`MDLText` e Indexação Semântica aplicados na Detecção de *Spam* nos Comentários do YouTube

Renato Moraes Silva[1], **Túlio C. Alberto**[2], **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,tuliocasagrande@acm.org
talmeida@ufscar.br

Abstract. *Several YouTube users regularly produce video content and make this task their main livelihood activity. However, such success is drawing the attention of malicious users which propagate undesired comments, looking for self-promotion or disseminating malicious links. The available text categorization methods commonly used to address this problem suffer from the following inherent characteristics: (1) the comments are usually short and poorly written and (2) the classification problem is naturally online. In this paper, we evaluate a classification method based on the minimum description length principle and compare its results with those of well-established online learning techniques. We also propose an ensemble approach which combines the classification methods with different natural language processing techniques. The performed experiments were carefully carried out and statistical analysis of the results indicates that the proposed technique was superior than when only the original comments were employed.*

Resumo. *Muitos usuários do YouTube produzem conteúdo regularmente e fazem desta tarefa seu principal meio de vida. Contudo, esse sucesso vem despertando a atenção de usuários mal-intencionados, que propagam comentários indesejados para se autopromoverem ou para disseminar links maliciosos. Neste cenário, métodos tradicionais de categorização de texto podem sofrer limitações devido às características inerentes ao problema: (1) os comentários costumam ser curtos e mal redigidos e (2) o problema de classificação é naturalmente online. Este artigo avalia um método de classificação baseado no princípio da descrição mais simples e compara os resultados com os de métodos tradicionais de aprendizado online. Também é proposta uma técnica ensemble, que combina os métodos de classificação com diferentes técnicas de processamento de linguagem natural. Os experimentos foram cuidadosamente realizados e a análise estatística dos resultados indica que a técnica proposta obteve desempenho superior ao obtido quando apenas os comentários originais foram empregados.*

1. Introdução

Desde a primeira vez que a palavra *spam* foi empregada para descrever uma mensagem não-solicitada enviada em massa, esta prática se alastrou para diversos meios de

comunicação eletrônica, como email, blogs, SMS, mídias sociais, entre outros. O *spam* está associado ao conteúdo indesejado, de baixa qualidade e que atrapalha a visualização da informação que realmente interessa ao usuário, podendo ser apresentado na forma de imagem, texto ou vídeo.

O *spam* muitas vezes está associado a práticas criminosas, tais como roubo de propriedade intelectual e de informações pessoais, vírus e *malwares*, exposição à fraudes, pornografia e propaganda enganosa [Goodman et al. 2005]. Além do custo gerado para as vítimas destes crimes, existem também consequências indiretas, como a utilização dos recursos da infraestrutura da rede para trafegar as mensagens e o tempo gasto pelo usuário para selecionar e filtrar as mensagens indesejadas [Bratko et al. 2006].

Segundo dados da Cisco, empresa que fornece soluções computacionais, cerca de 86% do volume diário de emails é *spam*, podendo ultrapassar a quantidade de 500 bilhões de mensagens *spam* por dia[1]. Os principais assuntos são notificações falsas de pagamentos e depósitos bancários, compras de produtos *online*, anexos maliciosos, sites de relacionamento, entre outros[2]. A Figura 1 ilustra o volume diário de emails *spam* enviados no ano de 2016.

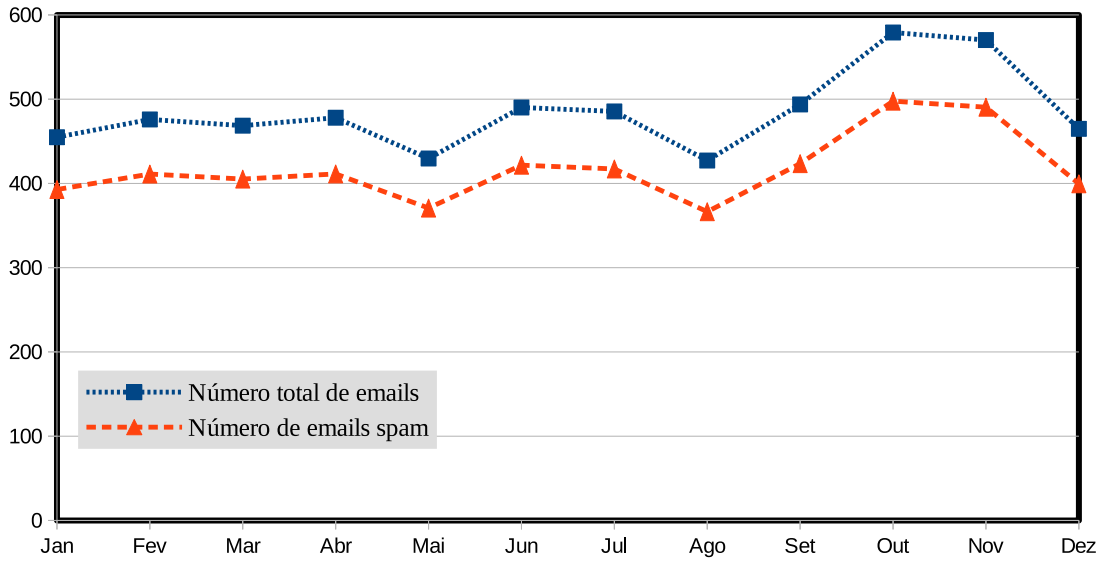


Figura 1. Volume diário (em bilhões) de emails *spam* enviados no ano de 2016 (Fonte: SenderBase, Cisco (adaptado)).

Desde o seu surgimento, diversas abordagens foram propostas para combater o *spam*, desde instrumentos jurídicos para punir os *spammers*, indivíduos que propagam *spam*, até sistemas de identificação e de reputação, em que os remetentes podem se certificar como usuários legítimos [Goodman et al. 2005]. No entanto, o método mais promissor consiste em aplicar filtros baseados no conteúdo da mensagem, capazes de discernir *spam* de mensagens legítimas automaticamente [Bratko et al. 2006]. Algoritmos de

[1] *Spam Overview*. Disponível em `https://www.senderbase.org/static/spam`, acessado em 20/03/2017.

[2] *Cisco 2014 Annual Security Report*. Disponível em `https://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf`, acessado em 20/03/2017.

aprendizado de máquina são particularmente eficazes nesta tarefa, pois são capazes de se ajustarem à base de treinamento e aprenderem as características das mensagens.

A filtragem de *spam*, no entanto, apresenta desafios adicionais para a categorização automática de texto, visto que os *spammers* constantemente manipulam as mensagens e tentam evadir a detecção. Este fenômeno é conhecido como classificação adversária e está presente em diversos outros cenários, tais como detecção de intrusão, detecção de fraude, contra-terrorismo, vigilância aérea ou compartilhamento ilegal de arquivos [Dalvi et al. 2004].

