`MDLText` aplicado na Filtragem Automática de SPIM e SMS Spam

Title: `MDLText` applied to automatic filtering of SPIM and SMS Spam

Renato Moraes Silva[1], **Tiago A. Almeida**[2], **Akebo Yamakami**[1]

[1]Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP) – Campinas, SP, Brasil

[2]Departamento de Computação (DComp)
Universidade Federal de São Carlos (UFSCar) – Sorocaba, SP, Brasil

{renatoms,akebo}@dt.fee.unicamp.br, talmeida@ufscar.br

Abstract. *Spam filtering in online instant messages and SMS is a challenging problem nowadays. It is because the messages are often very short and rife with slangs, idioms, symbols, emoticons, and abbreviations which hamper predicting and knowledge discovering. In order to face this problem, we evaluated a simple, fast, scalable, multiclass, and online text classification method based on the minimum description length principle. We conducted experiments using a real and public dataset, which demonstrate that our method is effective on instant messaging and SMS spam filtering in both online and offline learning contexts.*

Keywords. *Online learning; Occam's razor; Text categorization; Machine learning.*

Resumo. *A filtragem automática de spam em mensagens instantâneas e SMS é um problema desafiador, pois as mensagens são frequentemente curtas e repletas de ruídos, tais como gírias, expressões idiomáticas, símbolos, emoticons e abreviações, o que dificulta a extração de conhecimento e predição. Para enfrentar esse problema, neste artigo é avaliado um método de classificação de texto baseado no princípio da descrição mais simples, que é eficiente, rápido, escalável, multiclasse e possui aprendizado incremental. Experimentos realizados com uma base de dados real e pública, em cenários de aprendizado online e offline, indicam que o método proposto é promissor para a tarefa de detecção de spam em mensagens instantâneas e SMS.*

Palavras-Chave. *Aprendizado online; Navalha de Occam; Categorização de texto; Aprendizado de máquina.*

1. Introdução

A popularidade do serviço de mensagens curtas (SMS – *Short Message Service*) e das mensagens instantâneas compartilhadas por meio de dispositivos móveis tem aumentado cada vez mais. Isso tem atraído a atenção dos *spammers* que passaram a enviar mensagens indiscriminadamente sem o consentimento dos usuários, causando vários problemas. Quando é enviada por meio de serviços de SMS, esse tipo de mensagem é chamada de

SMS spam. Porém, quando o envio ocorre por meio de serviços de mensagens instantâneas, tais como WhatsApp, Hangouts e Facebook Messenger, ela é chamada de SPIM (*SPam over Instant Messaging*). A Figura 1 apresenta alguns exemplos.

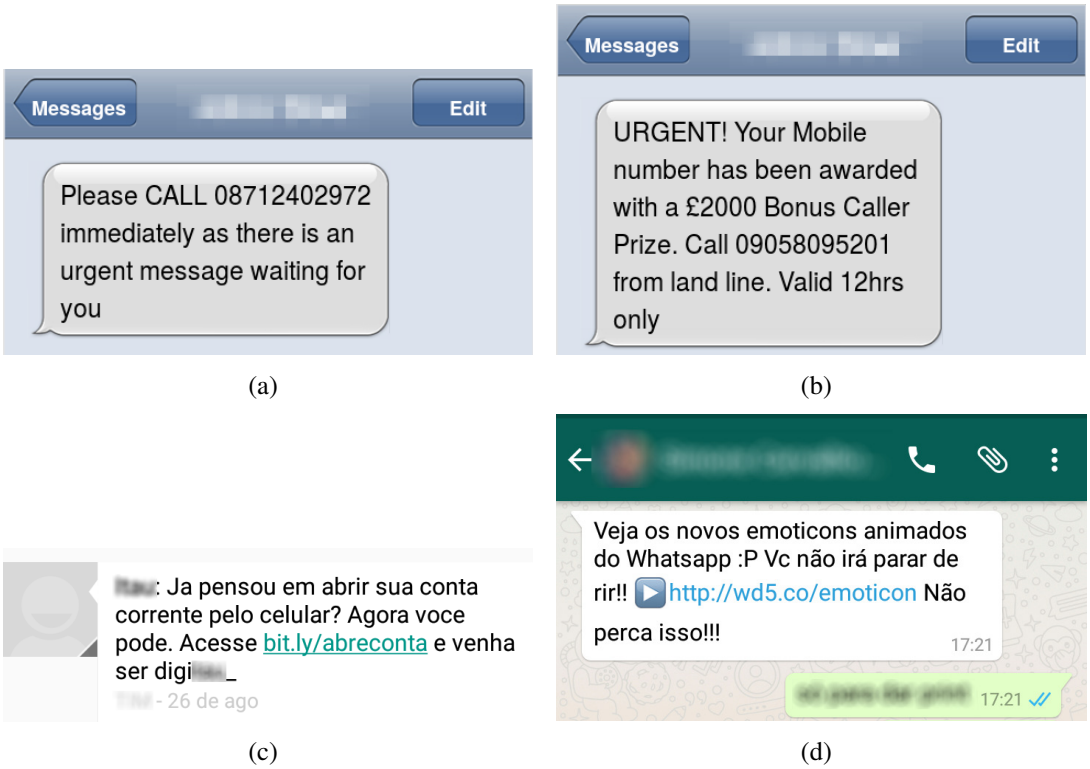


Figura 1. Exemplos de SPIM e SMS spam. As mensagens ilustradas nas Figuras 1(a) e 1(b) foram extraídas da base de dados *SMS Spam Collection* [Almeida et al. 2011a]. As demais mensagens foram recebidas por um dos autores deste artigo.

Essas mensagens podem provocar desperdício de tempo para serem lidas e apagadas e gerar aborrecimento aos usuários, pois normalmente são propagandas não solicitadas de produtos, serviços e descontos. Elas também podem causar prejuízos financeiros, pois muitas delas contém *links* para faturas falsas, promessas enganosas de empréstimos, investimentos ou perdão de dívidas, falsos cupons de desconto e falsos prêmios em concursos e jogos de loteria. Esse tipo de mensagem geralmente engana os usuários para que eles paguem taxas desnecessárias ou compartilhem informações pessoais e financeiras que, posteriormente, podem ser usadas para realizar transações bancárias não autorizadas [Almeida et al. 2011a, Goswami et al. 2016, Abdulhamid et al. 2017].

SMS spam e SPIM também podem ser usados para a prática de outros crimes cibernéticos, uma vez que os *spammers* podem inserir *links* para softwares maliciosos que ao serem instalados no dispositivo eletrônico do usuário podem violar sua privacidade [Abdulhamid et al. 2017]. Informações pessoais, conversas por aplicativos de mensagens, fotos e vídeos podem ser obtidos de forma ilegal por esses softwares e compartilhados na Internet. Isso expõe os usuários ao risco de sofrerem extorsão, assédio virtual (*cyberbullying*) e outras práticas criminosas que são capazes de prejudicar seriamente suas vidas, causando exclusão social, danos psicológicos ou até mesmo levando-os a cometerem suicídio [Gomez-Martin 2012, Carpenter e Hubbard 2014, Tsakalidis e Vergidis 2017].

Vários métodos têm sido avaliados para filtrar SMS spam e SPIM e grande parte deles emprega técnicas de aprendizado de máquina para classificar as mensagens como spam ou *ham* (legítima). Por exemplo, [Almeida et al. 2011a] propuseram uma base de dados de SMS e a utilizaram para comparar o desempenho de diversos métodos de aprendizado de máquina. Os autores demostraram que o método de máquinas de vetores de suporte (SVM – *Support Vector Machines*) obteve o melhor desempenho na tarefa. Já, no trabalho de [Uysal et al. 2012], foi proposto um *framework* para filtrar SMS que usa as métricas de ganho de informação e χ^2 para selecionar atributos discriminativos para o problema. Os atributos selecionados foram usados para treinar dois métodos de classificação bayesianos. Em outro estudo, [Ahmed et al. 2015] propuseram um método semi-supervisionado que explora conjuntos frequentes de itens e aprendizado *ensemble* para detectar SMS spam. Nos experimentos conduzidos pelos autores, os métodos *naïve* Bayes multinomial (M.NB), floresta aleatória (RF – *Random Forest*) e SVM foram usados como base para o aprendizado *ensemble*. Em um trabalho mais recente, [Goswami et al. 2016] usaram atributos estilísticos e atributos textuais específicos para mensagens de texto curtas em conjunto com o método SVM para classificar SMS.

Quanto ao problema de SPIM, [Liu et al. 2005] e [Bi et al. 2008] inicialmente estudaram mensagens instantâneas enviadas por serviços de chat *online*, tais como MSN e Yahoo. A disseminação de SPIM através de dispositivos móveis começou a ser estudado apenas em trabalhos mais recentes. Por exemplo, [Das et al. 2012] estudaram os efeitos da inclusão de um módulo de detecção de SPIM em uma rede LTE. Eles compararam o desempenho e o tempo de treinamento e teste dos métodos *naïve* Bayes, SVM, AdaBoost e J48, quando incorporados ao módulo de detecção de SPIM. Outra contribuição recente foi apresentada por [Almeida et al. 2016]. Eles propuseram um *framework* para a normalização e enriquecimento de textos curtos, que combina técnicas de normalização léxica e detecção de contexto, com o objetivo de obter melhores atributos e aprimorar o desempenho dos métodos de filtragem de SPIM e SMS spam. A Tabela 1 sumariza os trabalhos que abordaram esses problemas.

Tabela 1. Trabalhos correlatos que abordaram a filtragem de SPIM e SMS spam.

Referência	Problema abordado	
	SMS spam	SPIM
[Liu et al. 2005]		X
[Bi et al. 2008]		X
[Almeida et al. 2011a]	X	
[Uysal et al. 2012]	X	
[Das et al. 2012]		X
[Ahmed et al. 2015]	X	
[Goswami et al. 2016]	X	
[Almeida et al. 2016]	X	X
[Silva et al. 2016b]	X	X

Apesar de algumas técnicas tradicionais já terem sido avaliadas, a tarefa de classificar essas mensagens ainda oferece desafios. As técnicas consideradas referência para a classificação de spam por e-mail apresentaram desempenho degradado quando aplicadas na filtragem de SMS spam e SPIM [Almeida et al. 2011a]. Uma das causas é o baixo número de termos por mensagens, o que dificulta o aprendizado dos métodos de classificação. Além disso, o conteúdo das mensagens geralmente é repleto de ruídos, tais

como expressões idiomáticas, gírias, símbolos, *emoticons* e abreviações que dificultam até mesmo o processo de *tokenização*. A maioria dos trabalhos da literatura não tratou esse problema. Portanto, é necessário estudar se técnicas de pré-processamento e enriquecimento textual, capazes de diminuir o ruído e aumentar o número de informações disponíveis para serem usadas pelos métodos de classificação, podem ajudar a aumentar a eficiência na detecção de SMS Spam e SPIM.

A maioria dos trabalhos da literatura usa métodos de classificação que requerem que todos os exemplos sejam apresentados simultaneamente e mantidos em memória, em um processo conhecido como aprendizado *offline*. Tais métodos não são adequados para tratar problemas reais de SMS spam e SPIM. Portanto, existe uma carência de trabalhos que estudem esse assunto usando métodos de classificação *online*, que possam ser treinados de forma incremental.

Para contribuir com a solução desse problema, em [Silva et al. 2016b] foi analisado o desempenho do `MDLText`, um novo método de classificação de texto escalável, multiclasse, que possui aprendizado incremental e que realiza seleção de modelos baseada no princípio da descrição mais simples (MDL – *minimum description length*). Na maioria dos experimentos, o método proposto obteve melhores resultados na classificação de spam disseminado através de mensagens de texto curtas do que outros métodos considerados o estado-da-arte na solução do problema. Também foi analisado o impacto das técnicas de remoção de *stopwords* e *stemming* no desempenho dos métodos. Os resultados mostraram que a remoção de *stopwords* foi prejudicial, enquanto a técnica de *stemming* melhorou sutilmente os resultados. Outros resultados preliminares relativos ao `MDLText` foram publicados em [Silva et al. 2015, Silva et al. 2016a, Silva et al. 2017].

