma formalização do princípio da parcimônia, também conhecido como navalha de Occam. A base desse princípio foi formulada inicialmente pelo frade franciscano Guilherme de Occam (do inglês, *William of Occa*m), na Idade Média, como uma crítica à filosofia escolástica que empregava teorias muito complexas para explicar a realidade. Atualmente, ele é usado em teoria de aprendizagem e afirma que entre duas ou mais hipóteses que possam explicar um mesmo fenômeno, é preferível que se escolha a hipótese mais simples [24].

Em computação, o MDL tem suas raízes na complexidade de Kolmogorov que é definida como o tamanho do menor programa que descreve um objeto representado por uma sequência binária [25], [26]. Quanto menor a complexidade de Kolmogorov de uma determinada sequência, mais regularidade ela apresenta. Portanto, o menor programa que consegue imprimir uma determinada sequência é também o que mais a compreendeu e tem mais conhecimento sobre ela, logo ele deve ser escolhido para representá-la.

Dado que o MDL incorpora o princípio da navalha de Occam, a complexidade de Kolmogorov e, ao mesmo tempo, procura selecionar o modelo que melhor representa os dados, ele gera um *trade-off* entre a complexidade do modelo e a qualidade do seu ajuste aos dados de observação. Tais características são bastante desejadas na área de aprendizado de máquina, pois naturalmente evitam *overfitting*, já que uma hipótese altamente complexa que se ajusta extremamente bem aos dados observados, normalmente comete muitos erros em dados não conhecidos.

Matematicamente, dado um conjunto de potenciais modelos $M_1, M_2, ..., M_n$, o MDL seleciona aquele, tal que:

$$M_{mdl} = \arg\min_{M}[L(M) + L(D|M)], \tag{1}$$

sendo que, $L(M)$ corresponde ao tamanho da descrição do modelo M, representando sua complexidade, e $L(D|M)$ ao tamanho da descrição do dado D quando codificado com M, representando quão bem M descreve D. Rissanen [17] mostrou que $L(D|M)$ pode ser calculado através do código de Shannon-Fano e, portanto, $L(D|M) = -\log P(D|M)$, onde $P(D|M)$ é a probabilidade condicional de D dado M.

O MDL conforme definido na Equação 1, foi proposto por Rissanen [17], [18] e é conhecido como *crude* MDL. De acordo com Grünwald [27], um problema do *crude* MDL é encontrar uma boa codificação para os modelos M, pois existe um risco de que $L(M)$ se torne arbitrário, uma vez que ele pode ser muito grande para um determinado código e muito pequeno para outro. Dessa forma, Rissanen [28] contornou esse problema, propondo um nova versão do MDL que usa códigos universais para medir o tamanho da descrição de cada modelo. Basicamente, nessa nova versão, é empregado apenas uma parte do código com tamanho $\bar{L}(D|M)$.

III. Categorização de textos usando o princípio da descrição mais simples

Cada documento é definido como uma sequência de *tokens* (um *token* pode ser um termo ou uma palavra). Cada *token* t_i de um documento $\vec{d}$ tem um tamanho de descrição $L(t_i|c_j)$, calculado de acordo com a sequência de *tokens* dos documentos de treinamento que pertencem à classe c_j. O tamanho da descrição de $\vec{d}$ em relação à classe c_j corresponde à soma do tamanho da descrição de todos os *tokens* de $\vec{d}$, definido por:

$$L(\vec{d}\,|c_j) = \sum_{i=1}^{|\vec{d}|} L(t_i|c_j),$$

onde $|\vec{d}|$ corresponde à quantidade de *tokens* no documento $\vec{d}$, $L(t_i|c_j) = \lceil -\log_2 P(t_i|c_j) \rceil$ e $P(t_i|c_j)$ é a probabilidade condicional do *token* t_i dados os documentos da classe c_j. Considerando que $n_{c_j}(t_i)$ corresponde à quantidade de documentos pertencentes à classe c_j em que o *token* t_i aparece, a probabilidade condicional de qualquer *token* ocorrer em