A existência de um adversário ativo faz com que métodos tradicionais de aprendizado de máquina, que não possuem aprendizado incremental e geram modelos estáticos de predição, possam não ser apropriados para serem aplicados em filtros reais de *spam* [Dalvi et al. 2004, Bratko et al. 2006]. Como o *spam* evolui continuamente e a maioria das aplicações são baseadas no retorno *online* do usuário, a tarefa demanda algoritmos de classificação rápidos, robustos e incrementais [Bratko et al. 2006].

Com o surgimento e popularização de outros meios de comunicação, o *spam* rapidamente se propagou para diversas aplicações. O YouTube, uma plataforma de compartilhamento de vídeos que também possui recursos de redes sociais, também passou a ser alvo dos *spammers*. Os usuários do YouTube podem publicar e compartilhar conteúdo, interagir com outros usuários por meio da seção de comentários e ainda opinar sobre os vídeos, por meio dos botões *like* (gostei) e *dislike* (não gostei). O sucesso da plataforma pode ser mensurado por estatísticas divulgadas pela própria empresa: a rede possui mais de 1 bilhão de usuários em mais de 88 países. Também está disponível em 76 idiomas, cobrindo 95% da população da Internet[3]. Além disso, segundo o *ranking* Alexa, o YouTube é o segundo site mais acessado do mundo[4].

Recentemente, o YouTube adotou um sistema de monetização que recompensa os usuários de acordo com as visualizações em seus vídeos, o que incentivou a produção de conteúdo original, mas também abriu portas para a propagação de informações indesejadas, geralmente constituídas por mensagens de baixa qualidade. Dentre os diversos tipos de conteúdo indesejado, destacam-se os comentários de usuários se autopromovendo ou disseminando *links* maliciosos que podem propagar vírus e *malwares*. A Figura 2 ilustra três comentários *spam* encontrados no YouTube. Enquanto o primeiro é de um usuário pedindo inscrições em seu próprio canal, o segundo possui um *link* que pode ser perigoso para outros usuários. O terceiro corresponde a outra prática bastante comum, com conteúdo não relacionado ao vídeo original.

Um dos trabalhos que abordaram o problema de *spam* em comentários do YouTube foi [O'Callaghan et al. 2012]. Os autores aplicaram redes funcionais (*network motifs*) para identificar campanhas de *spam* no YouTube. O trabalho apresentou boas taxas de acerto, porém limita-se a identificar mensagens muito semelhantes propagadas por *bots*. Os autores também publicaram uma base de dados com comentários extraídos do YouTube, rotulados como *spam* ou comentários legítimos. Embora a base possua mais

[3] *YouTube Statistics*. Disponível em `https://www.youtube.com/yt/press/statistics.html`, acessado em 09/03/2017.

[4] *Top 500 Global Sites*. Disponível em `http://www.alexa.com/topsites`, acessado em 09/03/2017.

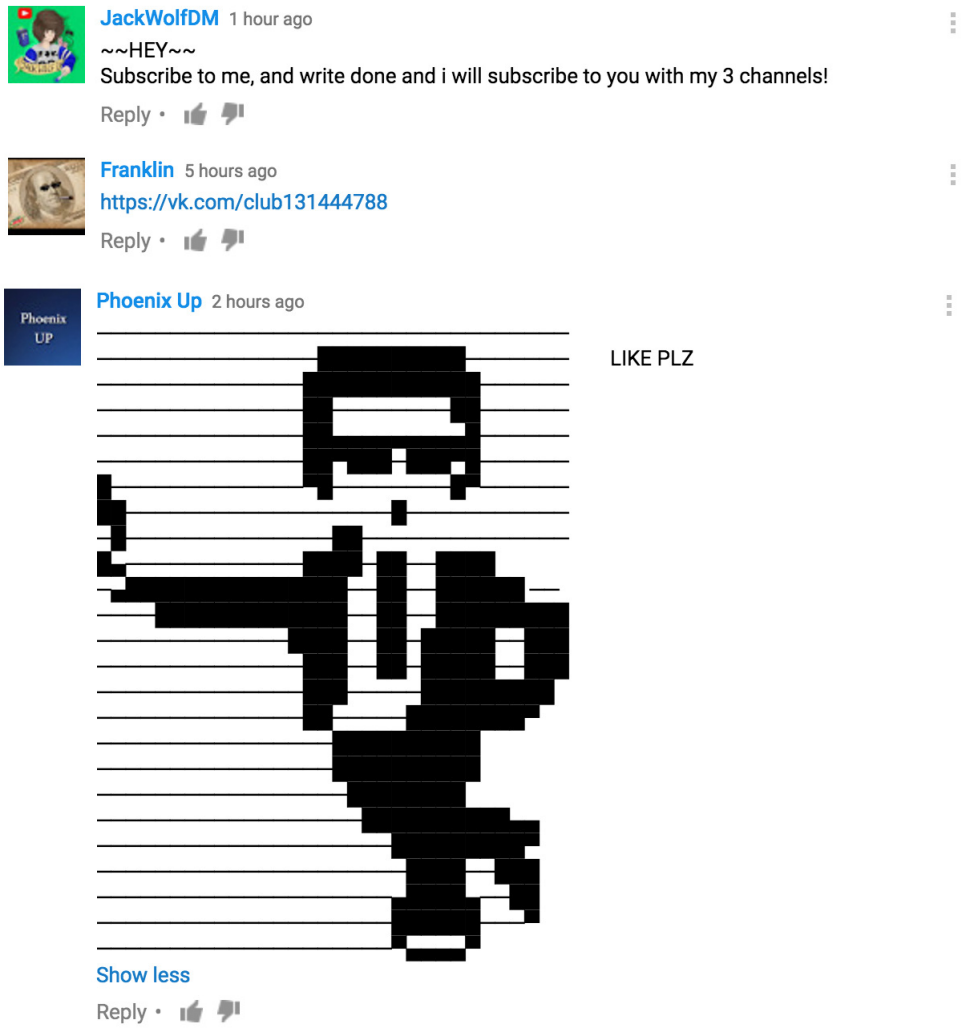


Figura 2. Exemplo de comentários *spam* comumente disseminados no YouTube (Fonte: YouTube).

de seis milhões de amostras, os rótulos foram atribuídos de acordo com o recurso *"reportar spam"* presente no YouTube e pode conter anotações incorretas, em que comentários legítimos estão marcados como *spam* e vice-e-versa.

[Rădulescu et al. 2014] avaliaram uma estratégia de detecção de *spam* nos comentários do YouTube baseada na identificação de textos desconexos, linguagem vulgar e inadequada, ou conteúdo não relacionado ao vídeo original. A abordagem não obteve resultados promissores e os autores atribuíram isso ao fato de que as mensagens na plataforma normalmente são curtas e ruidosas.

[Alberto et al. 2015] disponibilizaram cinco bases de dados de comentários extraídos do YouTube e compararam o desempenho de diversos métodos tradicionais de aprendizado de máquina. Segundo os autores, as características das mensagens, que possuem poucos termos e muito ruído, dificultam a aplicação direta dos métodos de classificação.