Possivelmente, os resultados obtidos por [Silva et al. 2016b] poderiam ter sido melhores se outras técnicas de pré-processamento para normalização dos termos das mensagens e enriquecimento textual tivessem sido utilizadas. Além disso, o método proposto pelos autores emprega uma função de penalização baseada na relevância de cada termo na separação das classes. A técnica empregada por [Silva et al. 2016b] para calcular a relevância dos termos foi o seletor de atributos distintivos (DFS – *distinguishing feature selector*). Porém, existe a hipótese de que o uso de outras técnicas capazes de expressar melhor a importância dos termos possa impactar positivamente a capacidade de predição do `MDLText`. Diante disso, neste artigo, o estudo do desempenho do `MDLText` na filtragem automática de SPIM e SMS spam foi ampliado e as seguintes contribuições são oferecidas:

- foram avaliadas diversas técnicas para calcular a pontuação dos termos que é usada na função de penalização empregada pelo método `MDLText`;
- foi analisado se o emprego de técnicas de pré-processamento, tais como normalização léxica e indexação semântica, podem trazer benefícios à tarefa de predição;
- em cada experimento, foi aplicada busca em grade para encontrar o melhor esquema de atribuição de pesos para os termos: frequência do termo (TF – *term frequency*), frequência do termo-frequência inversa dos documentos (TF-IDF – *term frequency-inverse document frequency*) ou binário. O objetivo foi encontrar o melhor resultado possível para cada método avaliado.

O restante deste trabalho está estruturado da seguinte forma: na Seção 2, é discutida a relação entre o problema de seleção de modelos e o aprendizado de máquina e são

apresentados conceitos básicos sobre o princípio MDL. Na Seção 3, é descrito o método de classificação de texto `MDLText`. As configurações adotadas nos experimentos são detalhadas na Seção 4. Os resultados obtidos na classificação *online* e *offline* são apresentados na Seção 5. Por fim, na Seção 6, são apresentadas as conclusões e direcionamentos para trabalhos futuros.

2. Seleção de modelos baseada na Navalha de Occam

Métodos de aprendizado de máquina podem ser vistos como métodos de seleção de modelos, pois dado um conjunto de modelos candidatos, eles procuram escolher um que se ajuste bem aos dados. Para isso, eles usam informações extraídas de um conjunto pré-existente de dados, com o objetivo de fazer boas predições em dados futuros.

No processo de seleção de modelos, um dos problemas que podem ser enfrentados pelos métodos de aprendizado é o dilema *bias*-variância: deseja-se selecionar um modelo que se ajuste bem aos dados observados, mas que ao mesmo tempo seja genérico o suficiente para reproduzir bem o comportamento de dados não observados. Modelos com alta variância geralmente são mais complexos e se ajustam muito bem aos dados conhecidos, porém possuem baixa capacidade de generalização e, portanto, cometem muitos erros em dados não conhecidos (sobreajustamento ou *overfitting*). Modelos com alto *bias* geralmente possuem baixa complexidade e não conseguem capturar regularidades importantes nos dados observados que os ajudem a fazer boas predições em dados futuros (subajustamento ou *underfitting*) [Bishop 2006, Hastie et al. 2009].

Os métodos de aprendizado de máquina podem adotar diferentes estratégias de seleção de modelos e, geralmente fazem uso de parâmetros que ajustam a complexidade do modelo. Por exemplo, métodos de aprendizado que usam funções de *kernel* podem adotar parâmetros que ajustam a largura do *kernel*; métodos baseados em funções polinomiais podem usar parâmetros que ajustam o grau do polinômio; e métodos baseados em vizinhança podem usar um parâmetro que ajusta o número de vizinhos que serão analisados [Hastie et al. 2009]. Porém, existem outros métodos que adotam estratégias de seleção de modelos baseadas no princípio da Navalha de Occam. Esse princípio foi formulado por William de Occam, na Idade Média, como uma crítica à filosofia escolástica que utilizava teorias complexas para explicar a realidade. Hoje em dia, o princípio da navalha de Occam é vastamente usado em metodologia científica, sugerindo que entre duas ou mais hipóteses que possam explicar um mesmo fenômeno, provavelmente a hipótese mais simples seja a mais adequada [Domingos 1999]. Portanto, métodos de aprendizado de máquina baseados nesse princípio consideram que, dado um conjunto de modelos candidatos com capacidade de predição similar, o modelo mais simples provavelmente é a melhor escolha.

O princípio MDL, introduzido por [Rissanen 1978], é uma das principais iniciativas de formalização da Navalha de Occam. Ele foi proposto para o problema de seleção de modelos com a prerrogativa de que o melhor modelo para explicar um conjunto de dados é aquele que oferece uma descrição mais compacta dos dados. A descrição mais compacta é oferecida pelo modelo que mais captura regularidade nos dados, ou seja, por aquele que melhor "conhece" os dados de observação [Grünwald et al. 2005]. A preferência do princípio MDL por modelos menos complexos pode ajudar a evitar o conhecido problema de *overfitting*, o que o torna atrativo para problemas de aprendizado supervisionado.

Matematicamente, dado um conjunto de potenciais modelos $M_1, M_2, ..., M_{|M|}$, o MDL seleciona aquele, tal que:

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

sendo que, $L(M)$ corresponde ao tamanho da descrição do modelo M, representando sua complexidade, e $L(X|M)$ ao tamanho da descrição do dado X quando codificado com M, representando quão bem M descreve X. Em [Rissanen 1978], foi demonstrado que $L(X|M)$ pode ser calculado através do código de Shannon-Fano e, portanto, $L(X|M) = -\log P(X|M)$, onde $P(X|M)$ é a probabilidade condicional de X dado M.

De acordo com [Grünwald et al. 2005], o problema do MDL definido na Eq. 1 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. Para contornar esse problema, [Rissanen 1996] propôs uma nova versão do MDL que usa códigos universais para medir o tamanho da descrição de cada modelo, onde é empregada apenas uma parte do código com tamanho $\bar{L}(X|M)$.

Não existe nenhum procedimento padrão ou universal para calcular o tamanho da descrição de um modelo. Porém, [Silva et al. 2016a, Silva et al. 2016b, Silva et al. 2017] apresentaram uma nova interpretação sobre como calcular o tamanho da descrição do modelo em problemas de classificação de texto. Eles propuseram um novo método, nomeado `MDLText`, que considera que cada classe pode ser representada como um modelo e cada exemplo a ser classificado pode ser representado por um dos modelos disponíveis.

3. O método `MDLText`

Dado um documento de texto d não rotulado, o `MDLText` utiliza a equação principal do princípio MDL (Eq. 1) para predizer a classe à qual o documento pertence. O conjunto de classes $c_1, c_2, \ldots, c_{|C|}$ representa o conjunto de potenciais modelos M, enquanto d representa o dado X. Além disso, d recebe o rótulo j correspondente à classe c_j que possui o menor tamanho de descrição em relação à d:

$$y(d) = \arg\min_{\forall c} L(d|c_j). \tag{2}$$

O `MDLText` ignora o tamanho de descrição das potenciais classes (modelos), pois ele pode tornar-se arbitrário e difícil de ser calculado [Silva et al. 2017]. Assim, o tamanho da descrição de d em relação à classe c_j é calculado multiplicando-se (1) uma penalidade $\hat{S}(d, c_j)$ para a classe classe c_j, (2) a soma do tamanho da descrição de todos os termos de d e (3) uma penalidade $K(t_i)$ para cada termo, o que pode ser definido por:

$$L(d|c_j) = \hat{S}(d, c_j) \times \left[\sum_{i=1}^{|d|} \mathcal{L}(t_i|c_j) \times K(t_i) \right], \tag{3}$$

onde $|d|$ corresponde à quantidade de termos no documento d e $\mathcal{L}(t_i|c_j)$ é o tamanho de descrição do termo t_i dada a classe c_j, que pode ser calculado por:

$$\mathcal{L}(t_i|c_j) = \lceil -\log_2 \beta(t_i|c_j) \rceil. \tag{4}$$

$\beta(t_i|c_j)$ é o peso condicional de t_i dada a classe c_j e é baseado nos pesos TF-

IDF [Wilbur e Kim 2009] do termo t_i em cada documento de treinamento. O peso TF-IDF de um termo t_i em um documento d é dado por:

$$w(t_i, d) = \log(1 + TF(t_i, d)) \times \log\left(\frac{|\mathcal{D}| + 2}{DF_{t_i} + 1}\right), \tag{5}$$

onde $TF(t_i, d)$ é a frequência do termo t_i no documento d, $|\mathcal{D}|$ é a quantidade de documentos no conjunto de treinamento e DF_{t_i} é o número de documentos de treinamento que contém o termo t_i. Então, a normalização L2 é aplicada:

$$\hat{w}(t_i, d) = \frac{w(t_i, d)}{\|w(:, d)\|_2}. \tag{6}$$

Na Eq. 4, o valor de $\beta(t_i|c_j)$ pode ser calculado da seguinte forma:

$$\beta(t_i|c_j) = \frac{n_{c_j,t_i} + \frac{1}{|\Omega|}}{\hat{n}_{c_j} + 1}, \tag{7}$$

onde $n_{c_j,t_i} = \sum_{\forall d} \hat{w}(t_i, d|c_j)$ (soma dos pesos TF-IDF normalizados do termo t_i nos documentos de treinamento da classe c_j) e $\hat{n}_{c_j}$ é a soma de n_{c_j,t_i} para todos os termos que aparecem em documentos pertencentes à classe c_j. O parâmetro $|\Omega|$ é usado para preservar uma porção do tamanho de descrição para termos que nunca apareceram em $\mathcal{D}$ e que possuem, portanto, $n_{c_j,t_i} = 0$. O valor de $L(d|c_j)$ é sensível ao parâmetro $|\Omega|$, pois quando o valor desse parâmetro é alto, o peso condicional de um termo que nunca apareceu em $\mathcal{D}$ é baixo e portanto, o tamanho da descrição desse termo é alto. Então, pode-se afirmar que o parâmetro $|\Omega|$ regula como os termos com $n_{c_j,t_i} = 0$ contribuem no tamanho de descrição de um documento em relação a uma classe c_j. Diante da sua importância, é recomendável usar uma busca em grade (*grid search*) para ajustar esse parâmetro.

Na Eq. 3, a função de penalidade $\hat{S}(d, c_j)$ é baseada na similaridade de cosseno (*cosine similarity*) entre o documento e um vetor protótipo da classe:

$$\hat{S}(d, c_j) = -\log_2\left(\frac{1}{2} \times S(d, \overline{c}_j)\right). \tag{8}$$

A similaridade de cosseno $S(d, \overline{c}_j)$ mede a similaridade entre o documento d e um vetor protótipo $\overline{c}_j$ de uma maneira similar ao que é feito no algoritmo Rocchio [Rocchio 1971, Manning et al. 2009]. A similaridade $S(d, \overline{c}_j)$ varia entre zero e um e valores mais próximos a um indicam que existe grande similaridade. A função $\hat{S}(d, c_j) >= 1$ penaliza a classe com menor similaridade em relação ao documento. Quanto menor a similaridade, maior é o valor obtido em $\hat{S}(d, c_j)$ e, portanto, maior é o tamanho de descrição do documento d dada à classe c_j.

A similaridade de cosseno pode ser calculada por:

$$S(d, \overline{c}_j) = \frac{\sum_{i=1}^{|d|} \hat{w}(t_i, d|c_j) \times \overline{c}_j(t_i)}{\|\hat{w}(:, d)\|_2 \times \|\overline{c}_j\|_2} \tag{9}$$

onde o símbolo $\|.\|_2$ representa a norma Euclidiana e $\hat{w}(:, d)$ corresponde aos pesos normalizados dos termos do documento d. O vetor protótipo $\overline{c}_j$ é formado pela média dos

pesos dos termos.

$$\overline{c}_j(t_i) = \frac{n_{c_j,t_i}}{|\hat{\mathcal{D}}_{c_j}|} \tag{10}$$

onde $|\hat{\mathcal{D}}_{c_j}|$ é a quantidade de documentos de treinamento da classe c_j.