documentos de c_j pode ser calculada pela estimativa de máxima verossimilhança:

$$P(t_i|c_j) = \frac{n_{c_j}(t_i) + \frac{1}{|\Omega|}}{n_{c_j} + 1},$$

onde n_{c_j} é a soma de $n_{c_j}(t_i)$ para todos os *tokens* que aparecem em documentos que pertencem à classe c_j e $|\Omega|$ é o tamanho do vocabulário. Esse parâmetro é usado para reservar uma porção de probabilidade para *tokens* que ainda não foram apresentados ao classificador.

Por fim, $\vec{d}$ recebe o rótulo j, correspondente à classe c_j com o menor tamanho de descrição relacionado ao documento $\vec{d}$:

$$c(\vec{d}) = \arg\min_{\forall c} L(\vec{d}\,|c_j).$$

A. Fatores de confidência

Os fatores de confidência (do inglês, *Confidence Factors* – CF) [10] foram inicialmente propostos com o intuito de reduzir o impacto de ruídos introduzidos por *tokens* com baixa significância. A técnica CF atribui uma pontuação de relevância para o *token* t_i ($0 \leq CF(t_i) \leq 1$) através da seguinte equação:

$$CF(t_i) = \frac{1}{m-1} \sum_{\forall j|j\neq k} \frac{\left(\frac{(H_k - H_j)^2 + (H_k H_j) - \frac{K_1}{SH}}{(SH)^2}\right)^{K_2}}{1 + (\frac{K_3}{SH})},$$

sendo que, k corresponde ao índice da classe com maior probabilidade local; $j = 1, \ldots, m$ aos índices das m classes; H_k ao número de documentos que possuem o termo t_i e que pertencem a classe com maior probabilidade local; H_j ao número de documentos que possuem o termo t_i e que pertencem a classe c_j; $SH = H_k + H_j$ e K_1, K_2, K_3 são constantes que ajustam a velocidade de decaimento do fator de confidência. Os valores usados neste trabalho foram $K_1 = 0,25$, $K_2 = 10,0$ e $K_3 = 8,0$, conforme proposto originalmente por Assis *et al.* [10].

Almeida *et al.* [20] e Almeida e Yamakami [11] demonstraram que um classificador de texto binário baseado no MDL pode ter seu desempenho melhorado se o CF for inserido na sua equação principal. Dessa forma, estendendo a ideia original para tratar problemas genéricos e multiclasses, o tamanho da descrição de um documento ($\vec{d}$) em relação à classe c_j é calculado por:

$$L(\vec{d}\,|c_j) = \sum_{i=1}^{|\vec{d}|} L(t_i|c_j) \frac{1}{1 - CF(t_i)}.$$

B. Seletor de atributos distintivos

O seletor de atributos distintivos (do inglês, *Distinguishing Feature Selector* – DFS) foi proposto por Uysal e Gunal [21] com o intuito de determinar a relevância de um *token* (t_i) analisando sua contribuição para a discriminação das classes. Ele é calculado pela seguinte expressão:

$$DFS(t_i) = \sum_{\forall j} \frac{P(c_j|t_i)}{P(\bar{t}_i|c_j) + P(t_i|\bar{c}_j) + 1},$$

onde $P(c_j|t_i)$ corresponde à probabilidade condicional da classe c_j, dada a presença do *token* t_i; $P(\bar{t}_i|c_j)$ é a probabilidade condicional da ausência de t_i, dada a classe c_j; e $P(t_i|\bar{c}_j)$ corresponde à probabilidade condicional do *token* t_i, dadas as classes diferentes de c_j.