O *spam* encontrado em comentários do YouTube e outras mídias sociais, também

conhecido como *social spam*, difere-se muito do *spam* em emails, pois as mensagens geralmente possuem uma pequena quantidade de termos e são repletas de erros de digitação, gírias e abreviações, podendo dificultar a tarefa de classificação [Ammari et al. 2012, Eisenstein 2013]. Segundo [Baldwin et al. 2013], os comentários presentes no YouTube possuem em média 16 palavras e 31,9% delas falharam em ser interpretadas por um analisador gramatical. Os autores recomendam aplicar técnicas de processamento de linguagem natural para diminuir o ruído do texto.

Para contribuir com a solução desse problema, em [Silva et al. 2016a] foi analisado o desempenho do `MDLText`, um método de classificação de texto baseado no princípio da descrição mais simples (MDL – *Minimum Description Length*). Este método é eficiente, adaptado para problemas binários e multiclasse e, ainda, é dinâmico e escalável, uma vez que suporta aprendizado incremental, podendo ser aplicado em problemas de grande escala e em aplicações reais e *online* de classificação de texto. Segundo os autores, esse método obteve resultados promissores na filtragem *online* de *spam* em comentários do YouTube. Outros resultados preliminares relativos ao `MDLText` foram publicados em [Silva et al. 2015, Silva et al. 2016b, Silva et al. 2017].

[Silva et al. 2016a] também analisaram se a aplicação das técnicas de normalização léxica, indexação semântica e desambiguação pode melhorar os resultados da classificação de *spam* em comentários do YouTube. Os autores estudaram todas as possíveis combinações dessas técnicas e verificaram que, em geral, os métodos de classificação tiveram melhor desempenho quando alguma delas foi aplicada.

Dadas as boas taxas de acerto reportadas por [Silva et al. 2016a], neste artigo, o estudo da filtragem de *spam* em comentários do YouTube foi ampliado. Para isso, foi proposta e avaliada uma técnica *ensemble* que combina as predições obtidas pelos métodos de classificação usando os documentos de texto originais e versões expandidas pela aplicação de normalização léxica, indexação semântica e desambiguação. Foi analisado se os resultados do `MDLText` e de outros métodos tradicionais de aprendizado *online* podem ser melhorados quando eles são associados a essa técnica *ensemble*.

O restante deste trabalho está estruturado da seguinte forma: os conceitos básicos sobre o princípio MDL são apresentados na Seção 2. Na Seção 3, é explicado o método de classificação de texto avaliado neste artigo. As técnicas de normalização léxica e indexação semântica são apresentadas na Seção 4. Na Seção 5, é proposta a técnica *ensemble* que combina as predições obtidas por diferentes regras de expansão. Na Seção 6, são apresentadas as configurações adotadas nos experimentos. Os resultados obtidos são apresentados na Seção 7. Na Seção 8, são apresentadas as conclusões e direções para trabalhos futuros.

2. Princípio da descrição mais simples (MDL)

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 [Rissanen 1978, Rissanen 1983]. Isso significa que modelos menos complexos são preferíveis [Barron et al. 1998, Grünwald et al. 2005]. A simplicidade no princípio MDL é interpretada como o tamanho de descrição do modelo quando ele é usado para representar o dado.

O conceito por trás do princípio MDL, de escolher modelos menos com-

plexos, pode ser visto como uma formalização do princípio da navalha de Occam [Grünwald et al. 1998]. Essa ideia também está associada à complexidade de Kolmogorov que, em termos gerais, é o tamanho do menor programa que descreve um objeto representado por uma sequência binária [Kolmogorov 1965, Solomonoff 1964, Chaitin 1969]. Uma sequência binária que possui bastante regularidade tem uma complexidade de Kolmogorov menor do que uma sequência com menos regularidade, pois pode ser descrita por um programa menor. O menor programa que consegue descrever uma sequência binária é o que mais detectou regularidades nela, o que no contexto de aprendizado pode ser entendido como o programa que possui o maior poder preditivo sobre ela.

A Figura 3 apresenta duas sequências binárias. A primeira claramente possui mais regularidade que a segunda. Ela é formada por k repetições de 0001, enquanto a segunda é aleatória e, portanto, não possui regularidade. Diante disso, é possível criar um programa que imprime a sequência A de forma compacta, com um tamanho de descrição menor do que a sequência original, como no exemplo ilustrado pela Figura 4. Por outro lado, como a sequência B não possui regularidade, qualquer programa usado para descrevê-la terá aproximadamente o mesmo tamanho de descrição dela, pois terá que imprimi-la usando uma representação literal, como no exemplo ilustrado pela Figura 5.

```
A. 00010001000100010001...000100010001000100010001000100010001
B. 01110100110100100110...10101110101110110001011000100
```

Figura 3. Sequências binárias (Fonte: [Grünwald 2007, pág. 4]).

```
for i in range(k): print('0001',end = '')
```

Figura 4. Programa em Python que imprime a sequência binária A da Figura 3.

```
print('01110100110100100110...10101110101110110001011000100')
```

Figura 5. Programa em Python que imprime a sequência binária B da Figura 3.

É importante mencionar que não é possível determinar o tamanho de todos os programas que podem descrever todos os conjuntos de sequências binárias e, portanto, a complexidade de Kolmogorov é incomputável [Li and Vitányi 2008].

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 balanceamento entre a complexidade do modelo e a qualidade do seu ajuste aos dados de observação. Tais características são bastante desejáveis 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 $M_1, M_2, \ldots, M_k$, o MDL seleciona aquele, tal que:

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

sendo que $L(M)$ corresponde ao tamanho da descrição do modelo M, representando sua complexidade, e $L(Y|M)$ ao tamanho da descrição do dado Y quando codificado com M, representando quão bem M descreve Y. [Rissanen 1978] mostrou que $L(Y|M)$ pode ser calculado através do código de Shannon-Fano [Shannon 1948] e, portanto, $L(Y|M) = -\log p(Y|M)$, onde $p(Y|M)$ é a probabilidade condicional de Y dado M.

O MDL definido na Eq. 1 foi proposto por [Rissanen 1978] e [Rissanen 1983] e é conhecido como *crude* MDL. De acordo com [Grünwald 2005], 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. [Rissanen 1996] contornou o problema do *crude* MDL propondo uma nova versão do MDL que usa códigos universais para medir o tamanho da descrição de cada modelo. Basicamente, nessa nova versão, é empregada apenas uma parte do código com tamanho $\bar{L}(Y|M)$, ou seja, o modelo e os dados são codificados conjuntamente.

Não existe nenhum procedimento padrão ou universal para calcular o tamanho da descrição de um modelo. Porém, na próxima seção é apresentado o método `MDLText` [Silva et al. 2017], que oferece uma nova interpretação sobre como calcular o tamanho da descrição do modelo (ou classe) em problemas de classificação de texto.

3. O método de classificação `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 Y. Além disso, d recebe o rótulo j correspondente à classe c_j que possui o menor tamanho de descrição em relação à d:

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

No `MDLText`, o tamanho da descrição de d em relação à classe c_j é calculado multiplicando-se três termos: o tamanho da descrição dos termos de d, uma penalidade $K(t_i)$ para cada termo e uma penalidade $\hat{S}(d,c_j)$ para a classe c_j, como descrito na seguinte equação:

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

onde $|d|$ corresponde à quantidade de termos no documento d.