Na Eq. 3, a função de penalidade $K(t_i)$ associada a cada termo pode ser calculada da seguinte maneira:

$$K(t_i) = \frac{1}{(1+\alpha) - F(t_i)}, \tag{11}$$

onde $0 \leq F(t_i) \leq 1$ corresponde à pontuação de contribuição do termo t_i e $\alpha > 0$ é uma constante usada para evitar que o denominador seja zero. Nesse trabalho, empiricamente determinou-se que $\alpha = 10^{-3}$.

Para calcular $F(t_i)$, pode ser usada qualquer função que retorne uma pontuação de relevância do termo entre zero e um, conforme detalhado na Seção 3.2.

3.1. Complexidade computacional do `MDLText`

O estágio de treinamento requer apenas uma passagem completa em cada documento. Portanto, com $|\mathcal{D}^\star|$ documentos e $|\bar{d}|$ termos, onde $|\bar{d}|$ corresponde ao número médio de termos por documento, a complexidade é de ordem linear $\mathcal{O}(|\mathcal{D}^\star| \times |\bar{d}|)$.

No estágio de classificação, para cada documento não rotulado d, o `MDLText` calcula $K(t_i)$ examinando a frequência de cada termo em d para cada classe. Em seguida, o valor de cada termo em cada classe é usado para calcular o tamanho de descrição de d. Finalmente, o valor de cada termo para cada classe também é usado para calcular $\hat{S}(d, c_j)$ e para penalizar o tamanho de descrição de d. Consequentemente, a complexidade computacional é da ordem de $\mathcal{O}(|\mathcal{C}| \times |\bar{d}| + |\mathcal{C}| \times |\bar{d}| + |\mathcal{C}| \times |\bar{d}|)$ ou $\mathcal{O}(3 \times |\mathcal{C}| \times |\bar{d}|)$, onde $|\mathcal{C}|$ corresponde ao número de classes. Dado que $3 \times |\mathcal{C}|$ é constante, a complexidade do estágio de classificação é assintoticamente de ordem linear $\mathcal{O}(|\bar{d}|)$ [Silva et al. 2017].

3.2. Técnicas para calcular a pontuação dos termos

As técnicas que podem ser usadas em $F(t_i)$ (Eq. 11), para calcular a pontuação dos termos, visam retornar um valor que procura refletir a "importância" de cada termo na predição das classes do problema com base na frequência do termo em cada classe. Por exemplo, em [Silva et al. 2016b], a técnica usada foi o seletor de atributos distintivos (DFS – *distinguishing feature selector*) [Uysal e Gunal 2012]. Conforme apresentado pelos autores, o `MDLText` obteve resultados promissores na detecção automática de SPIM e SMS spam quando essa técnica foi usada para calcular $F(t_i)$.

Neste estudo, foi avaliado o desempenho do `MDLText` com a aplicação de outras técnicas de atribuição de pontuação para os termos. As técnicas avaliadas foram as seguintes: fatores de confidência (CF – *confidence factors*) [Assis et al. 2006], índice de Gini (GI – Gini *index*) [Shang et al. 2007], ganho de informação (IG – *information gain*) [Sebastiani 2002, Yang e Pedersen 1997], chi-quadrado (*chi-square*) ou χ^2 [Sebastiani 2002, Yang e Pedersen 1997], coeficiente NGL (Ng-Goh-Low) [Ng et al. 1997], coeficiente GSS (Galavotti-Sebastiani-Simi) [Galavotti et al. 2000] e razão de chances (OR – *odds ratio*) [Sebastiani 2002, Yang e Pedersen 1997]. Cada uma delas está sumarizada na Tabela 2.

Tabela 2. Técnicas de atribuição de pontuação para os termos.

Técnica	Equação
CF	$\mathcal{F}(t_i) = \frac{1}{\lvert\mathcal{C}\rvert-1} \times \sum_{\forall j \lvert j \neq \tau} \frac{\left(\frac{\left(\phi_{c_\tau,t_i} - \phi_{c_j,t_i}\right)^2 + \left(\phi_{c_\tau,t_i} \times \phi_{c_j,t_i}\right) - \frac{\lambda_1}{\phi_{c_\tau,t_i} + \phi_{c_j,t_i}}}{\left(\phi_{c_\tau,t_i} + \phi_{c_j,t_i}\right)^2} \right)^{\lambda_2}}{1 + \left(\frac{\lambda_3}{\phi_{c_\tau,t_i} + \phi_{c_j,t_i}} \right)}$
DFS	$\mathcal{F}(t_i) = \sum_{\forall j} \frac{p(c_j\lvert t_i)}{p(\bar{t}_i\lvert c_j)+p(t_i\lvert \bar{c}_j)+1}$
GI	$\mathcal{F}(t_i) = \sum_{\forall j} p(t_i\lvert c_j)^2 \times p(c_j\lvert t_i)^2$
IG	$\mathcal{F}(t_i) = \sum_{\forall j} p(t_i\lvert c_j) \times \log \frac{p(t_i\lvert c_j)}{p(t_i)\times p(c_j)} + p(\bar{t}_i\lvert c_j) \times \log \frac{p(\bar{t}_i\lvert c_j)}{p(\bar{t}_i)\times p(c_j)}$
χ^2	$\mathcal{F}_{\text{local}}(t_i, c_j) = \frac{\lvert\mathcal{D}^\star\rvert \times [p(t_i\lvert c_j)\times p(\bar{t}_i\lvert \bar{c}_j) - p(t_i\lvert \bar{c}_j)\times p(\bar{t}_i\lvert c_j)]^2}{p(t_i)\times p(\bar{t}_i)\times p(c_j)\times p(\bar{c}_j)}$
NGL	$\mathcal{F}_{\text{local}}(t_i, c_j) = \frac{\sqrt{\lvert\mathcal{D}^\star\rvert} \times [p(t_i\lvert c_j)\times p(\bar{t}_i\lvert \bar{c}_j) - p(t_i\lvert \bar{c}_j)\times p(\bar{t}_i\lvert c_j)]}{\sqrt{p(t_i)\times p(\bar{t}_i)\times p(c_j)\times p(\bar{c}_j)}}$
GSS	$\mathcal{F}_{\text{local}}(t_i, c_j) = p(t_i\lvert c_j) \times p(\bar{t}_i\lvert \bar{c}_j) - p(t_i\lvert \bar{c}_j) \times p(\bar{t}_i\lvert c_j)$
OR	$\mathcal{F}_{\text{local}}(t_i, c_j) = \frac{p(t_i\lvert c_j)\times [1-p(t_i\lvert \bar{c}_j)]}{[1-p(t_i\lvert c_j)]\times p(t_i\lvert \bar{c}_j)}$

Na Tabela 2, na função da técnica CF, τ corresponde ao índice da classe mais frequente; $j = 1, \ldots, |\mathcal{C}|$ são os índices das $|\mathcal{C}|$ classes; ϕ_{c_τ,t_i} é o número de documentos que possuem o termo t_i e que pertencem a classe mais frequente; ϕ_{c_j,t_i} é a quantidade de documentos que possuem o termo t_i e que pertencem a classe c_j; e $\lambda_1, \lambda_2, \lambda_3$ são constantes que ajustam a velocidade de decaimento do fator de confidência. Os valores usados neste trabalho foram $\lambda_1 = 0{,}25$, $\lambda_2 = 10{,}0$, e $\lambda_3 = 8{,}0$, conforme proposto originalmente por [Assis et al. 2006].

Nas demais funções, $p(c_j|t_i)$ corresponde à probabilidade condicional da classe c_j, dada a presença do termo t_i; $p(\bar{t}_i|c_j)$ é a probabilidade condicional da ausência de t_i, dada a classe c_j; $p(t_i|\bar{c}_j)$ corresponde à probabilidade condicional do termo t_i, dadas as classes diferentes de c_j; $p(\bar{t}_i|\bar{c}_j)$ corresponde à probabilidade condicional da ausência do termo t_i, dadas as classes diferentes de c_j; e $|\mathcal{D}^\star|$ é a quantidade de documentos de treinamento.

A seguir, são apresentadas as faixas de valores que podem ser retornadas pelas funções apresentadas na Tabela 2.

- *CF, GI e IG*: retornam valores entre zero e um. Valores mais próximos de um indicam que o termo tem boa contribuição na identificação da classe do problema.
- *DFS*: retorna valores entre 0 e $0{,}5$. Valores mais próximos de $0{,}5$ significam que o termo contribui mais para a identificação da classe do que termos com valores mais próximos de 0.
- χ^2: tem valor próximo de zero quando o termo t_i aparece frequentemente em muitas classes e tem valor zero quando o termo t_i e a classe c_j são independentes. Por outro lado, quando a ocorrência do termo t_i e da classe c_j é altamente dependente,

o χ^2 tem valor alto;

- *NGL e GSS*: se o termo t_i aparece com frequência em documentos da classe c_j, o valor será positivo. Se o termo aparece com frequência para as outras classes, o valor será negativo.
- *OR*: gera valores entre zero e $+\infty$. O valor um é neutro e valores que tendem ao $+\infty$ indicam que o termo t_i aparece com frequência para a classe c_j, enquanto aparece com pouca frequência em documentos de outras classes. Por outro lado, valores próximos a zero indicam que o termo t_i aparece com pouca frequência em documentos da classe c_j, enquanto ocorre com frequência nas outras classes. Para que a OR não retorne valor que tende ao infinito, no cálculo das probabilidades, somou-se um ao número de documentos de treinamento de cada classe e à frequência de aparecimento do termo em cada classe.

As técnicas χ^2, NGL, GSS e OR não atribuem um valor global para cada termo. Elas atribuem um valor para o termo t_i relativo a cada classe. Diante disso, para essas técnicas foi calculada uma pontuação global usando a seguinte equação [Yang e Pedersen 1997, Joachims 1998]:

$$\mathcal{F}(t_i) = \max_{i=1}^{|\mathcal{C}|} \mathcal{F}_{\text{local}}(t_i, c_j). \tag{12}$$

Conforme mencionado anteriormente, o valor de $F(t_i)$ deve estar no intervalo $0 \leq F(t_i) \leq 1$. Para as técnicas que retornam uma pontuação para o termo t_i fora desse intervalo, foi realizada uma normalização. Portanto, o cálculo de $F(t_i)$ pode ser formalizado da seguinte maneira:

$$F(t_i) = \begin{cases} \mathcal{F}(t_i), & \text{se } 0 \leq \mathcal{F}(t_i) \leq 1 \\ \dfrac{\mathcal{F}(t_i) - min\mathcal{F}}{max\mathcal{F} - min\mathcal{F}}, & \text{caso contrário,} \end{cases} \tag{13}$$

onde $max\mathcal{F}$ e $min\mathcal{F}$ são, respectivamente, o maior e o menor valor que poderia ser retornado por $\mathcal{F}(t_i)$, dada a quantidade de exemplos de treinamento de cada classe.

Com o intuito de oferecer uma explicação didática das diferenças entre as diversas técnicas de atribuição de pontuação para os termos, a Tabela 3 ilustra a pontuação de 10 termos em um problema de classificação de 4 classes, cada uma com 100 documentos. As primeiras quatro colunas apresentam a frequência de cada termo ($t_i, \forall i \in \{1 \ldots 10\}$) nos documentos de cada classe ($c_j, \forall j \in \{1 \ldots 4\}$). É importante notar que as frequências foram atribuídas empiricamente.

Pode ser observado que todas as técnicas atribuem uma pontuação alta para termos que aparecem em muitos documentos de uma única classe (t_1). Além disso, as técnicas CF e OR favorecem termos que aparecem em poucos documentos que pertencem a uma única classe (t_6), enquanto as técnicas DFS, IG, χ^2 e GSS retornam uma baixa pontuação para o mesmo termo. A técnica NGL gera um valor intermediário nesta situação, se comparada às outras técnicas. Em resumo, enquanto as técnicas CF e OR favorecem a exclusividade de um termo em documentos de uma dada classe, as técnicas DFS, IG, χ^2 e GSS tentam balancear a exclusividade e a frequência de um termo em documentos que pertencem a mesma classe.