Como tal equação retorna valores entre 0,5 e 1, foi realizada uma normalização para o intervalo $[0, 1]$, de acordo com a seguinte equação:

$$DFS(t_i) = \frac{DFS(t_i) - 0,5}{1 - 0,5}$$

Seguindo a mesma ideia do MDL-CF, o DFS foi inserido na equação principal do classificador MDL, resultando no algoritmo de classificação de texto MDL-DFS. Nele, o tamanho de descrição de um documento $\vec{d}$ para uma classe c_j é calculado da seguinte forma:

$$L(\vec{d}\,|c_j) = \sum_{i=1}^{|\vec{d}|} L(t_i|c_j) \frac{1}{(1+\alpha) - DFS(t_i)},$$

onde $\alpha > 0$ é uma constante de suavização para evitar que em alguma hipótese o denominador torne-se igual a 0. Neste trabalho, foi usado $\alpha = 10^{-3}$.

IV. Metodologia experimental

Para dar credibilidade aos resultados e tornar os experimentos reprodutíveis, todos os testes foram realizados com as seguintes bases de dados clássicas, reais e públicas:

- *20 Newsgroups*[1]: contém 18.828 documentos extraídos de vinte grupos de discussão da Usenet.
- *TechTC-300*[2]: coleção com diversas bases de dados textuais. Cada base possui duas categorias. Foram realizados experimentos com duas bases dessa coleção: a Exp-1092-135724 e a Exp-1092-789236.
- *7sectors*[3]: composta por 3.417 documentos HTML divididos em uma hierarquia de setores industriais: são sete setores principais com diversos subsetores.
- *Reuters-21578*[4]: composta por 21.578 documentos que foram publicados na agência de notícias Reuters em 1987, divididos em 135 tópicos. Os documentos podem pertencer a mais de um tópico ou a nenhum.
- *WebKB*[5]: composta por páginas *Web* extraídas dos sites dos departamentos de computação das Universidade de Cornell (867), Texas (827), Washington (1.205) e Winconsin (1.263). Existem também 4.120 páginas coletadas de outras universidades. As páginas estão classificadas

[1] A base de dados 20 Newsgroups está disponível em http://qwone.com/~jason/20Newsgroups/.

[2] A base de dados *Tech300* está disponível em http://techtc.cs.technion.ac.il/.

[3] A base de dados 7sectors está disponível em http://goo.gl/r1HVk9.

[4] A base de dados Reuters-21578 está disponível em http://goo.gl/mJJTAI.

[5] A base de dados WebKB está disponível em http://www.cs.cmu.edu/~webkb/.

em sete categorias: *student* (1.641), *faculty* (1.124), *staff* (137), *department* (182), *course* (930), *project* (504) e *other* (3.764).

A. Pré-processamento e tokenização

Os documentos das bases de dados 7sectors e WebKB são páginas HTML. Por isso, foi feita a extração do conteúdo dessas páginas (*parser*) por meio da biblioteca `Beautiful Soup`[6] disponível para Python. Ainda, para a base de dados WebKB, devido à pouca quantidade de dados de treinamento, foram desconsideradas as amostras pertencentes as classes *others*, *staff* e *department*.

Para a base de dados Reuters-21578 foram removidos os documentos que não são rotulados e também aqueles que são rotulados como pertencentes a mais de um tópico ou classe.

Na etapa de pré-processamento todos os textos foram convertidos para letras minúsculas. Além disso, as *stopwords* foram removidas utilizando-se a biblioteca `NLTK (Natural Language Toolkit)`[7] disponível para Python. A técnica de pré-processamento *stemming* não foi empregada pois alguns estudos reportaram que ela apresenta pouco impacto ou até mesmo impacto negativo nos resultados da classificação [29], [30], [31]. Na etapa de tokenização, os delimitadores utilizados foram quaisquer caracteres não alfanuméricos.

B. Medidas de desempenho

Para avaliar o desempenho dos algoritmos, foram realizadas 10 rodadas de validação cruzada k-*fold* estratificada, com $k = 5$. O k-*fold* estratificado é uma variação do k-*fold* tradicional que preserva a proporção de exemplos em cada classe.