Tamanho da descrição dos termos de d

Na Equação 3, $L(t_i|c_j)$ é o tamanho de descrição do termo t_i dada a classe c_j e que pode ser calculado por:

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

O fator $\beta(t_i|c_j)$ é o peso condicional de t_i dada a classe c_j, baseado nos pesos TF-IDF [Wilbur and 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, $n_{c_j,t_i} = 0$. Neste trabalho, empiricamente foi adotado $|\Omega| = 2^{10}$.

Penalidade $K(t_i)$ para cada termo

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

$$K(t_i) = \frac{1}{(1 + \alpha) - F(t_i)}, \tag{8}$$

onde $0 \leq F(t_i) \leq 1$ corresponde à relevância (ou importância) 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 contribuição do termo entre zero e um. Essa pontuação é baseada na frequência do termo para cada classe. Se ele aparece em vários documentos de uma determinada classe (por exemplo, $j = 1$), mas em poucos documentos de outra classe ($j > 1$), a sua pontuação deve ser próxima de um. Se, por outro lado, ele aparece igualmente para todas as classes, a sua pontuação deve ser mais próxima de zero.

Neste artigo, para calcular $F(t_i)$, foi empregada a técnica de fatores de confidência (CF – *Confidence Factors*) proposta por [Assis et al. 2006]. Ela atribui uma pontuação de relevância para o termo t_i, usando a seguinte equação:

$$F(t_i) = \frac{1}{|\mathcal{C}| - 1} \times \sum_{\forall j | j \neq \tau} \frac{\left(\frac{\left(\phi_{c_\tau,t_i} - \phi_{c_j,t_i}\right)^2 + \left(\phi_{c_\tau,t_i} \times \phi_{c_j,t_i}\right) - \frac{\lambda_1}{\phi_{c_\tau,t_i} + \phi_{c_j,t_i}}}{\left(\phi_{c_\tau,t_i} + \phi_{c_j,t_i}\right)^2}\right)^{\lambda_2}}{1 + \left(\frac{\lambda_3}{\phi_{c_\tau,t_i} + \phi_{c_j,t_i}}\right)}, \tag{9}$$

onde 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 à classe mais frequente; ϕ_{c_j,t_i} é a quantidade de documentos que possuem o termo t_i e que pertencem à classe c_j; e λ_1,λ_2,λ_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].

Penalidade $\hat{S}(d,c_j)$ para a classe c_j

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:

$$\hat{S}(d,c_j) = -\log_2\left(\frac{1}{2} \times S(d,\overline{c}_j)\right). \tag{10}$$

O fator $S(d,\overline{c}_j)$ mede a similaridade entre o documento d e um vetor protótipo $\overline{c}_j$ de uma maneira parecida ao que é feito no algoritmo Rocchio [Rocchio 1971, Manning et al. 2009] e varia entre zero e um, sendo que 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:

$$S(d,\overline{c}_j) = \frac{\sum_{i=1}^{|d|} \hat{w}(t_i,d|c_j) \times \overline{c}_j(t_i)}{\|\hat{w}(:,d)\|_2 \times \|\overline{c}_j\|_2}, \tag{11}$$

onde $\hat{w}(:,d)$ é a soma dos pesos TF-IDF normalizados dos termos presentes em d e o vetor protótipo $\overline{c}_j$ é formado pela média dos pesos dos termos. Portanto, para cada termo t_i, $\overline{c}_j(t_i)$ é:

$$\overline{c}_j(t_i) = \frac{n_{c_j,t_i}}{|\hat{\mathcal{D}}_{c_j}|}, \tag{12}$$

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

Os Algoritmos 1 e 2, respectivamente, sumarizam os estágios de treinamento e teste do `MDLText`. Em ambos, considera-se que os documentos de entrada são representados por modelos espaço-vetoriais usando pesos TF-IDF.

O estágio de treinamento do `MDLText` (Algoritmo 1) consiste apenas na coleta de informações dos documentos de treinamento, referentes aos pesos TF-IDF dos termos e da frequência de vezes em que eles aparecem em cada classe. Tais informações são guardadas nas variáveis $\mathcal{T}$, n, $\hat{n}$, ϕ e $|\hat{\mathcal{D}}|$, que formam o modelo de predição. A variável $\mathcal{T}$ armazena a lista dos termos que aparecem nos documentos de treinamento; n armazena os valores de n_{c_j,t_i} para todas as classes em $\mathcal{C}$ e todos os termos em $\mathcal{T}$; $\hat{n}$ armazena os valores de $\hat{n}_{c_j}$ para todas as classes em $\mathcal{C}$; ϕ armazena a frequência ϕ_{c_j,t_i} para todos os termos em $\mathcal{T}$ e todas as classes em $\mathcal{C}$; e $|\hat{\mathcal{D}}|$ armazena os valores de $|\hat{\mathcal{D}}_{c_j}|$ para todas as classes em $\mathcal{C}$. Depois do treinamento, essas informações são usadas para a classificação de dados não rotulados, baseada no Algoritmo 2, onde é selecionada a classe com o menor tamanho de descrição.

As variáveis $\mathcal{T}$, n, $\hat{n}$, ϕ e $|\hat{\mathcal{D}}|$ compõem o modelo de predição, mas são entradas opcionais no estágio de treinamento (Algoritmo 1), pois o método proposto é naturalmente adaptado para o aprendizado *online* e incremental. Portanto, se as variáveis do modelo de predição são dadas como entrada no Algoritmo 1, elas são atualizadas com as informações obtidas pela análise dos novos documentos de treinamento.

Algoritmo 1 Estágio de treinamento do `MDLText`

função MDL_TREINAMENTO($\mathcal{D}^\star$, $\mathcal{C}$, $\mathcal{T}$, n, $\hat{n}$, ϕ, $|\hat{\mathcal{D}}|$)
Entradas: $\mathcal{D}^\star$ (conjunto de treinamento), $\mathcal{C}$ (conjunto de todas as possíveis classes), $\mathcal{T}$ (lista de termos), n (soma dos pesos de cada termo em $\mathcal{T}$ para cada classe em $\mathcal{C}$), $\hat{n}$ (soma dos pesos em n para cada classe em $\mathcal{C}$), ϕ (frequência de cada termo em $\mathcal{T}$ para cada classe em $\mathcal{C}$) e $|\hat{\mathcal{D}}|$ (número de documentos para cada classe em $\mathcal{C}$). Os parâmetros $\mathcal{T}$,n, $\hat{n}$, ϕ e $|\hat{\mathcal{D}}|$ são opcionais.
Saídas: $\mathcal{T}$, n, $\hat{n}$, ϕ e $|\hat{\mathcal{D}}|$.