Tabela 3. Pontuação dos termos calculada pelas técnicas CF, DFS, IG, χ^2, OR, NGL e GSS.

	c_1	c_2	c_3	c_4	CF	DFS	GI	IG	χ^2	OR	NGL	GSS
t_1	100	0	0	0	0.926	1.000	1.000	0.811	1.000	1.000	1.000	1.000
t_2	100	100	0	0	0.617	0.500	0.500	1.000	0.333	0.362	0.577	0.667
t_3	100	100	100	0	0.309	0.200	0.333	0.811	1.000	0.182	0.333	0.333
t_4	100	100	100	100	0.000	0.000	0.250	0.000	0.000	0.000	0.000	0.000
t_5	50	50	50	50	0.000	0.000	0.062	0.000	0.000	0.126	0.000	0.000
t_6	10	0	0	0	0.554	0.053	0.010	0.051	0.077	0.587	0.277	0.100
t_7	1	0	0	0	0.006	0.005	0.000	0.005	0.008	0.299	0.087	0.010
t_8	100	50	20	10	0.019	0.305	0.328	0.445	0.407	0.409	0.638	0.733
t_9	100	50	20	1	0.231	0.359	0.364	0.534	0.446	0.433	0.668	0.763
t_10	80	50	20	10	0.010	0.179	0.185	0.243	0.222	0.361	0.471	0.533

Outro ponto a ser notado é que a técnica IG atribui a pontuação máxima para termos que aparecem com bastante frequência em metade das classes, enquanto aparecem com pouca frequência na outra metade (t_2). Já, os termos que aparecem com exclusividade para uma única classe, recebem uma pontuação inferior ao que é atribuído pelas outras técnicas (t_1). Além disso, a técnica IG e a χ^2 atribuem uma pontuação alta para termos que aparecem com frequência na maioria das classes, mas aparecem com pouca frequência em apenas uma classe (t_3).

4. Metodologia experimental

Os experimentos foram realizados com a base de dados pública e bem conhecida *SMS Spam Collection* [Almeida et al. 2011a]. Não foi encontrada nenhuma base de dados pública de spam via mensagens instantâneas. Porém, dado que SMS e mensagens instantâneas compartilham as mesmas características (são curtas e repletas de gírias, expressões idiomáticas, etc), estima-se que os métodos de classificação obtenham comportamentos semelhantes em ambos os cenários. Até mesmo um possível efeito da presença de *emoticons*, que são bastante usados em mensagens instantâneas e pouco usados em SMS, o que poderia fazer com que o desempenho dos métodos fossem diferentes nos dois cenários, pode ser anulado traduzindo-os para palavras que os representem.

4.1. Pré-processamento e *tokenização*

[Silva et al. 2016b] estudou o impacto da aplicação das técnicas de pré-processamento *stemming* e remoção de *stopwords* na classificação automática de SMS spam e SPIM. Foi demostrado experimentalmente que a remoção de *stopwords* prejudicou o desempenho dos métodos de classificação, enquanto que a técnica de *stemming* levemente melhorou os resultados. Uma hipótese formulada para explicar o baixo desempenho dos métodos de classificação quando essas técnicas foram empregadas foi o fato das mensagens serem curtas e repletas de ruídos, tais como gírias, expressões idiomáticas, símbolos, *emoticons* e abreviações. Para testar essa hipótese, neste trabalho, foi avaliada se a aplicação de técnicas como normalização léxica, indexação semântica e desambiguação podem trazer benefícios à tarefa de predição. Conjectura-se que o emprego delas pode melhorar o desempenho da classificação de mensagens instantâneas e SMS.

A normalização léxica é uma técnica de pré-processamento que faz a tradução de gírias, expressões idiomáticas e abreviações, frequentemente usadas pelos usuários da Internet, para a sua forma canônica. Por outro lado, a indexação semântica, também conhecida como geração de conceitos, retorna diferentes significados para um determinado termo e, portanto, tende a aumentar a quantidade de termos de uma amostra textual. Por fim, a desambiguação (*word sense disambiguation*) seleciona apenas os conceitos mais relevantes de acordo com o contexto da mensagem [Almeida et al. 2016]. A aplicação dessas três técnicas de pré-processamento foi feita por meio da ferramenta *online* `TextExpansion`[1], proposta por [Almeida et al. 2016].

Tanto as mensagens originais, quanto as geradas pela aplicação das técnicas de normalização léxica, indexação semântica e desambiguação foram convertidas para letras minúsculas. Além disso, na etapa de *tokenização*, os delimitadores utilizados foram quaisquer caracteres não alfanuméricos (exceto *underline*).

A base de dados utilizada nesse estudo possui 4.827 mensagens da classe spam e 747 mensagens da classe *ham*. A Tabela 4 apresenta outras informações importantes da base de dados: a quantidade de termos e a mediana e amplitude interquartil (IQR – *interquartile range*) do número de termos por mensagem.

Tabela 4. Informações da base de dados *SMS Spam Collection*.

	Quantidade de termos	Mediana	IQR
Base de dados original	8.706	11	13
Após normalização léxica	7.223	12	13
Após indexação semântica	14.610	42	54
Após desambiguação	7.236	12	14

Observa-se que a quantidade de termos por amostra (evidenciados pelas colunas $|d|$, mediana e IQR) é bastante elevada pela aplicação da geração de conceitos por indexação semântica. Por outro lado, a quantidade mediana de termos por amostra da base original é significativamente baixa (apenas 11), o que indica que dos 8.706 atributos, apenas 11 possuem algum valor (em mediana), evidenciando a alta esparsidade da representação vetorial de cada mensagem. Essa característica é preservada após a aplicação de normalização léxica e desambiguação. Isso significa que essas técnicas objetivam melhorar a qualidade dos atributos, enquanto que a indexação semântica visa aumentar a quantidade de termos por amostra.

4.2. Avaliação

O método `MDLText` foi avaliado em duas tarefas distintas de classificação: aprendizado *offline* (ou aprendizado em batelada) e aprendizado *online*.

No aprendizado *offline*, o mais comum na literatura, todos os exemplos de treinamento são apresentados ao método de classificação de uma única vez e este cria um modelo de predição global para classificar dados não conhecidos. Os modelos gerados no aprendizado *offline* geralmente são mais poderosos, porém são estáticos. Isso significa que uma vez que o treinamento é realizado, caso seja necessário atualizar o modelo

[1]A ferramenta `TextExpansion` está publicamente disponível em: http://lasid.sor.ufscar.br/expansion. Acessado em 30/11/2017.

de predição usando novos dados, é necessário refazer o treinamento com todos os dados usados anteriormente e com os novos dados.

No aprendizado *online*, é apresentado um exemplo por vez e o modelo de predição é atualizado de forma incremental. Portanto, métodos de aprendizado *online* são mais apropriados para problemas de larga escala, pois eles não precisam armazenar todos os exemplos na memória. Eles também são apropriados para problemas de classificação de spam, pois os modelos de predição são dinâmicos e adaptáveis às técnicas usadas pelos *spammers*, que estão em contínua evolução. Porém, em cenários onde métodos *offline* podem ser aplicados, eles geralmente obtém melhores resultados [Crammer et al. 2012].

4.2.1. Medidas de desempenho

Para comparar os resultados, foram empregadas as seguintes medidas de desempenho tradicionalmente usadas para avaliar e comparar filtros de spam:

- *acurácia*: taxa de acerto global do método de classificação, ou seja, a proporção de predições corretas em relação ao tamanho do conjunto de dados;
- *spam caught*: proporção de mensagens spam identificadas corretamente;
- *blocked ham*: proporção de mensagens *ham* identificadas incorretamente como spam;
- coeficiente de correlação de Matthews (MCC – *Matthews Correlation Coefficient*): medida amplamente utilizada em problemas de classificação de spam, que tem como principal característica levar em conta o desbalanceamento entre as classes. O MCC retorna um valor que varia entre -1 e +1. Um valor próximo de +1 indica uma boa predição. Por outro lado, um valor próximo de -1 indica que o classificador faz predições inversas. Por fim, um MCC igual a zero significa que o método de classificação tem desempenho equivalente a um preditor aleatório [Almeida et al. 2011b].

4.3. Métodos de aprendizado *offline*

O `MDLText` foi comparado aos seguintes métodos consolidados de aprendizado *offline*: *naïve* Bayes multinomial (M.NB) [McCallum e Nigam 1998], *naïve* Bayes Bernoulli (B.NB) [McCallum e Nigam 1998], Rocchio [Rocchio 1971], SVM [Cortes e Vapnik 1995], k-vizinhos mais próximos (KNN – *K-Nearest Neighbors*) [Cover e Hart 1967], árvores de decisão (DT – *Decision Trees*) [Breiman et al. 1984] e floresta aleatória (RF – *Random Forests*) [Breiman 2001]. Esses métodos são amplamente utilizados como *baseline* em trabalhos de classificação de texto.

Os métodos M.NB, B.NB, G.NB, Rocchio, KNN, DT e RF foram utilizados a partir da biblioteca `scikit-learn`[2], disponível em `Python`. O método `MDLText` foi implementado em `C++`. Já, os experimentos com o método SVM foram realizados usando a biblioteca `LibSVM`[3] para `C++`. Foi usada a função de *kernel* linear, que é mais apropriada para problemas de alta dimensionalidade [Hsu et al. 2003].

[2]A biblioteca `scikit-learn` está disponível em http://scikit-learn.org/. Acessado em: 30/11/2017.

[3]A biblioteca `LibSVM` está disponível em https://www.csie.ntu.edu.tw/~cjlin/libsvm/. Acessado em: 30/11/2017.

O desempenho de alguns métodos de categorização de texto pode ser bastante afetado pelo tipo de peso escolhido para representar os documentos. Portanto, para os métodos DT, M.NB, KNN, RF e SVM foi feita uma busca em grade usando validação cruzada *k-fold* estratificada com $k = 5$ para encontrar o melhor esquema de pesos (binário, TF ou TF-IDF). Para os métodos `MDLText` e Rocchio foi usada apenas a representação TF-IDF, enquanto para o método B.NB foi usada a representação textual binária, pois estes esquemas de representação são intrínsecos a estes métodos.

O desempenho dos métodos KNN, RF, SVM e `MDLText` também pode ser bastante afetado pela escolha dos seus parâmetros. Diante disso, foi realizada uma busca em grade para encontrar o melhor valor para o número de vizinhos (KNN), número de árvores (RF) e para os parâmetros $|\Omega|$ (`MDLText`) e C (SVM). Para os outros métodos avaliados, foram usados os parâmetros padrões de suas respectivas bibliotecas, com exceção do método Rocchio, onde em vez de usar a medida de distância padrão, que é a distância euclidiana, foi usada a distância de cosseno que é amplamente utilizada em problemas textuais.

4.4. Métodos de aprendizado *online*

O `MDLText` foi comparado com os seguintes métodos tradicionais de aprendizado *online*: M.NB [McCallum e Nigam 1998], B.NB [McCallum e Nigam 1998], Perceptron [Freund e Schapire 1999], gradiente descendente estocástico (SGD – *Stochastic Gradient Descent*) [Zhang 2004], algoritmo de margem larga aproximada (ALMA – *Approximate Large Margin Algorithm*) [Hoi et al. 2014, Gentile 2002], gradiente descendente *online* (OGD – *Online Gradient Descent*) [Zinkevich 2003] e algoritmo de margem máxima *online* relaxada (ROMMA – *Relaxed Online Maximum Margin Algorithm*) [Li e Long 2002].

Os métodos M.NB, B.NB foram utilizados a partir da biblioteca `scikit-learn` em Python. Os experimentos com os métodos Perceptron, ALMA, OGD, SGD e ROMMA foram realizados usando funções da biblioteca `LIBOL`[4] [Hoi et al. 2014] para `MATLAB`. Já, o `MDLText` foi implementado em `C++`. Uma busca em grade também foi usada para encontrar o melhor esquema de representação dos textos para os métodos M.NB, perceptron, ALMA, OGD, SGD e ROMMA. Para o método `MDLText` foi usado o esquema de representação TF-IDF em todos os experimentos e uma busca em grade foi usada para encontrar o melhor valor do parâmetro $|\Omega|$. Por fim, para o método B.NB, foi usado o esquema de representação binária.