Dada a seguinte matriz de confusão generalizada para problemas multiclasses (Tabela I), para cada classe c_j foi calculada a quantidade de:

Tabela I
MATRIZ DE CONFUSÃO PARA PROBLEMAS MULTICLASSES.

	Classe atual			
	c_1	c_2	...	c_m
Classificado como c_1	n_{11}	n_{12}	...	n_{1m}
Classificado como c_2	n_{21}	n_{22}	...	n_{2m}
⋮	⋮	⋮	⋱	⋮
Classificado como c_m	n_{m1}	n_{m2}	...	n_{mm}

- *Verdadeiros positivos* $\rightarrow TP_j = n_{jj}$: quantidade de exemplos corretamente classificados como pertencentes à classe c_j;
- *Falsos positivos* $\rightarrow FP_j = \sum_{k=1|k\neq j}^{m} n_{jk}$: quantidade de exemplos incorretamente classificados como pertencentes à classe c_j;
- *Verdadeiros negativos* $\rightarrow TN_j = \sum_{i=1|i\neq j}^{m} \sum_{k=1|k\neq j}^{m} n_{i_k}$: quantidade de exemplos corretamente classificados como não pertencentes à classe c_j;
- *Falsos negativos* $\rightarrow FN_j = \sum_{k=1|k\neq j}^{m} n_{kj}$: quantidade de exemplos da classe c_j incorretamente classificados como pertencentes a outra classe.

Posteriormente, foram calculadas as seguintes medidas de desempenho amplamente utilizadas na literatura [32], [33], [13]:

- Acurácia $= \frac{1}{m} \sum_{j=1}^{m} \frac{TP_j+TN_j}{TP_j+TN_j+FP_j+FN_j}$;
- Sensitividade ou *recall* $= \frac{1}{m} \sum_{j=1}^{m} \frac{TP_j}{TP_j+FN_j}$;
- Especificidade $= \frac{1}{m} \sum_{j=1}^{m} \frac{TN_j}{TN_j+FP_j}$;
- Precisão $= \frac{1}{m} \sum_{j=1}^{m} \frac{TP_j}{TP_j+FP_j}$;
- F-medida $= 2 \cdot \frac{\text{precisão} \cdot \text{sensitividade}}{\text{precisão} + \text{sensitividade}}$.

O cálculo das medidas de desempenho, usando as equações apresentadas acima, utilizam a técnica conhecida como *macro-averaging*, em que o resultado final é a média dos resultados obtidos para cada classe do problema [9], [32].

C. Métodos

Os resultados obtidos pelos métodos propostos foram comparados com o *naïve* Bayes multinomial (NB) [34], [35] e máquinas de vetores de suporte (SVM) [36], [15]. Esses métodos foram escolhidos para comparação, pois diversos trabalhos consolidados presentes na literatura indicam claramente que esses são os melhores métodos para categorização de textos atualmente disponíveis [5], [12], [13], [14].

Todos os métodos foram implementados usando o software MATLAB. O SVM foi implementado usando a biblioteca `LibSVM` [37] disponível para MATLAB e os experimentos foram executados usando-se *kernel* linear com os parâmetros padrões da biblioteca. Foi escolhido esse *kernel* para o SVM pois quando a dimensionalidade do espaço de atributos é muito alta, o SVM linear, em geral, obtém melhor performance, se comparado aos outros tipos de *kernel* [15], [16], [13].

Em relação aos métodos baseados no princípio MDL, foi empiricamente adotado o parâmetro $|\Omega| = 2^7$ nos experimentos com a base de dados WebKB. Para as outras bases de dados foi usado $|\Omega| = 2^{10}$.

Os experimentos foram realizados em um PC Intel Quad-Core i7 2,93 GHz, 8 Gb de RAM e Windows 7.

V. RESULTADOS

A Tabela II apresenta os resultados obtidos por cada método avaliado na categorização dos documentos das bases de dados 20Newsgroups, TechTC-300-Exp-1092-135724, TechTC-300-Exp-1092-789236, 7sectors, Reuters-21578 e WebKB, respectivamente.