    se $\mathcal{T}$, n, $\hat{n}$, ϕ e $|\hat{\mathcal{D}}|$ não forem apresentados **então**
        $\mathcal{T} \leftarrow \emptyset$ ▷ Cria um dicionário vazio.
        $|\hat{\mathcal{D}}_{\forall_c}| \leftarrow 0$ ▷ Inicializa o número de documentos para cada classe em $\mathcal{C}$.
    fim se
    para cada documento d em $\mathcal{D}^\star$ **faça**
        $c \leftarrow$ classe de d
        $|\hat{\mathcal{D}}_c| \leftarrow |\hat{\mathcal{D}}_c| + 1$ ▷ Número de documentos da classe c.
        para cada termo t_i in d **faça**
            se t_i não estiver em $\mathcal{T}$ **então**
                $\mathcal{T} \leftarrow \mathcal{T} \cup t_i$ ▷ Insere o termo t_i no dicionário.
                $n_{\forall_c,t_i} \leftarrow 0$ ▷ Inicializa a soma dos pesos de t_i para cada classe em $\mathcal{C}$.
                $\hat{n}_{\forall_c} \leftarrow 0$ ▷ Inicializa a soma dos pesos em n para cada classe em $\mathcal{C}$.
                $\phi_{\forall_c,t_i} \leftarrow 0$ ▷ Inicializa a frequência de t_i para cada classe em $\mathcal{C}$.
            fim se
            $n_{c,t_i} \leftarrow n_{c,t_i} + \hat{w}(t_i,d)$ ▷ Soma dos pesos de t_i na classe c (n_{c,t_i} é uma posição de n).
            $\hat{n}_c \leftarrow \hat{n}_c + n_{c,t_i}$ ▷ Soma dos pesos em dos pesos em n para a classe c ($\hat{n}_c$ é uma posição de $\hat{n}$).
            $\phi_{c,t_i} \leftarrow \phi_{c,t_i} + 1$ ▷ Frequência do termo t_i na classe c (ϕ_{c,t_i} é uma posição de ϕ).
        fim para
    fim para
    retorne $\mathcal{T}$, n, $\hat{n}$, ϕ e $|\hat{\mathcal{D}}|$
fim função

4. Técnicas de normalização léxica e indexação semântica

Os comentários no YouTube normalmente são curtos e repletos de gírias, expressões idiomáticas, símbolos e *emoticons* que podem dificultar até mesmo a tarefa de *tokenização*. Nesse cenário, métodos tradicionalmente empregados em categorização de textos podem apresentar baixo desempenho [Almeida et al. 2016]. Contudo, técnicas tra-

Algoritmo 2 Estágio de classificação do `MDLText`

1: **função** MDL_CLASSIFICAÇÃO(d, $\mathcal{C}$, $\mathcal{T}$, n, $\hat{n}$, ϕ e $|\hat{\mathcal{D}}|$)
2: **Entradas:** d (documento não rotulado), $\mathcal{C}$ (conjunto de todas as possíveis classes), $\mathcal{T}$ (dicionário dos termos obtido no estágio de treinamento), n (soma dos pesos de cada termo em $\mathcal{T}$ para cada classe em $\mathcal{C}$), $\hat{n}$ (soma dos pesos em n para cada classe em $\mathcal{C}$), ϕ (frequência de cada termo em $\mathcal{T}$ para cada classe em $\mathcal{C}$) e $|\hat{\mathcal{D}}|$ (número de documentos para cada classe em $\mathcal{C}$).
3: **Saída**: $c(d)$ (a classe predita para o documento d).

4: **para cada** termo t_i em d **faça**
5: **se** t_i estiver em $\mathcal{T}$ **então**
6: $F(t_i) \leftarrow$ pontuação_do_termo$(t_i, \phi_{\forall_c,t_i})$
7: **senão**
8: $F(t_i) \leftarrow 0$
9: **fim se**
10: Baseado em $F(t_i)$, use a Eq. 8 para calcular $K(t_i)$.
11: **fim para**
12: **para cada** classe c_j em $\mathcal{C}$ **faça**
13: $L(d|c_j) \leftarrow 0$
14: **para cada** termo t_i em d **faça**
15: **se** t_i não estiver em $\mathcal{T}$ **então**
16: $n_{c_j,t_i} \leftarrow 0$
17: $\hat{n}_{c_j} \leftarrow 0$
18: **fim se**
19: Baseado em n_{c_j,t_i} e $\hat{n}_{c_j}$, use a Eq. 7 para calcular $\beta(t_i|c_j)$.
20: Baseado em $\beta(t_i|c_j)$, use a Eq. 4 para calcular $L(t_i,c_j)$.
21: $L(d|c_j) \leftarrow L(d|c_j) + L(t_i,c_j) \times K(t_i)$. ▷ Acumula a soma dos tamanhos de descrição dos termos penalizado pelas suas pontuações. Esta linha de código é uma parte da Eq. 3.
22: Baseado em n_{c_j,t_i} e $|\hat{\mathcal{D}}_{c_j}|$, use a Eq. 12 para calcular $\overline{c}_j(t_i)$ (essa variável é uma posição do vetor protótipo $\overline{c}_j$).
23: **fim para**
24: **fim para**
25: **para cada** classe c_j em $\mathcal{C}$ **faça**
26: Calcule a similaridade $S(d,\overline{c}_j)$ entre d e $\overline{c}_j$ usando a Eq. 11.
27: Baseado em $S(d,\overline{c}_j)$, use a Eq. 10 para calcular $\hat{S}(d,c_j)$.
28: $L(d|c_j) \leftarrow \hat{S}(d,c_j) \times L(d|c_j)$ ▷ Essa linha de código é uma parte da Eq. 3 iniciada na linha 21.
29: **fim para**
30: $c(d) = \arg\min_{\forall c} L(d|c_j)$ ▷ O rótulo da classe com o menor tamanho de descrição.
31: **retorne** $c(d)$
32: **fim função**

dicionais de processamento de linguagem natural podem ajudar a amenizar esses proble-

mas, tais como a normalização léxica, indexação semântica e desambiguação.

- *Normalização léxica*: consiste em traduzir gírias, expressões idiomáticas e abreviações de palavras geralmente usadas pelos usuários da Internet para a sua forma canônica. São utilizados dicionários de *Lingo*, como o dicionário `NoSlang`[5] [Almeida et al. 2016].
- *Indexação semântica*: técnica de extração de informações semânticas que obtém diferentes significados para um determinado termo. O uso dessa técnica tende a aumentar a quantidade de termos de um texto e por isso ela é considerada uma técnica de geração de conceitos, expansão ou enriquecimento textual [Almeida et al. 2016]. Para a geração de conceitos, geralmente é utilizado algum repositório semântico, como o `LDB BabelNet`[6]. Porém, outras fontes de informação podem ser utilizadas, como o Wikipedia [Almeida et al. 2016, Vo and Ock 2015].
- *Desambiguação*: também conhecida como *word sense disambiguation*. É empregada para selecionar os conceitos mais relevantes de um termo de acordo com o contexto do texto. Em algumas aplicações, tais como a proposta por [Almeida et al. 2016], a desambiguação é considerada uma etapa posterior à indexação semântica, pois primeiramente são encontrados diferentes significados para o termo e depois aqueles mais relevantes são selecionados.