5. Experimentos e resultados

Nesta seção, são apresentados os resultados obtidos nas tarefas de classificação *offline* e *online* das mensagens de SMS. Os métodos foram avaliados com as mensagens originais e com as mensagens processadas pela normalização léxica, indexação semântica e desambiguação.

5.1. Classificação *offline*

Antes de comparar o `MDLText` a outros métodos de aprendizado *offline*, foi analisado o seu desempenho usando diferentes técnicas para calcular a pontuação dos termos.

[4]A biblioteca `LIBOL` está disponível em: http://libol.stevenhoi.org/. Acessado em 30/11/2017.

5.1.1. Avaliação do MDLText usando diferentes técnicas de pontuação de termos

A Tabela 5 mostra os resultados obtidos pelo MDLText usando diferentes técnicas para calcular a pontuação dos termos. Os resultados foram obtidos usando validação cruzada 5-*fold* estratificada. Os métodos estão ordenados pelo MCC e os valores em negrito indicam o melhor resultado por medida de desempenho. Ainda, os valores são apresentados usando tons de cinza, onde quando melhor o valor para uma dada medida de desempenho, mais escura é a célula que o contém.

Tabela 5. Resultados obtidos pelo MDLText usando diferentes técnicas para calcular a pontuação de termos em um cenário de classificação *offline*.

Método	Acurácia	*Blocked ham*	*Spam caught*	MCC
Base de dados original				
MDLText (GI)	**0.983**	0.010	**0.938**	**0.927**
MDLText (GSS)	**0.983**	0.010	**0.938**	**0.927**
MDLText (CF)	**0.983**	**0.009**	0.929	0.925
MDLText (χ^2)	0.982	0.011	0.937	0.923
MDLText (OR)	0.982	0.011	0.933	0.922
MDLText (IG)	0.980	0.013	0.937	0.918
MDLText (NGL)	0.980	0.012	0.933	0.916
MDLText (DFS)	0.980	0.012	0.932	0.916
Normalização léxica				
MDLText (CF)	**0.982**	**0.011**	0.939	**0.924**
MDLText (OR)	**0.982**	0.012	0.938	0.922
MDLText (χ^2)	**0.982**	**0.011**	0.940	0.922
MDLText (DFS)	0.981	0.012	0.941	0.921
MDLText (GI)	0.981	0.013	0.940	0.919
MDLText (IG)	0.981	0.014	**0.944**	0.919
MDLText (NGL)	0.981	0.013	0.941	0.918
MDLText (GSS)	0.980	0.014	0.938	0.913
Indexação semântica				
MDLText (CF)	**0.955**	**0.035**	0.890	**0.817**
MDLText (DFS)	0.948	0.047	**0.911**	0.798
MDLText (GI)	0.947	0.047	0.906	0.795
MDLText (IG)	0.946	0.048	0.909	0.794
MDLText (OR)	0.946	0.046	0.900	0.792
MDLText (NGL)	0.944	0.050	0.906	0.787
MDLText (GSS)	0.944	0.049	0.898	0.784
MDLText (χ^2)	0.944	0.049	0.898	0.784
Desambiguação				
MDLText (OR)	**0.981**	**0.012**	0.936	**0.919**
MDLText (GI)	0.980	0.014	0.940	0.913
MDLText (χ^2)	0.979	0.015	0.941	0.913
MDLText (DFS)	0.979	0.015	0.941	0.912
MDLText (NGL)	0.979	0.016	**0.942**	0.911
MDLText (IG)	0.979	0.015	0.936	0.910
MDLText (CF)	0.979	0.014	0.932	0.910
MDLText (GSS)	0.978	0.016	**0.942**	0.909

O MDLText, em geral, obteve melhores resultados quando a técnica CF foi utilizada. Nos experimentos com a base de dados original e com a base de dados processada pela normalização léxica e indexação semântica, o MDLText (CF) obteve a melhor acu-

rácia e taxa de *blocked ham*. Porém, a diferença entre os valores do MCC obtidos pelas diferentes técnicas foi pequena, na maioria das avaliações. A diferença média entre o melhor MCC e o pior foi apenas 0,016.

Uma das prováveis causas para a pouca diferença entre as diversas técnicas usadas para calcular a pontuação dos termos é o baixo número de termos por mensagem. Conforme mostra a Tabela 4, a quantidade mediana de termos por amostra da base de dados original é onze, o que é um número muito baixo para ser utilizado pelas técnicas de pontuação de forma satisfatória. Além disso, mesmo que a forma de calcular os pesos dos termos varie para cada técnica, é possível observar na Tabela 2, que geralmente os termos que possuem uma grande pontuação em uma técnica, tendem a obter boa pontuação nas outras.

Para facilitar a comparação dos resultados e verificar se alguma das técnicas avaliadas foi significativamente melhor que as demais, foi realizada uma análise estatística usando o teste não paramétrico de Friedman, seguindo cuidadosamente a metodologia descrita em [Demšar 2006]. O uso do teste de Friedman também é recomendado por [García et al. 2009], [García et al. 2010], [Japkowicz e Shah 2011] e [Santafe et al. 2015]. A Figura 2 apresenta o *ranking* médio obtido pelo `MDLText` para cada uma das técnicas de pontuação de termos que foi avaliada, onde *rankings* com médias mais baixas indicam melhor desempenho.

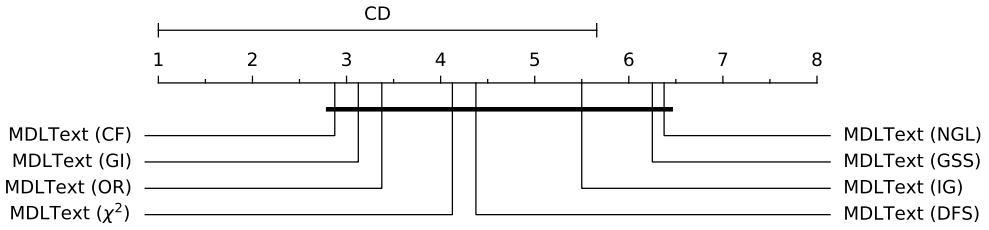


Figura 2. Média dos *rankings* do `MDLText` usando diferentes técnicas de pontuação de termos em um cenário de classificação *offline* e diferença crítica calculada usando o teste *post-hoc* de Bonferroni-Dunn. O eixo de cada gráfico apresenta os *rankings* médios, sendo que quanto menor o valor, melhor o desempenho. Como todas as técnicas estão conectadas, isso significa que não existe evidência estatística para afirmar que alguma delas é superior às demais.

Seja k o número de modelos ou grupos avaliados, q o número de observações e R_j a média dos *rankings* do j-ésimo modelo ou grupo avaliado. O teste de Friedman verifica se a hipótese nula, que afirma que todos os métodos possuem desempenhos equivalentes, pode ser rejeitada. Para isso, é empregada a seguinte equação:

$$\chi_F^2 = \frac{12 \times q}{k \times (k+1)} \times \sum_{j=1}^{k} \left(R_j^2 - \frac{k+1}{2} \right)^2. \tag{14}$$

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 [Demšar 2006]. Para um intervalo de confiança $\alpha = 0{,}05$, $k = 8$ e $q = 4$, o valor crítico é $2{,}170$. Portanto, como $\chi_F^2 = 9{,}021$, a hipótese nula pode ser rejeitada.

Uma vez que a hipótese nula pôde ser rejeitada, foi empregado o teste *post-hoc* para a comparação par-a-par entre os métodos. Para isso, foi usado o teste de Bonferroni-Dunn, conforme sugerido por [Demšar 2006], [García et al. 2009], [García et al. 2010], [Japkowicz e Shah 2011] e [Santafe et al. 2015]. Este teste estatístico afirma que os desempenhos de dois métodos são significativamente divergentes se a diferença entre as médias dos *rankings* deles for maior ou igual a uma diferença crítica calculada pela seguinte equação [García et al. 2009, Demšar 2006]:

$$CD = q_\alpha \times \sqrt{\frac{k \times (k+1)}{6 \times q}}, \tag{15}$$

onde q_α corresponde ao valor crítico apresentado em Demšar [Demšar 2006, pág. 12]. Para um intervalo de confiança $\alpha = 0{,}05$, a diferença crítica foi igual a $4{,}659$. Portanto, pode-se afirmar que as técnicas de pontuação dos termos foram estatisticamente equivalentes, isto é, não existe evidência estatística para afirmar que alguma das técnicas foi significantemente superior às demais. Apesar disso, como o `MDLText` obteve o menor *ranking* médio quando foi usada a técnica CF, ela foi utilizada em todos os experimentos posteriores.

5.1.2. Comparação do `MDLText` com outros métodos

A Tabela 6 apresenta o desempenho obtido por cada método avaliado, sendo que os resultados foram calculados usando a validação cruzada 5-*fold* estratificada. Os resultados estão ordenados pelo MCC e os valores em negrito indicam o melhor valor por medida de desempenho. Ainda, os valores são apresentados usando tons de cinza onde quanto mais escura é a célula da tabela, melhor é o desempenho.

Conforme mostrado na Tabela 6, o SVM obteve o melhor MCC em todos os experimentos. Porém, a taxa de *spam caught* ficou bem abaixo da melhor taxa obtida para cada teste executado. O `MDLText` também ofereceu resultados promissores, pois obteve a segunda melhor acurácia, taxa de *spam caught* e MCC nas avaliações que usaram a base de dados original e quando foi aplicada normalização léxica e desambiguação.

O método RF obteve as melhores taxas de *blocked ham* em todos os testes. Porém, a taxa de *spam caught* foi uma das três piores em todos os experimentos. Isso significa que ele se especializou em classificar as mensagens *ham*, ao custo de cometer muitos erros na classificação de spam. Uma taxa baixa de *spam caught* é um problema, pois deixar de filtrar muitas mensagens spam é indesejável, uma vez que essas mensagens podem causar desperdício de tempo aos usuários e ou até carregar conteúdos prejudiciais, como *links* maliciosos que podem ser usados para a prática de crimes cibernéticos.

A normalização léxica foi a melhor das técnicas de pré-processamento avaliadas, uma vez que melhorou os resultados de quase todos os métodos. Isso indica que várias mensagens de texto são compostas por gírias, expressões idiomáticas e abreviações, que ao terem seus termos normalizados, facilitaram o processo de classificação dos métodos. Porém, de maneira geral, nenhuma das técnicas melhorou significativamente os resultados.

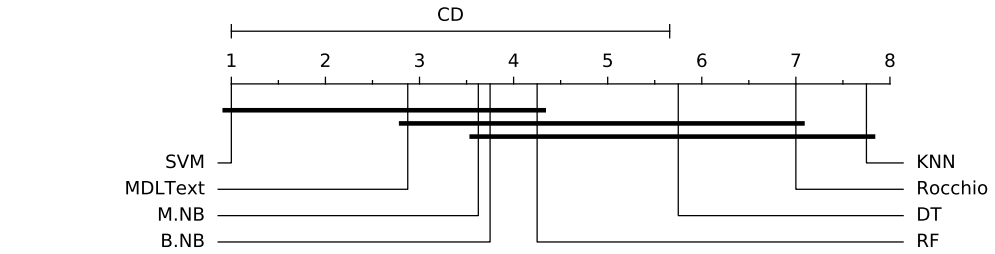
Tabela 6. Resultados obtidos por cada método em um cenário de classificação *offline*.