A tabela apresenta a média de cada medida de desempenho e do custo computacional (em segundos) consumido por cada classificador em 10 rodadas da validação cruzada 5-*fold* estratificada. Os resultados estão ordenados por F-medida e os valores em negrito indicam os melhores resultados.

De maneira geral, os métodos MDL-CF e MDL-DFS obtiveram bom desempenho em todas as bases de dados. Eles foram superiores aos métodos considerados estado-da-arte em categorização de textos (NB e SVM) em três das seis bases de dados avaliadas. O SVM linear obteve melhor desempenho

[6] A biblioteca `Beautiful Soup` está disponível em http://www.crummy.com/software/BeautifulSoup/.

[7] A biblioteca `NLTK (Natural Language Toolkit)` está disponível em http://www.nltk.org/.

Tabela II
RESULTADOS OBTIDOS NA CATEGORIZAÇÃO DOS TEXTOS DE CADA BASE DE DADOS.

Método	Acurácia	Sensitividade	Especificidade	Precisão	F-medida	Tempo
20Newsgroups						
MDL-CF	**0.913**	**0.910**	**0.995**	**0.916**	**0.913**	71.420
MDL-DFS	0.904	0.902	**0.995**	0.909	0.906	72.242
SVM	0.880	0.879	0.994	0.883	0.880	131.515
NB	0.877	0.869	0.994	0.890	0.879	**69.471**
TechTC-300-Exp-1092-135724						
MDL-CF	**0.973**	**0.973**	**0.973**	**0.973**	**0.973**	0.240
MDL-DFS	0.972	0.972	0.972	0.972	0.972	0.234
SVM	0.957	0.957	0.957	0.958	0.958	0.251
NB	0.950	0.951	0.951	0.952	0.951	**0.219**
TechTC-300-Exp-1092-789236						
MDL-CF	**0.988**	**0.988**	**0.988**	**0.988**	**0.988**	0.265
MDL-DFS	0.984	0.985	0.985	0.984	0.984	0.259
SVM	0.971	0.972	0.972	0.971	0.971	0.248
NB	0.969	0.971	0.971	0.969	0.970	**0.242**
7sectors						
SVM	**0.930**	**0.918**	**0.988**	**0.935**	**0.927**	10.026
MDL-CF	0.884	0.858	0.979	0.897	0.877	2.433
MDL-DFS	0.873	0.846	0.977	0.889	0.867	2.425
NB	0.738	0.617	0.952	0.841	0.712	**2.206**
Reuters-21578						
SVM	**0.962**	**0.920**	**0.995**	**0.943**	**0.931**	7.676
MDL-DFS	0.924	0.872	0.990	0.896	0.884	2.769
MDL-CF	0.927	0.838	0.990	0.902	0.869	2.652
NB	0.912	0.761	0.988	0.908	0.828	**2.413**
WebKB						
SVM	**0.932**	**0.921**	**0.976**	**0.932**	**0.926**	5.615
MDL-CF	0.869	0.851	0.953	0.861	0.856	1.575
MDL-DFS	0.861	0.848	0.951	0.847	0.847	1.559
NB	0.857	0.804	0.947	0.878	0.839	**1.475**

apenas nas bases de dados 7Sectors, Reuters-21578 e WebKB. Contudo, para oferecer maior credibilidade na comparação dos métodos, foi realizada a análise estatística através do teste não paramétrico de Friedman, seguindo cuidadosamente a metodologia descrita em Demšar [38], usando como referência a F-medida obtida pelos métodos.

A Tabela III apresenta os *rankings* dos algoritmos avaliados de acordo com a F-medida obtida nos experimentos realizados com cada base de dados.