Na Tabela 1, é apresentado um exemplo de uma amostra de texto original (a), a mesma amostra após normalização léxica (b), após indexação semântica (c) e após a etapa de desambiguação (d).

Tabela 1. Exemplo de uma amostra original (a) e após normalização léxica (b), indexação semântica (c) e desambiguação (d).

(a) Amostral original	*plz there is sth u shud knw abt ur bf.*
(b) Após normalização léxica	*please there be something you should know about your boyfriend*
(c) Após indexação semântica	*please there be something you should know knowledge noesis about your boyfriend fellow swain young_man*
(d) Após desambiguação	*please there be something you should cognition about your boyfriend*

Neste trabalho, foi utilizada a ferramenta *online* `TextExpansion`[7], proposta por [Almeida et al. 2016], para normalizar e expandir as amostras de texto com os comentários das bases de dados do YouTube. Basicamente, essa ferramenta processa cada exemplo de texto puro (sem pré-processamento) em três diferentes estágios, cada um gerando uma nova representação em ciclos [Almeida et al. 2016]: (i) normalização léxica, (ii) indexação semântica e (iii) desambiguação. Então, a partir de uma regra de expansão, as amostras resultantes em cada estágio são unificadas para formar um único exemplo, que pode ser processado por uma técnica de aprendizado de máquina. A Figura 6 ilustra o processo.

[5]O dicionário `NoSlang` está disponível em: `http://www.noslang.com/dictionary/full/`. Acessado em: 09/03/2017.

[6]O `LDB BabelNet` está disponível em: `http://babelnet.org/`. Acessado em: 09/03/2017.

[7]A ferramenta `TextExpansion` está publicamente disponível em: `http://lasid.sor.ufscar.br/expansion`. Acessado em 09/03/2017.

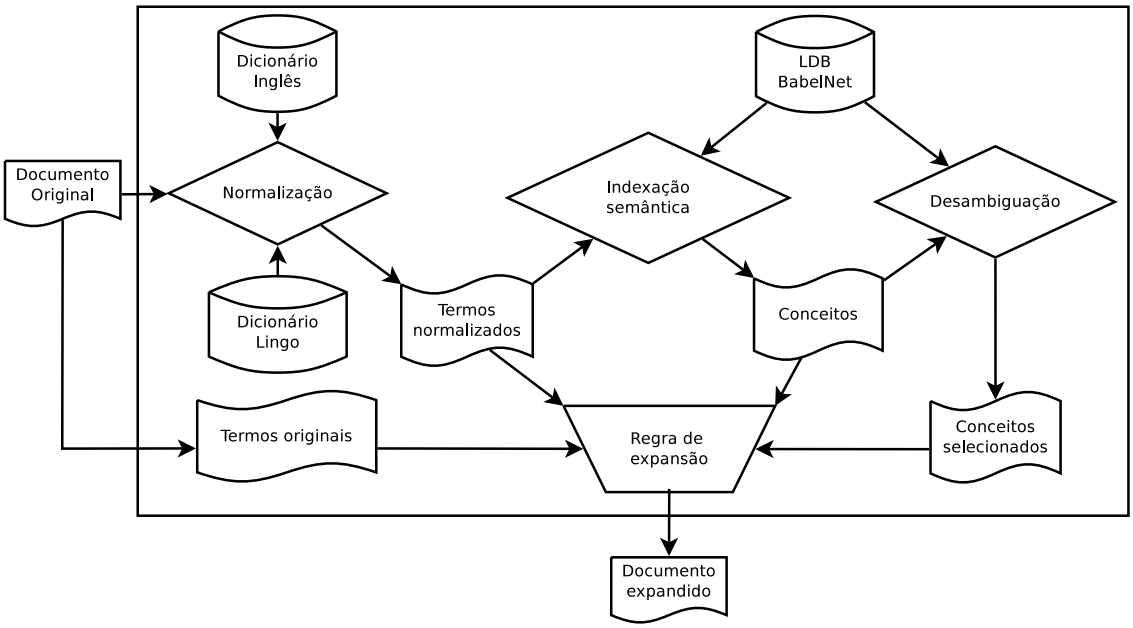


Figura 6. O documento de texto original é processado por técnicas de normalização léxica, indexação semântica e detecção de contexto, criando-se novos documentos expandidos e normalizados. Dada uma regra de expansão, os documentos são unificados formando uma amostra final (Fonte: [Almeida et al. 2016] (adaptado)).

Após processar um documento de texto usando a `TextExpansion`, dez novas versões expandidas desse documento podem ser geradas, uma para cada possível regra de expansão, definida pela combinação de quatro parâmetros: (1) manter os termos originais, (2) aplicar normalização léxica, (3) obter conceitos por indexação semântica e (4) aplicar desambiguação. Todas as possíveis regras de expansão são apresentadas na Tabela 2. Assim, através da `TextExpansion`, é possível criar dez novas amostras para cada documento de texto original, sendo que cada uma é o resultado de cada possível regra de expansão. Para maiores detalhes, consulte [Almeida et al. 2016].

Tabela 2. Possíveis regras de expansão utilizadas na `TextExpansion`.

	(1) Manter os termos originais	(2) Aplicar normalização	(3) Aplicar indexação semântica	(4) Aplicar desambiguação
Expansão 1	X		X	
Expansão 2	X			X
Expansão 3	X	X		
Expansão 4	X	X	X	
Expansão 5	X	X		X
Expansão 6			X	
Expansão 7				X
Expansão 8		X		
Expansão 9		X	X	
Expansão 10		X		X

5. Ensemble das predições obtidas por diferentes regras de expansão

[Silva et al. 2016a] verificaram que os métodos de classificação obtêm melhores resultados na filtragem *online* de *spam* em comentários do YouTube quando algumas das regras de expansão apresentadas na Tabela 1 são aplicadas. Contudo, nenhuma regra de expansão foi estatisticamente superior a todas as demais. Diante disso, é necessário testar todas elas para cada base de dados, para garantir que a melhor regra seja sempre selecionada. Para evitar esse processo de seleção de regra, o presente artigo propõe o uso de uma técnica *ensemble* que faz a combinação das predições usando diferentes regras.

Dado que é possível criar dez variações para cada documento de texto original, pode-se propor uma combinação das predições obtidas pelos métodos de classificação usando cada uma dessas onze amostras ao invés de usá-las individualmente. Assim, a Figura 7 ilustra a proposta de uma técnica *ensemble* que combina as predições individuais obtidas usando o documento de texto original e os documentos gerados pela `TextExpansion`, onde:

- h(orig.) é o modelo de predição treinado com os documentos originais;
- h(exp. 1),...,h(exp. 10) são os modelos de predição treinados com documentos gerados por cada regra de expansão;
- d(orig.) é o documento original;
- d(exp. 1),...,d(exp. 10) são os documentos gerados por cada regra de expansão;
- $\hat{y}$(orig.) é a classe predita pelo modelo h(orig.);
- $\hat{y}$(exp. 1),...,$\hat{y}$(exp. 10) são as classes preditas pelos modelos h(exp. 1),...,h(exp. 10);
- y é a classe final
- E(orig.) é o erro de predição do modelo h(orig.);
- E(exp. 1),...,E(exp. 10) são os erros de predição dos modelos h(exp. 1),...,h(exp. 10).