Método	Acurácia	*Blocked ham*	*Spam caught*	MCC
Base de dados original				
SVM	**0.984**	0.002	0.891	**0.930**
MDLText	0.983	0.009	0.929	0.925
M.NB	0.981	0.012	0.936	0.919
B.NB	0.982	0.002	0.877	0.917
RF	0.978	**0.000**	0.841	0.904
DT	0.969	0.015	0.870	0.867
Rocchio	0.963	0.034	**0.946**	0.856
KNN	0.952	0.001	0.648	0.781
Normalização léxica				
SVM	**0.984**	0.002	0.894	**0.931**
MDLText	0.982	0.011	**0.939**	0.924
M.NB	0.981	0.010	0.930	0.920
B.NB	0.980	0.004	0.880	0.911
RF	0.979	**0.000**	0.840	0.905
Rocchio	0.965	0.031	0.938	0.860
DT	0.963	0.018	0.841	0.839
KNN	0.962	**0.000**	0.722	0.830
Indexação semântica				
SVM	**0.983**	0.005	**0.913**	**0.928**
RF	0.976	**0.002**	0.835	0.893
B.NB	0.966	0.020	0.882	0.857
DT	0.967	0.019	0.873	0.856
MDLText	0.955	0.035	0.890	0.817
M.NB	0.958	0.006	0.727	0.810
KNN	0.943	0.004	0.605	0.735
Rocchio	0.918	0.080	0.903	0.716
Desambiguação				
SVM	**0.984**	0.001	0.890	**0.931**
M.NB	0.979	0.012	0.920	0.910
MDLText	0.979	0.014	0.932	0.910
B.NB	0.979	0.004	0.872	0.909
RF	0.977	**0.000**	0.830	0.899
DT	0.965	0.019	0.855	0.846
Rocchio	0.961	0.035	**0.933**	0.845
KNN	0.909	**0.000**	0.324	0.540

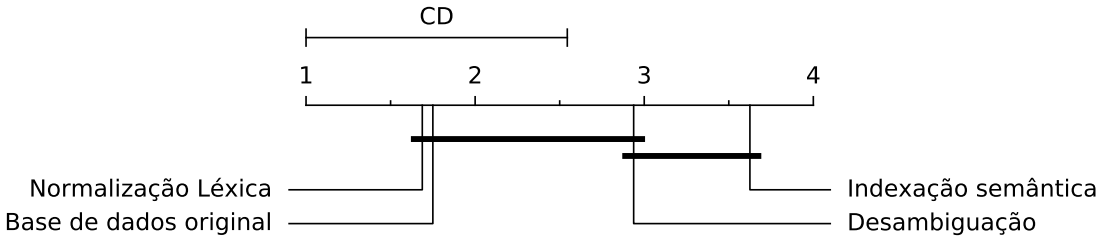
Para verificar se o resultado obtido por algum dos métodos avaliados foi significativamente superior aos demais, foi realizada uma análise estatística usando o teste de Friedman. A Figura 3(a) mostra o *ranking* médio dos métodos baseado no MCC.

Para um intervalo de confiança $\alpha = 0{,}05$, o teste de Friedman indicou que a hipótese nula de igualdade entre os métodos pode ser seguramente descartada. Portanto, foi realizada uma comparação par-a-par usando o teste *post-hoc* de Bonferroni–Dunn.

Para um intervalo de confiança $\alpha = 0{,}05$, a diferença crítica calculada pelo teste *post-hoc* de Bonferroni–Dunn foi igual a $4{,}659$. Portanto, o desempenho do MDLText foi significativamente superior ao desempenho do KNN. Porém, não há evidência estatística suficiente para afirmar que o desempenho geral do MDLText foi inferior ao SVM ou superior aos demais métodos.



(a)



(b)

Figura 3. *Rankings* médios obtidos na classificação *offline* e diferenças críticas calculadas usando o teste *post-hoc* de Bonferroni–Dunn. O eixo de cada gráfico apresenta os *rankings* médios, sendo que quanto menor o valor, melhor o desempenho. Os métodos que não estão conectados são estatisticamente diferentes. A Figura 3(a) mostra o *ranking* médio para cada método. A Figura 3(b) mostra o *ranking* médio obtido nos experimentos usando as mensagens de texto originais e as processadas pela normalização léxica, indexação semântica e desambiguação.

Analisando o impacto das técnicas de pré-processamento, pode-se observar que as taxas de acurácia e MCC do `MDLText` foram prejudicadas pela aplicação dessas técnicas. O pior resultado relativo a essas medidas de desempenho foi obtido com a aplicação da indexação semântica. Por outro lado, a taxa de *spam caught* do `MDLText` obteve o melhor resultado com a aplicação da normalização léxica.

Diferentemente do que ocorreu com o `MDLText`, o MCC dos métodos SVM, M.NB, RF, Rocchio e KNN foi melhor nos testes em que alguma das técnica de pré-processamento foi aplicada. A técnica que mais beneficiou a maioria dos métodos foi a normalização léxica.

A Figura 3(b) apresenta o *ranking* médio obtido nos experimentos usando as mensagens de texto originais e as processadas pela normalização léxica, indexação semântica e desambiguação.

De acordo com o método não-paramétrico de Friedman, para um intervalo de confiança $\alpha = 0{,}05$, a hipótese nula de igualdade entre os resultados obtidos usando as bases originais e as técnicas de pré-processamento pode ser descartada. De acordo com o teste

post-hoc de Bonferroni–Dunn, para um intervalo de confiança $\alpha = 0{,}05$, a diferença crítica foi $1{,}545$. Portanto, apesar do *ranking* médio obtido pela aplicação da normalização léxica ter sido o melhor, não existe evidência estatística suficiente para afirmar que a aplicação dessa técnica melhora significativamente os resultados. Porém, a análise estatística confirma que o emprego da técnica de normalização léxica produz resultados estatisticamente superiores aos obtidos após a aplicação de indexação semântica.

5.2. Classificação *online*

Neste experimento, foi considerado o seguinte cenário: (1) a ordem dos dados é modificada aleatoriamente; (2) um pequeno número de mensagens de texto (20% da base de dados) é usado para formar o conjunto de treinamento, enquanto as mensagens restantes são usadas para teste; (3) as mensagens do conjunto de treinamento são usadas na busca em grade realizada para definir o melhor esquema de pesos para os métodos M.NB, Perceptron, ALMA, OGD, SGD e ROMMA e o melhor valor do parâmetro $|\Omega|$ para o `MDLText`; (4) depois, elas são disponibilizadas para o treinamento inicial dos métodos; (5) posteriormente, uma mensagem por vez do conjunto de teste é apresentada ao classificador, que emite a predição; (6) em seguida, o classificador recebe um *feedback* e, caso o erro seja maior que zero, o modelo de predição é atualizado com a classe correta.

É importante mencionar que o esquema de representação de pesos usado no início de cada experimento é o TF. Logo, se esse esquema for escolhido na busca em grade, não é feito nenhum tipo de conversão de pesos. Se o esquema de representação de pesos escolhido na busca em grade for o TF-IDF, o processo de conversão de TF para TF-IDF é aplicado usando apenas as informações sobre os documentos apresentados previamente no processo de treinamento. Por outro lado, caso o esquema de representação escolhido seja o binário, o processo de conversão de TF para binário usa apenas as informações sobre o próprio documento que será convertido.

A configuração dos experimentos descrita acima é comum em aplicações reais que envolvem classificação de spam, como nos serviços de email. Por exemplo, quando um usuário recebe um email indesejado na caixa de entrada, ele pode clicar no botão "Spam" para enviar o email para a caixa de spam e avisar ao serviço de email que um erro foi cometido. Por outro lado, se um email importante para o usuário for enviado equivocadamente para a caixa de spam, ele pode clicar no botão "Não é spam" para enviar esse email para o local correto e avisar ao serviço de email que ele fez uma classificação incorreta.

Em aplicações que envolvem SMS ou mensagens instantâneas, algo semelhante ao que é feito nos serviços de email poderia ser implementado. Por exemplo, uma opção para reportar spam poderia ser adicionada em um determinado aplicativo de mensagens. Além disso, poderia ser criado um local para onde são enviadas todas as mensagens classificadas como spam, dando ao usuário a chance de visualizar e resgatar mensagens consideradas spam de maneira equivocada pelo filtro usado pelo aplicativo. Portanto, o método de classificação usado pelo aplicativo, previamente treinado com algum conjunto de mensagens rotuladas, faria uma predição para cada mensagem recebida pelo usuário que, por sua vez, poderia fornecer um *feedback* caso não concorde com a classe escolhida.

Antes de comparar o `MDLText` com outros métodos, foi analisado o seu desempenho usando as diferentes técnicas para o cálculo da pontuação dos termos.

5.2.1. Avaliação do `MDLText` usando diferentes técnicas de pontuação de termos

A Tabela 7 mostra os resultados obtidos pelo `MDLText` usando diferentes técnicas para o cálculo da pontuação dos termos. Os valores apresentados na tabela são as médias de cada medida de desempenho obtidas em dez repetições do processo de classificação *online*. Os valores são apresentados usando tons de cinza onde quanto mais escura é a célula da tabela, melhor é o desempenho. Ainda, os métodos estão ordenados pelo MCC e o melhor resultado por medida de desempenho é apresentado em negrito.

Tabela 7. Resultados obtidos pelo `MDLText` usando diferentes técnicas para calcular a pontuação de termos em um cenário de classificação *online*.

Método	Acurácia	*Blocked ham*	*Spam caught*	MCC
Base de dados original				
`MDLText` (OR)	**0.976**	**0.018**	0.934	**0.899**
`MDLText` (GSS)	0.975	**0.018**	0.933	0.896
`MDLText` (GI)	0.975	0.019	**0.936**	0.894
`MDLText` (NGL)	0.975	0.019	0.935	0.894
`MDLText` (χ^2)	0.974	0.019	0.932	0.893
`MDLText` (DFS)	0.974	0.020	**0.936**	0.892
`MDLText` (CF)	0.974	**0.018**	0.922	0.891
`MDLText` (IG)	0.973	0.021	0.935	0.890
Normalização léxica				
`MDLText` (OR)	**0.976**	**0.017**	0.931	**0.897**
`MDLText` (IG)	**0.976**	0.018	0.933	**0.897**
`MDLText` (GSS)	0.975	0.019	0.936	0.894
`MDLText` (χ^2)	0.975	0.019	0.935	0.894
`MDLText` (DFS)	0.974	0.020	0.937	0.893
`MDLText` (CF)	0.974	0.019	0.932	0.892
`MDLText` (GI)	0.974	0.020	0.935	0.891
`MDLText` (NGL)	0.973	0.021	**0.940**	0.890
Indexação semântica				
`MDLText` (CF)	**0.951**	**0.040**	0.898	**0.807**
`MDLText` (GSS)	0.950	0.043	**0.907**	0.805
`MDLText` (χ^2)	0.950	0.042	0.904	0.805
`MDLText` (DFS)	0.949	0.043	0.900	0.799
`MDLText` (GI)	0.948	0.045	0.901	0.797
`MDLText` (OR)	0.948	0.046	0.906	0.797
`MDLText` (IG)	0.947	0.046	0.906	0.797
`MDLText` (NGL)	0.947	0.046	0.904	0.795
Desambiguação				
`MDLText` (GSS)	**0.972**	**0.021**	0.929	**0.884**
`MDLText` (GI)	**0.972**	0.022	0.930	0.883
`MDLText` (NGL)	**0.972**	**0.021**	0.927	0.882
`MDLText` (χ^2)	**0.972**	0.022	0.933	0.882
`MDLText` (IG)	0.971	0.024	**0.937**	0.881
`MDLText` (DFS)	0.971	0.022	0.929	0.880
`MDLText` (CF)	0.971	0.023	0.932	0.880
`MDLText` (OR)	0.971	0.023	0.930	0.879

A Tabela 7 mostra que, nos experimentos com a base de dados original e com a aplicação da normalização léxica, o `MDLText` obteve os melhores resultados quando a técnica OR foi usada para calcular a pontuação dos termos. Porém, essa técnica obteve

desempenho inferior à maioria das outras técnicas avaliadas nos experimentos com aplicação da indexação semântica e desambiguação. Por outro lado, a técnica GSS obteve o melhor desempenho nos experimentos com aplicação da desambiguação e, ainda, obteve um dos três melhores desempenhos nas demais avaliações.

Outro ponto importante é que o desvio padrão entre os resultados obtidos pelo `MDLText` usando as diferentes técnicas foi pequeno. Portanto, a alteração das técnicas de pontuação a serem empregadas na função de penalização do `MDLText` não causa nenhuma diferença significativa nos resultados da classificação *online* de SMS spam e SPIM. É provável que as diferenças de desempenho entre as técnicas seriam mais expressivas em problemas com uma grande quantidade de termos, uma vez que o maior desvio padrão entre os resultados, tanto no cenário *online*, quanto no *offline*, foi obtido com a aplicação da indexação semântica, que aumenta a mediana de termos por mensagem.