Seja k a quantidade de algoritmos avaliados, n a quantidade de bases de dados e R_j a média dos *ranks* do j-ésimo algoritmo. O teste de Friedman verifica se a hipótese nula, que afirma que todos os algoritmos possuem desempenhos equivalentes, pode ser rejeitada. Para isso, é empregada a seguinte equação:

$$\chi_F^2 = \frac{12n}{k(k+1)} \sum_{j=1}^{k} \left(R_j^2 - \frac{k+1}{2} \right)^2.$$

A hipótese nula é rejeitada se o valor crítico na distribuição χ^2, com $k-1$ graus de liberdade, é menor que χ_F^2, para n e k grandes o suficiente [38]. Para $n <= 10$ e $k <= 5$, Demšar [38] recomenda que sejam usados os valores computados por Zar [39, pag. 763]. Para um intervalo de confiança $\alpha = 0,05$, $k = 4$ e $n = 6$, o valor crítico é $7,600$. Portanto, como $\chi_F^2 = 10,450$, a hipótese nula pode ser rejeitada.

Uma vez que a hipótese nula pôde ser rejeitada, foi realizado um teste *post-hoc* para comparar o desempenho individual dos métodos de classificação. Para isso, foi usado o teste de Bonferroni-Dunn que afirma que os desempenhos de dois algoritmos são significativamente divergentes se a diferença da média dos *ranks* entre eles for maior ou igual à diferença crítica, calculada por [40], [38]:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}},$$

onde q_α corresponde ao valor crítico apresentado em Demšar [38, pag. 12]. Para um intervalo de confiança $\alpha = 0,05$, $CD = 1,784$. Portanto, é possível afirmar que com um grau de confiança de 95%, o desempenho do MDL-CF é significativamente superior ao do *naïve* Bayes. Por outro lado, ambos os métodos MDL-CF e MDL-DFS possuem desempenho estatisticamente equivalente ao SVM.

Dado que as aplicações reais de categorização de textos geralmente envolvem o processamento de um grande volume de amostras, diversos métodos de classificação consolidados pela literatura podem não ser adequados para serem empregados na maioria dessas aplicações, pois o processo de aprendizado pode ter um custo computacional muito elevado e, além disso, nesses casos, o aprendizado incremental é extremamente necessário. Dessa forma, são cada vez mais

Tabela III
RANKINGS DOS MÉTODOS AVALIADOS DE ACORDO COM A F-MEDIDA OBTIDA EM CADA BASE DE DADOS.

	MDL-CF	MDL-DFS	SVM	NB
20Newsgrups	1	2	3.5	3.5
TechTC-300-Exp-1092-1110	1	2	3	4
TechTC-300-Exp-1092-135724	1	2	3	4
7Sectors	2	3	1	4
Reuters-21578	3	2	1	4
WebKB	2	3	1	4
Rank médio	1,666	2,333	2,083	3,916

desejáveis métodos de classificação de textos que tenham um processo de aprendizado rápido e incremental e, ao mesmo tempo, apresentem um alto poder de predição. Diante disso, também foi comparada a eficiência dos métodos por meio da análise da média da sua velocidade de aprendizado (em segundos), medida em cada *fold*.

Os valores apresentados nas tabelas de resultados indicam claramente que o SVM tem custo computacional muito superior aos métodos MDL. Por exemplo, nos experimentos com a base de dados 20Newsgroups, o tempo computacional do SVM foi quase o dobro do tempo do MDL-CF e MDL-DFS. Já nos experimentos com as base de dados 7sectors, Reuters e WebKB, os métodos baseados no princípio MDL foram, respectivamente, $4,12$, $2,77$ e $3,56$ vezes mais rápidos que o SVM. Além disso, como o SVM não oferece aprendizado incremental, seria necessário realizar a etapa de treinamento completa sempre que um novo documento for acrescentado ao conjunto de treino, fato que aumenta consideravelmente o custo computacional e inviabiliza sua aplicação em cenários reais e dinâmicos. Por outro lado, os métodos MDL e *naïve* Bayes são eficientes e naturalmente oferecem aprendizado incremental, o que possibilita sua aplicação nesses cenários.