Conforme ilustrado, há um modelo preditivo gerado usando os documentos de treinamento originais e dez outros modelos preditivos gerados usando os documentos pré-processados pela `TextExpansion` (um modelo preditivo para cada regra de expansão apresentada na Tabela 2). Esses modelos de predição podem ser gerados usando qualquer método de classificação *online*. Quando um documento de texto com classe desconhecida é apresentado, cada modelo preditivo emite uma predição. Então, a classe final será atribuída ao documento por meio do voto majoritário, ou seja, a classe que foi mais votada pelos modelos é escolhida. Se o erro de predição de um modelo preditivo for maior que zero, então a classe verdadeira do documento é apresentada ao modelo de predição e ele é atualizado.

Como é de se esperar, a técnica *ensemble* demanda um alto custo computacional, pois ela cria e atualiza onze modelos preditivos simultaneamente. Porém, este problema pode ser minimizado se os modelos preditivos forem processados de modo paralelo.

6. Metodologia experimental

Para simular um ambiente real de filtragem de *spam* em comentários do YouTube, foi considerado o seguinte cenário: um pequeno número de amostras é disponibilizado para treinar o método de classificação (20% de cada base de dados). O restante das amostras é usada para formar o conjunto de teste. Uma mensagem por vez do conjunto de teste é

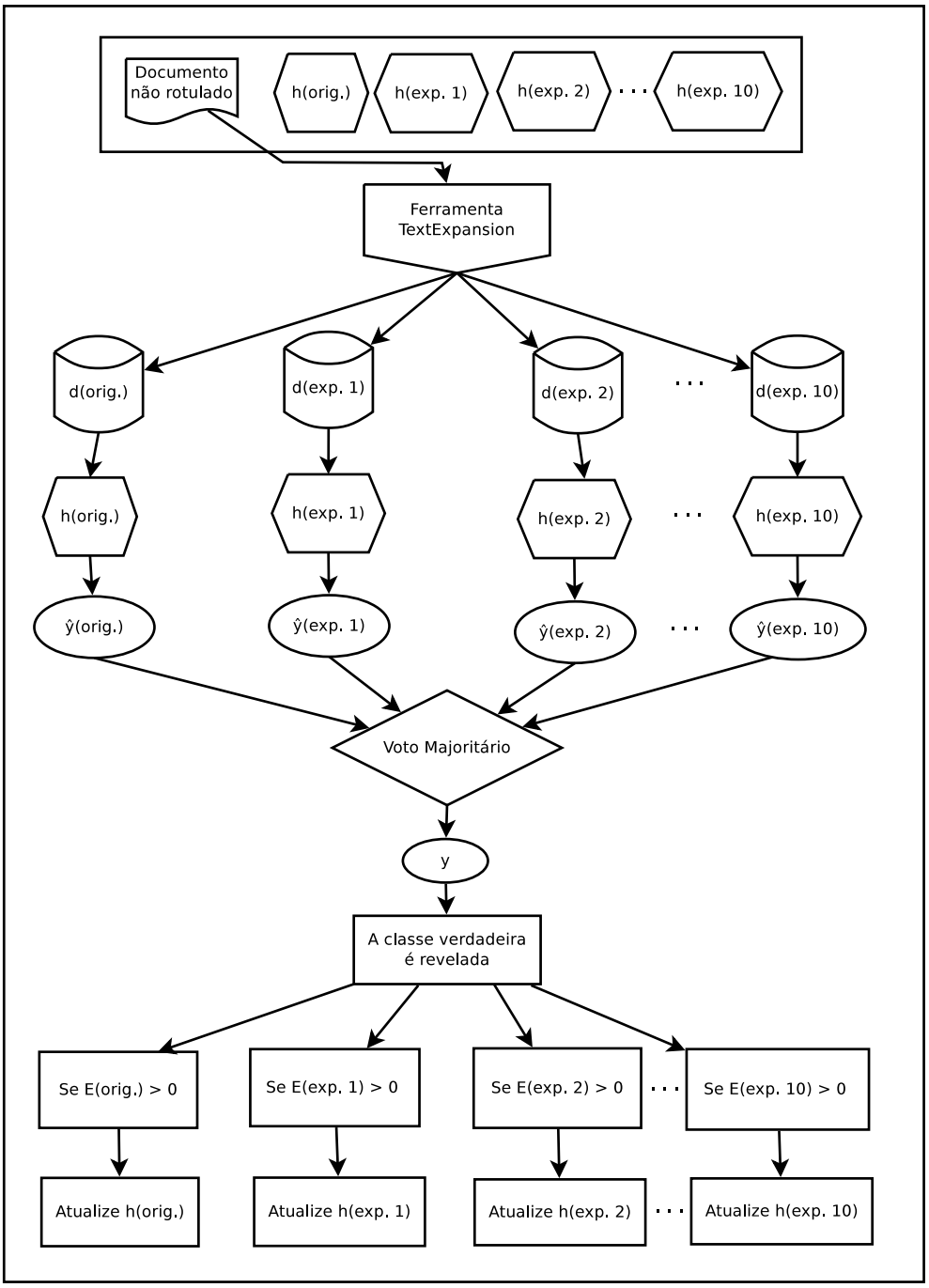


Figura 7. Técnica *ensemble* que combina os classificadores usando os documentos originais e os documentos gerados pelas regras de expansão.

apresentada ao classificador, que faz uma predição. Em seguida, o classificador recebe um *feedback* e, caso o erro de predição seja maior que zero, o modelo de treinamento é atualizado com a classe correta da amostra.

O cenário descrito acima é comum em aplicações *online*, como classificação de email. Por exemplo, quando o usuário recebe uma mensagem, o classificador usado pelo serviço de email pode considerar que aquela mensagem é *spam* e enviá-la para a caixa de *spam*. Porém, o usuário pode acessar a caixa de *spam* e avisar ao serviço de email que

ele fez uma classificação incorreta, clicando no botão "Não é spam". Esse *feedback* é usado pelo serviço de email para treinar o classificador e evitar que novos erros ocorram futuramente.

O método proposto neste artigo poderia ser usado de forma semelhante ao que é feito nos serviços de email. Por exemplo, uma opção para reportar *spam* poderia ser fornecida ao usuário do YouTube para cada comentário. Também, poderia existir uma caixa de *spam*, onde ele poderia ter acesso aos comentários que foram banidos. Portanto, o método de classificação faria uma predição para cada comentário e o usuário poderia fornecer um *feedback* caso não concorde com a classe escolhida para um determinado comentário.