Para verificar se alguma das técnicas avaliadas foi significativamente melhor que as demais, foi realizada uma análise estatística usando o teste não paramétrico de Friedman, considerando os *rankings* médios ilustrados na Figura 4.

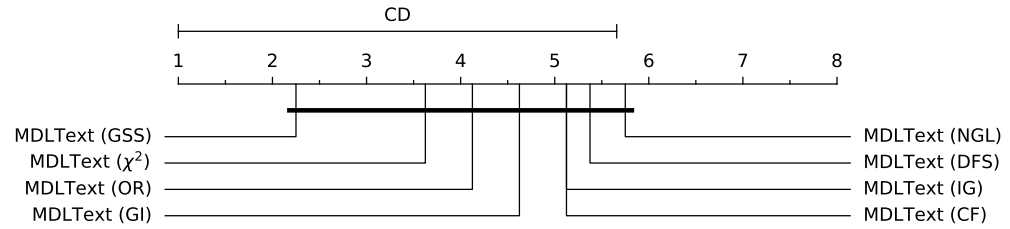


Figura 4. Média dos *rankings* do `MDLText` usando diferentes técnicas de pontuação de termos em um cenário de classificação *online* e diferença crítica calculada usando o teste *post-hoc* de Bonferroni–Dunn. O eixo do gráfico apresenta os *rankings* médios, sendo que quanto menor o valor, melhor o desempenho. Como todas as técnicas estão conectadas, isso expressa que não existe diferença significativa de desempenho entre elas.

Para um intervalo de confiança $\alpha = 0{,}05$, $k = 8$ e $q = 4$, o valor crítico calculado pelo teste de Friedman foi $2{,}170$. Portanto, como $\chi^2_F = 6{,}062$, a hipótese nula de igualdade entre as técnicas pode ser rejeitada. Diante disso, foi feita uma comparação par-a-par usando o teste *post-hoc* de Bonferroni-Dunn. Para um intervalo de confiança $\alpha = 0{,}05$, a diferença crítica foi igual a $4{,}659$. Portanto, as técnicas de pontuação dos termos foram consideradas estatisticamente equivalentes. Apesar disso, como o `MDLText` obteve o menor *ranking* médio quando foi usada a técnica GSS, ela foi utilizada em todos os experimentos posteriores.

5.2.2. Comparação do `MDLText` com outros métodos

A Tabela 8 apresenta as médias dos resultados obtidos por cada método em dez rodadas da classificação *online*. Os métodos estão ordenados pelo MCC e os valores em negrito indicam o melhor resultado por medida de desempenho. Ainda, são usados tons de cinza

para apresentar os resultados, onde quanto mais escura é a célula da tabela, melhor é o desempenho.

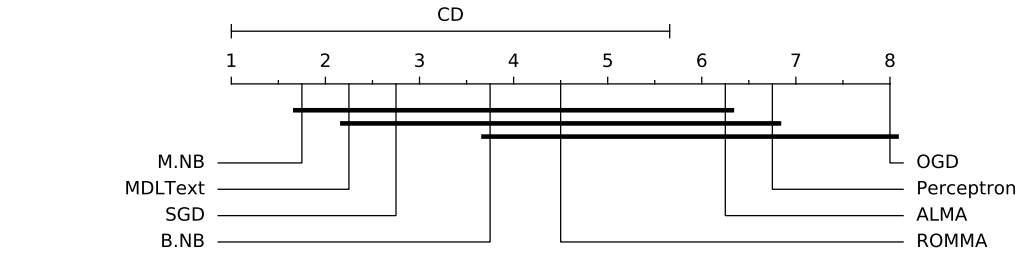
Tabela 8. Resultados obtidos por cada método em um cenário de classificação *online*.

Método	Acurácia	*Blocked ham*	*Spam caught*	MCC
Base de dados original				
M.NB	**0.979**	0.007	0.888	**0.907**
MDLText	0.975	0.018	**0.933**	0.896
SGD	0.959	0.030	0.891	0.833
ROMMA	0.955	0.033	0.881	0.816
B.NB	0.953	**0.001**	0.651	0.784
ALMA	0.922	0.062	0.819	0.703
Perceptron	0.921	0.020	0.540	0.623
OGD	0.894	0.007	0.255	0.426
Normalização léxica				
M.NB	**0.978**	0.006	0.875	**0.901**
MDLText	0.975	0.019	**0.936**	0.894
SGD	0.966	0.023	0.896	0.859
ROMMA	0.956	0.033	0.882	0.818
B.NB	0.959	**0.001**	0.698	0.813
ALMA	0.923	0.060	0.817	0.703
Perceptron	0.912	0.014	0.431	0.559
OGD	0.884	0.005	0.166	0.346
Indexação semântica				
B.NB	**0.965**	**0.007**	0.789	**0.844**
SGD	0.961	0.024	0.865	0.833
MDLText	0.950	0.043	**0.907**	0.805
M.NB	0.941	0.052	0.893	0.773
ROMMA	0.929	0.053	0.812	0.716
Perceptron	0.924	0.018	0.551	0.638
ALMA	0.876	0.098	0.704	0.543
OGD	0.899	0.013	0.335	0.477
Desambiguação				
M.NB	**0.976**	0.007	0.865	**0.893**
MDLText	0.972	0.021	**0.929**	0.884
SGD	0.961	0.028	0.885	0.837
B.NB	0.956	**0.001**	0.674	0.798
ROMMA	0.946	0.041	0.861	0.780
ALMA	0.926	0.055	0.804	0.714
Perceptron	0.901	0.025	0.418	0.504
OGD	0.877	0.011	0.156	0.283

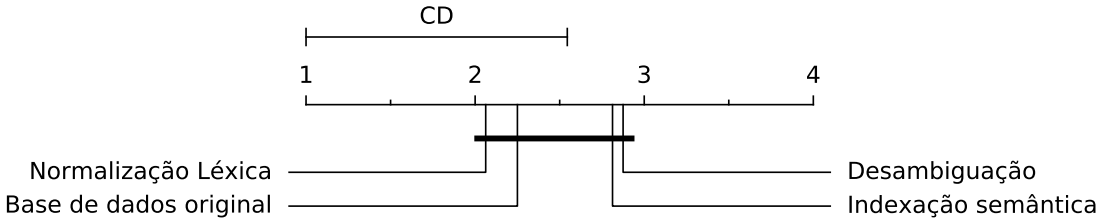
Os resultados indicam que o MDLText também obteve excelente desempenho na detecção *online* de spam em mensagens de texto. Ele obteve a melhor taxa de *spam caught* e um dos três melhores valores de acurácia e MCC em todos os experimentos. O M.NB obteve a melhor acurácia e MCC na maioria dos experimentos, porém suas taxas de *spam caught*, em geral, foram bastante inferiores aos valores obtidos pelo MDLText.

Para certificar que os resultados não foram obtidos por acaso, foi realizada uma análise estatística usando o teste de Friedman. A Figura 5(a) mostra o *ranking* médio dos métodos baseado no MCC.

Para um intervalo de confiança $\alpha = 0{,}05$, o teste de Friedman indicou que a



(a)



(b)

Figura 5. *Rankings* médios obtidos na classificação *online* e diferenças críticas calculadas usando o teste *post-hoc* de Bonferroni–Dunn. O eixo de cada gráfico apresenta os *rankings* médios, sendo que quanto menor o valor, melhor o desempenho. Os métodos que não estão conectados são estatisticamente diferentes. A Figura 5(a) mostra o *ranking* médio para cada método. A Figura 5(b) mostra o *ranking* médio obtido nos experimentos usando as mensagens de texto originais e as processadas pela normalização léxica, indexação semântica e desambiguação.

hipótese nula de igualdade entre os métodos pode ser seguramente descartada. Portanto, foi realizada uma comparação par-a-par usando o teste *post-hoc* de Bonferroni–Dunn. A diferença crítica, para um intervalo de confiança $\alpha = 0{,}05$, foi igual a $4{,}659$. Portanto, há evidência estatística suficiente para afirmar que o desempenho geral do `MDLText` foi significativamente melhor que o desempenho geral do OGD. Porém, o `MDLText` foi considerado estatisticamente equivalente aos demais métodos, apesar de ter obtido um *ranking* médio melhor do que a maioria.

Assim como no cenário *offline*, o MCC obtido pelo `MDLText` foi melhor nos experimentos sem a aplicação de nenhuma técnica de pré-processamento, enquanto que a melhor taxa de *spam caught* foi obtida nos testes com aplicação da normalização léxica. O melhor MCC dos métodos SGD, ROMMA e B.NB foi obtido nos experimentos com normalização léxica. Por outro lado, o melhor MCC dos métodos OGD e Perceptron foi obtido com a aplicação da indexação semântica. A desambiguação foi a melhor técnica de pré-processamento apenas para o método ALMA, considerando o MCC. Já, o M.NB obteve o melhor MCC quando nenhuma técnica de pré-processamento foi aplicada.

A Figura 5(b) apresenta o ranking médio obtido nos experimentos usando as men-

sagens de texto originais e as processadas pela normalização léxica, indexação semântica e desambiguação.

De acordo com o método não-paramétrico de Friedman, para um intervalo de confiança $\alpha = 0{,}05$, a hipótese nula de igualdade entre os resultados obtidos usando a bases de dados original e as técnicas de pré-processamento pode ser descartada. De acordo com a comparação par-a-par usando o teste *post-hoc* de Bonferroni–Dunn, para um intervalo de confiança $\alpha = 0{,}05$, a diferença crítica entre os métodos foi $1{,}545$. Portanto, pode-se afirmar que o desempenho obtido usando as mensagens originais e as processadas foram estatisticamente equivalentes.

6. Conclusões e trabalhos futuros

A filtragem de spam em SMS e mensagens instantâneas é um problema desafiador. Esse tipo de mensagem contém características que tendem a impactar negativamente o desempenho até mesmo de métodos de classificação que obtiveram sucesso em outros problemas de detecção de spam: elas são curtas e contém ruídos, como gírias e expressões idiomáticas. Para contribuir com a solução desse problema, este trabalho propôs a aplicação do `MDLText`, um método de classificação de texto que realiza seleção de modelo com base no princípio MDL. Este método possui uma função de penalidade que faz com que os termos mais relevantes exerçam maior influência na predição e ao mesmo tempo não menospreza as informações carregadas por termos menos relevantes.

[Silva et al. 2016a] mostraram que o `MDLText`, usando a técnica DFS para calcular a pontuação dos termos, é efetivo na detecção de SMS Spam e SPIM. Baseado nisso, neste artigo considerou-se a hipótese de que o desempenho do `MDLText` pode ser ainda melhor se outras técnicas de pontuação forem utilizadas. Outra hipótese considerada é que o tratamento das mensagens para remoção de ruído e enriquecimento do texto pode aumentar o desempenho do `MDLText` e de outros métodos de classificação.

Diante disso, foram avaliadas as técnicas CF, DFS, IG, χ^2, NGL, GSS e OR para serem empregadas na função de penalização de termos do `MDLText`. No cenário de classificação *offline*, em geral, o `MDLText` obteve melhores desempenhos nos experimentos em que a técnica CF foi usada para retornar a pontuação global dos termos. Por outro lado, no cenário de classificação *online*, os melhores resultados do `MDLText` foram obtidos nos testes em que a técnica GSS foi usada. Contudo, em ambos os cenários, as diferenças de desempenho foram insignificantes e, portanto, a mudança de escolha de qualquer técnica avaliada tende a não alterar significativamente o resultado geral do `MDLText`. Provavelmente, o baixo número de termos por mensagem tenha dificultado a atuação das técnicas de pontuação e, portanto, recomenda-se o estudo do impacto das diferentes técnicas em outros problemas de categorização de texto onde a mediana de termos por mensagem seja maior.