Em resumo, os resultados obtidos indicam que os métodos MDL oferecem um excelente balanceamento entre poder de predição, generalização e custo computacional. Embora eles sejam estatisticamente equivalentes ao SVM em termos de poder de predição, eles são claramente mais eficientes e possuem menor custo computacional. Além disso, ao contrário do SVM, os métodos propostos podem ser treinados de forma incremental, característica desejada para a maioria das aplicações reais de categorização de textos.

VI. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foram propostos dois métodos de categorização de textos baseados no princípio da descrição mais simples. As técnicas apresentadas são simples de implementar, podem ser treinadas de forma incremental, possuem baixo custo computacional e, além disso, oferecem proteção natural contra *overfitting*.

Para avaliar o desempenho dos métodos propostos, foram conduzidos experimentos com seis bases de dados grandes, reais, públicas e bem conhecidas pela literatura. Além disso, os resultados obtidos foram comparados com os métodos considerados estado-da-arte em categorização de textos: *naïve* Bayes multinomial e máquinas de vetores de suporte com *kernel* linear.

A análise estatística dos resultados indicou que o desempenho obtido pelos métodos MDL foi equivalente ao SVM, em termos de capacidade preditiva. Porém, os métodos MDL foram bem mais eficientes em termos de custo computacional, além de oferecerem a possibilidade de realizar aprendizado incremental. Dessa forma, os métodos propostos demonstraram oferecer um bom balanceamento entre poder preditivo e eficiência. Tais resultados evidenciam que eles possuem características desejadas em aplicações reais de categorização de textos *online* e de larga escala.

Como trabalho futuro, pretende-se conduzir experimentos com mais bases de dados textuais e propor outras técnicas de atribuição de relevância para os *tokens*, além do CF e DFS.

AGRADECIMENTOS

Os autores são gratos à Fapesp e CNPq pelo apoio financeiro ao desenvolvimento desse projeto.

REFERÊNCIAS

[1] S. Jiang, G. Pang, M. Wu, and L. Kuang, "An improved K-nearest-neighbor algorithm for text categorization," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1503–1509, Jan. 2012.

[2] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Using KNN model for automatic text categorization," *Soft Computing*, vol. 10, no. 5, pp. 423–430, Mar. 2006.

[3] D. D. Lewis and M. Ringuette, "A comparison of two learning algorithms for text categorization," in *Proceedings of the 3rd annual symposium on document analysis and information retrieval (SDAIR'94)*, vol. 33, Las Vegas, NV, Apr. 1994, pp. 81–93.

[4] D. Hull, "Improving text retrieval for the routing problem using latent semantic indexing," in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*. Dublin, Ireland: ACM/Springer, Jul. 1994, pp. 282–291.

[5] T. Joachims, "Text categorization with suport vector machines: Learning with many relevant features," in *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*. Chemnitz, Germany: Springer, Apr. 1998, pp. 137–142.

[6] Z. Sun, Y. Ye, W. Deng, and Z. Huang, "A cluster tree method for text categorization," *Procedia Engineering*, vol. 15, pp. 3785–3790, Dec. 2011.

[7] E. Wiener, J. O. Pedersen, and A. S. Weigend, "A neural network approach to topic spotting," in *Proceedings of the 4th annual symposium on document analysis and information retrieval (SDAIR'95)*, Las Vegas, NV, Apr. 1995, pp. 317–332.

[8] H. T. Ng, W. B. Goh, and K. L. Low, "Feature selection, perceptron learning, and a usability case study for text categorization," in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*. Philadelphia, PA, USA: ACM, Jul. 1997, pp. 67–73.

[9] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, Mar. 2002.

[10] F. Assis, W. Yerazunis, C. Siefkes, and S. Chhabra, "Exponential differential document count – a feature selection factor for improving Bayesian filters accuracy," in *Proceedings of the 2006 MIT Spam Conference (SP'06)*, Cambridge, MA, USA, 2006, pp. 1–6.

[11] T. A. Almeida and A. Yamakami, "Facing the spammers: A very effective approach to avoid junk e-mails," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6557–6561, Jun. 2012.

[12] 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)*. Washington, DC, USA: ACM, Nov. 1998, pp. 148–155.

[13] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2011.

[14] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Online learning from imbalanced data streams," in *Proceedings of the 3rd International Conference of Soft Computing and Pattern Recognition (SoCPaR'11)*. Dalian, China: IEEE, Oct. 2011, pp. 347–352.

[15] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," National Taiwan University, Tech. Rep., 2003.

[16] R. Caruana, N. Karampatziakis, and A. Yessenalina, "An empirical evaluation of supervised learning in high dimensions," in *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. Helsinki, Finland: ACM, Jun. 2008, pp. 96–103.

[17] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, Sep. 1978.

[18] ——, "A universal prior for integers and estimation by minimum description length," *The Annals of Statistics*, vol. 11, no. 2, pp. 416–431, Jun. 1983.

[19] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, Mar. 2002.

[20] T. A. Almeida, A. Yamakami, and J. Almeida, "Filtering spams using the minimum description length principle," in *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC'10)*. Sierre, Switzerland: ACM, Mar. 2010, pp. 1854–1858.

[21] A. K. Uysal and S. Gunal, "A novel probabilistic feature selection method for text classification," *Knowledge-Based Systems*, vol. 36, pp. 226–235, Dec. 2012.

[22] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Transaction on Information Theory*, vol. 44, no. 6, pp. 2743–2760, Oct. 1998.

[23] P. D. Grünwald, I. J. Myung, and M. A. Pitt, *Advances in Minimum Description Length: Theory and Applications*. The MIT Press, 2005.

[24] P. Domingos, "The role of Occam's razor in knowledge discovery," *Data Mining and Knowledge Discovery*, vol. 3, pp. 409–425, 1999.

[25] A. N. Kolmogorov, "Three approaches to the quantitative definition of information," *Problems of Information Transmission*, vol. 1, pp. 1–7, 1965.

[26] M. Li and P. M. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 3rd ed. New York, NY: Springer, 2008.

[27] P. D. Grünwald, "A tutorial introduction to the minimum description length principle," in *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

[28] J. Rissanen, "Fisher information and stochastic complexity," *IEEE Transaction on Information Theory*, vol. 42, no. 1, pp. 40–47, Jan. 1996.

[29] F. Song, S. Liu, and J. Yang, "A comparative study on text representation schemes in text categorization," *Pattern Analysis and Applications*, vol. 8, no. 1–2, pp. 199–209, Jul. 2005.

[30] J. Pomikálek and R. Řehuřek, "The influence of preprocessing parameters on text categorization," *International Journal of Applied Science, Engineering and Technology*, pp. 430–434.

[31] J. R. Méndez, E. L. Iglesias, F. Fdez-Riverola, F. Díaz, and J. M. Corchado, "Tokenising, stemming and stopword removal on anti-spam filtering domain," in *Proceedings of the 11th Spanish Association Conference on Current Topics in Artificial Intelligence (CAEPIA'05)*. Santiago de Compostela, Spain: Springer-Verlag, Nov. 2006, pp. 449–458.

[32] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, Jul. 2009.

[33] T. A. Almeida, A. Yamakami, and J. Almeida, "Spam filtering: how the dimensionality reduction affects the accuracy of naive Bayes classifiers," *Journal of Internet Services and Applications*, vol. 1, no. 3, pp. 183–200, Feb. 2011.

[34] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in *Proceedings of the 15th AAAI Workshop on Learning for Text Categorization (AAAI'98)*, Madison, Wisconsin, Jul. 1998, pp. 41–48.

[35] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2009.

[36] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[37] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, Apr. 2011.

[38] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, Dec. 2006.

[39] J. H. Zar, *Biostatistical Analysis*, 5th ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2010.

[40] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability," *Soft Computing*, vol. 13, no. 10, pp. 959–977, Apr. 2009.