O `MDLText` também foi comparado a outros métodos considerados *baseline* em aprendizado *online* e *offline* e amplamente empregados na classificação de spam. No cenário de aprendizado *offline*, apesar da análise estatística indicar equivalência entre os métodos comparados, o `MDLText` obteve o segundo melhor *ranking* médio, superando as técnicas M.NB, B.NB, RF, DT, Rocchio e KNN. Ele foi inferior apenas ao SVM, porém ele possui importantes vantagens: (1) é naturalmente adaptado para problemas multiclasses, (2) seu aprendizado pode ocorrer de forma incremental e (3) possui complexidade

linear, tanto no estágio de treinamento, quanto na predição. No cenário de aprendizado *online*, o `MDLText` também obteve o segundo melhor *ranking* médio superando as técnicas SGD, B.NB, ROMMA, ALMA, Perceptron e OGD. Contudo, a análise estatística indicou haver equivalência entre os métodos comparados.

Neste trabalho, também foi avaliado o impacto das seguintes técnicas de pré-processamento de texto na filtragem de SMS spam e SPIM: normalização léxica, indexação semântica e desambiguação. Foi observado que a normalização léxica foi a única que melhorou o desempenho da maioria dos métodos avaliados em ambos os cenários. Isso indica que as mensagens de texto contêm ruídos, mas eles foram reduzidos pela normalização dos termos, o que aumentou a qualidade dos modelos de predição.

Em trabalhos futuros, pretende-se propor e avaliar novas técnicas de atribuição de pontuação para os atributos que superem a técnica CF e GSS. Além disso, prevê-se o estudo de outras técnicas de pré-processamento que possam contornar as principais características das amostras de texto curtas e ruidosas e, consequentemente, beneficiar a filtragem automática de SMS spam e SPIM.

Além das mensagens instantâneas e SMS, os *spammers* têm atacado várias outras mídias e plataformas eletrônicas, como blogs, redes sociais, redes de compartilhamento de vídeos, sites de busca, sites de notícias, fóruns de discussão e aplicativos de *smartphones*. Diante disso, em trabalhos futuros, pretende-se analisar o desempenho do `MDLText` na filtragem desses outros tipos de spam. Pretende-se também fazer a combinação do `MDLText` com outros métodos que não sejam especializados em classificação de texto, pois acredita-se que a eficácia na detecção de spam possa aumentar quando são explorados tanto os atributos textuais, quanto os não-textuais.

Agradecimentos

Os autores são gratos à CAPES (Proc. 1709642) e CNPq (Proc. 141089/2013-0) pelo apoio financeiro concedido ao desenvolvimento desse projeto.

Referências

Abdulhamid, S. M., Latiff, M. S. A., Chiroma, H., Osho, O., Abdul-Salaam, G., Abubakar, A. I., e Herawan, T. (2017). A review on mobile SMS spam filtering techniques. *IEEE Access*, 5:15650–15666. doi: 10.1109/ACCESS.2017.2666785 [GS Search]

Ahmed, I., Ali, R., Guan, D., Lee, Y.-K., Lee, S., e Chung, T. (2015). Semi-supervised learning using frequent itemset and ensemble learning for SMS classification. *Expert Systems with Applications*, 42(3):1065–1073. doi: 10.1016/j.eswa.2014.08.054 [GS Search]

Almeida, T. A., Hidalgo, J. M. G., e Yamakami, A. (2011a). Contributions to the study of SMS spam filtering: new collection and results. In *Proceedings of the 11th ACM Symposium on Document engineering (DocEng'11)*, pages 259–262, Mountain View, CA, USA. ACM. doi: 10.1145/2034691.2034742 [GS Search]

Almeida, T. A., Silva, T. P., Santos, I., e Hidalgo, J. M. G. (2016). Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering. *Knowledge-Based Systems*, 108:25–32. doi: 10.1016/j.knosys.2016.05.001 [GS Search]

Almeida, T. A., Yamakami, A., e Almeida, J. (2011b). Spam filtering: how the dimensionality reduction affects the accuracy of naive Bayes classifiers. *Journal of Internet Services and Applications*, 1(3):183–200. doi: 10.1007/s13174-010-0014-7 [GS Search]

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

Bi, J., Wu, J., e Zhang, W. (2008). A trust and reputation based anti-SPIM method. In *Proceedings of the 27th IEEE Conference on Computer Communications (INFOCOM'08)*, pages 1–5, Phoenix, Arizona, USA. IEEE Computer Society. doi: 10.1109/INFOCOM.2008.319 [GS Search]

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 1th edition. [GS Search]

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32. doi: 10.1023/A:1010933404324 [GS Search]

Breiman, L., Friedman, J. H., Olshen, R. A., e Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group, Belmont, California, USA. [GS Search]

Carpenter, L. M. e Hubbard, G. B. (2014). Cyberbullying: Implications for the psychiatric nurse practitioner. *Journal of Child and Adolescent Psychiatric Nursing*, 27(3):142–148. doi: 10.1111/jcap.12079 [GS Search]

Cortes, C. e Vapnik, V. N. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. doi: 10.1007/BF00994018 [GS Search]

Cover, T. M. e Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transaction on Information Theory*, 13(1):21–27. doi: 10.1109/TIT.1967.1053964 [GS Search]

Crammer, K., Dredze, M., e Pereira, F. (2012). Confidence-weighted linear classification for text categorization. *Journal of Machine Learning Research*, 13(1):1891–1926. [GS Search]

Das, S., Pourzandi, M., e Debbabi, M. (2012). On SPIM detection in LTE networks. In *Proceedings of the 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'2012)*, pages 1–4, Montreal, Québec, Canada. IEEE. doi: 10.1109/CCECE.2012.6334959 [GS Search]

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30. [GS Search]

Domingos, P. (1999). The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3:409–425. doi: 10.1023/A:1009868929893 [GS Search]

Freund, Y. e Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296. doi: 10.1023/A:1007662407062 [GS Search]

Galavotti, L., Sebastiani, F., e Simi, M. (2000). Experiments on the use of feature selection and negative evidence in automated text categorization. In *Proceedings of the 4th*

European Conference on Research and Advanced Technology for Digital Libraries (ECDL'00), Lisbon, Portugal. Springer. doi: 10.1007/3-540-45268-0_6 [GS Search]

García, S., Fernández, A., Luengo, J., e Herrera, F. (2009). A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Soft Computing*, 13(10):959–977. doi: 10.1007/s00500-008-0392-y [GS Search]

García, S., Fernández, A., Luengo, J., e Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064. doi: 10.1016/j.ins.2009.12.010 [GS Search]

Gentile, C. (2002). A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242. [GS Search]

Gomez-Martin, L. E. (2012). Smartphone usage and the need for consumer privacy laws. *Pittsburgh Journal of Technology Law and Policy*, 12:217–237. doi: 10.5195/tlp.2012.96 [GS Search]

Goswami, G., Singh, R., e Vatsa, M. (2016). Automated spam detection in short text messages. In Singh, R., Vatsa, M., Majumdar, A., e Kumar, A., editors, *Machine Intelligence and Signal Processing*, volume 390, pages 85–98. Springer India, New Delhi. doi: 10.1007/978-81-322-2625-3_8 [GS Search]

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

Hastie, T. J., Tibshirani, R. J., e Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer, New York, NY, USA, 2th edition. [GS Search]

Hoi, S. C. H., Wang, J., e Zhao, P. (2014). Libol: A library for online learning algorithms. *Journal of Machine Learning Research*, 15(1):495–499. [GS Search]

Hsu, C., Chang, C., e Lin, C. (2003). A practical guide to support vector classification. Technical report, National Taiwan University. [GS Search]

Japkowicz, N. e Shah, M. (2011). *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, New York, NY, USA. [GS Search]

Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*, pages 137–142, Chemnitz, Germany. Springer. doi: 10.1007/BFb0026683 [GS Search]

Li, Y. e Long, P. M. (2002). The relaxed online maximum margin algorithm. *Machine Learning*, 46(1-3):361–387. doi: 10.1023/A:1012435301888 [GS Search]

Liu, Z., Lin, W., Li, N., e Lee, D. (2005). Detecting and filtering instant messaging spam: A global and personalized approach. In *Proceedings of the First International Conference on Secure Network Protocols (NPSEC'05)*, pages 19–24. IEEE Computer Society. doi: 10.1109/NPSEC.2005.1532048 [GS Search]

Manning, C. D., Raghavan, P., e Schütze, H. (2009). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA. [GS Search]

McCallum, A. e Nigam, K. (1998). A comparison of event models for naive Bayes text classification. In *Proceedings of the 15th AAAI Workshop on Learning for Text Categorization (AAAI'98)*, pages 41–48, Madison, Wisconsin. [GS Search]

Ng, H. T., Goh, W. B., e Low, K. L. (1997). 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)*, pages 67–73, Philadelphia, PA, USA. ACM. doi: 10.1145/258525.258537 [GS Search]

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471. doi: 10.1016/0005-1098(78)90005-5 [GS Search]

Rissanen, J. (1996). Fisher information and stochastic complexity. *IEEE Transaction on Information Theory*, 42(1):40–47. doi: 10.1109/18.481776 [GS Search]

Rocchio, J. J. (1971). Relevance feedback in information retrieval. In Salton, G., editor, *The Smart retrieval system - experiments in automatic document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ. [GS Search]

Santafe, G., Inza, I. n., e Lozano, J. A. (2015). Dealing with the evaluation of supervised classification algorithms. *Artificial Intelligence Review*, 44(4):467–508. doi: 10.1007/s10462-015-9433-y [GS Search]

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47. doi: 10.1145/505282.505283 [GS Search]

Shang, W., Huang, H., Zhu, H., Lin, Y., Qu, Y., e Wang, Z. (2007). A novel feature selection algorithm for text categorization. *Expert Systems with Applications*, 33(1):1–5. doi: 10.1016/j.eswa.2006.04.001 [GS Search]

Silva, R. M., Alberto, T. C., Almeida, T. A., e Yamakami, A. (2016a). Filtrando comentários do YouTube através de classificação online baseada no princípio MDL e indexação semântica. In *Anais do 13th Encontro Nacional de Inteligência Artificial e Computacional (ENIAC'16)*, pages 2–15, Recife, PE, Brasil. [GS Search]

Silva, R. M., Almeida, T. A., e Yamakami, A. (2015). Quanto mais simples, melhor! Categorização de textos baseada na navalha de Occam. In *Anais do 12th Encontro Nacional de Inteligência Artificial e Computacional (ENIAC'15)*, pages 2–15, Natal, RN, Brasil. [GS Search]

Silva, R. M., Almeida, T. A., e Yamakami, A. (2016b). Detecção automática de SPIM e SMS spam usando método baseado no princípio da descrição mais simples. In *Anais do 13th Encontro Nacional de Inteligência Artificial e Computacional (ENIAC'16)*, pages 2–15, Recife, PE, Brasil. [GS Search]

Silva, R. M., Almeida, T. A., e Yamakami, A. (2017). MDLText: An efficient and lightweight text classifier. *Knowledge-Based Systems*, 118:152–164. doi: 10.1016/j.knosys.2016.11.018 [GS Search]

Tsakalidis, G. e Vergidis, K. (2017). A systematic approach toward description and classification of cybercrime incidents. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP(99):1–20. doi: 10.1109/TSMC.2017.2700495 [GS Search]

Uysal, A. K. e Gunal, S. (2012). A novel probabilistic feature selection method for text classification. *Knowledge-Based Systems*, 36:226–235. doi: 10.1016/j.knosys.2012.06.005 [GS Search]

Uysal, A. K., Gunal, S., Ergin, S., e Gunal, E. S. (2012). A novel framework for SMS spam filtering. In *Proceedings of the 2012 International Symposium on Innovations in Intelligent Systems and Applications (INISTA'12)*, pages 1–4, Trabzon, Turkey. IEEE. doi: 10.1109/INISTA.2012.6246947 [GS Search]

Wilbur, W. J. e Kim, W. (2009). The ineffectiveness of within-document term frequency in text classification. *Information Retrieval*, 12(5):509–525. doi: 10.1007/s10791-008-9069-5 [GS Search]

Yang, Y. e Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pages 412–420, Nashville, TN, USA. Morgan Kaufmann. [GS Search]

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21th International Conference on Machine Learning (ICML'04)*, pages 116–123, Banff, Alberta, Canada. ACM. doi: 10.1145/1015330.1015332 [GS Search]

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pages 928–936, Washington, DC, USA. AAAI Press. [GS Search]