É importante notar que no cenário considerado neste artigo, as medidas de desempenho consideram as classes escolhidas pelo método antes dele receber o *feedback*.

O processo de classificação *online* é detalhado no Algoritmo 3, que considera que as amostras são apresentadas usando representações *bag-of-words*, sendo os pesos a frequência de cada termo em cada documento. Para a técnica *ensemble*, o processo de classificação *online* foi adaptado para corresponder ao fluxograma apresentado na Figura 7.

Algoritmo 3 *Framework* utilizado nos experimentos

função APRENDIZADO_ONLINE($\mathcal{D}$)
    $\mathcal{D}_{treino} \leftarrow$ seleciona aleatoriamente 20% dos exemplos contidos em $\mathcal{D}$
    $\mathcal{D}_{teste} \leftarrow$ seleciona o restante dos documentos contidos em $\mathcal{D}$

    $DF \leftarrow$ quantidade de documentos em que cada termo aparece
    $n_{treino} \leftarrow$ quantidade de documentos em $\mathcal{D}_{treino}$
    $\mathcal{D}_{treino} \leftarrow$ converte os documentos de $\mathcal{D}_{treino}$ para $TF\text{-}IDF$ usando DF e n_{treino}

    $modelo \leftarrow$ inicializa e atualiza o modelo usando $\mathcal{D}_{treino}$

    $i \leftarrow 1$
    para cada documento d em $\mathcal{D}_{teste}$ **faça**
        $\hat{d} \leftarrow$ cópia de d antes de convertê-lo para $TF\text{-}IDF$
        $d \leftarrow$ converte d para $TF\text{-}IDF$ usando DF e n_{treino}
        $\hat{c}_i \leftarrow$ o classificador faz a predição da classe do documento d
        $c_i \leftarrow$ classe verdadeira do documento d
        $erro \leftarrow$ o classificador calcula o erro de predição
        se $erro > 0$ **então**
            $DF \leftarrow$ atualiza DF
            $n_{treino} \leftarrow n_{treino} + 1$
            $d \leftarrow$ converte $\hat{d}$ para $TF\text{-}IDF$ usando DF e n_{treino}
            modelo $\leftarrow$ atualiza o modelo usando a classe c_i
        fim se
        $i \leftarrow i + 1$
    fim para
    retorna $\hat{c}$
fim função

No Algoritmo 3, o erro para o método MDLText é igual a 1 se $\hat{c}_i \neq c_i$, caso contrário o erro é igual a 0. O cálculo do erro para os outros métodos avaliados neste artigo varia de acordo com a estratégia de atualização do modelo de predição de cada um

deles.

6.1. Métodos

O desempenho do MDLText foi comparado com os dos seguintes métodos tradicionais de aprendizado *online*: (1) *naïve* Bayes multinomial (M.NB) [Manning et al. 2009], (2) *naïve* Bayes Bernoulli (B.NB) [Almeida et al. 2011], (3) Perceptron [Freund and Schapire 1999], (4) algoritmo de margem larga aproximada (ALMA – *Approximate Large Margin Algorithm*) [Hoi et al. 2014, Gentile 2002], (5) gradiente descendente *online* (OGD – *Online Gradient Descent*) [Zinkevich 2003], (6) gradiente descendente estocástico (SGD – *Stochastic Gradient Descent*) [Zhang 2004] e (7) algoritmo de margem máxima *online* relaxada (ROMMA – *Relaxed Online Maximum Margin Algorithm*) [Li and Long 2002].

Todos os métodos foram avaliados no problema de detecção de *spam* em um cenário de aprendizado *online*. Neste caso, cada amostra é apresentada por vez ao algoritmo de classificação que atualiza seu modelo de forma incremental. Os classificadores *online* são mais apropriados para a filtragem de comentários do YouTube, pois a capacidade de aprendizado incremental torna o modelo de predição mais dinâmico e adaptável.

Os métodos M.NB e B.NB foram implementados em Python usando funções da biblioteca scikit-learn[8] [Pedregosa et al. 2011]. Os métodos Perceptron, ALMA, OGD, PA e ROMMA foram implementados em MATLAB usando funções da biblioteca LIBOL[9] [Hoi et al. 2014]. Já, o MDLText foi implementado em C++. Todos os métodos foram testados com os parâmetros padrões definidos em suas respectivas bibliotecas.

6.2. Bases de dados

Para dar credibilidade aos resultados e tornar os experimentos reprodutíveis, todos os testes foram realizados com as seguintes bases de dados reais, públicas e não-codificadas[10]: *Eminem*, *Katy Perry*, *LMFAO*, *Psy* e *Shakira*. Elas foram coletadas e rotuladas por [Alberto et al. 2015].

6.3. Pré-processamento e *tokenização*

Em todos os experimentos com as bases de dados originais e com as obtidas pela expansão textual, os textos foram convertidos para letras minúsculas. Em seguida, na etapa de *tokenização*, os delimitadores utilizados foram quaisquer caracteres não alfanuméricos (exceto *underline*).

A Tabela 3 apresenta as estatísticas básicas sobre cada base de dados gerada pelas regras de combinação apresentadas na Tabela 2. $|\mathcal{D}|_{spam}$ indica a quantidade de comentários *spam*, $|\mathcal{D}|_{ham}$ é a quantidade de comentários legítimos, $|d|$ corresponde ao número de atributos (tamanho do vocabulário), $|\bar{d}|$ é a média do número de termos por documento e $|\tilde{d}|$ é o desvio padrão do número de termos por documento. Apesar da

[8] A biblioteca scikit-learn está disponível em http://scikit-learn.org/. Acessado em: 09/03/2017.

[9] A biblioteca LIBOL está disponível em: http://libol.stevenhoi.org/. Acessado em 09/03/2017.

[10] As bases de dados estão publicamente disponíveis em http://goo.gl/O1iExL. Acessado em 09/03/2017.

quantidade de amostras por classe ser a mesma para as bases de dados originais e com expansão textual, o tamanho do vocabulário, a média e o desvio padrão do número de termos por comentário variam para cada regra de combinação.

Tabela 3. Estatísticas das base de dados de comentários do YouTube.

	Eminem			Katy Perry			LMFAO			Psy			Shakira		
	$\|\mathcal{D}\|_{spam}$		$\|\mathcal{D}\|_{ham}$	$\|\mathcal{D}\|_{spam}$		$\|\mathcal{D}\|_{ham}$	$\|\mathcal{D}\|_{spam}$		$\|\mathcal{D}\|_{ham}$	$\|\mathcal{D}\|_{spam}$		$\|\mathcal{D}\|_{ham}$	$\|\mathcal{D}\|_{spam}$		$\|\mathcal{D}\|_{ham}$
	245		203	175		175	236		202	175		175	174		196
	$\|d\|$	$\|\bar{d}\|$	$\|\tilde{d}\|$	$\|d\|$	$\|\bar{d}\|$	$\|\tilde{d}\|$	$\|d\|$	$\|\bar{d}\|$	$\|\tilde{d}\|$	$\|d\|$	$\|\bar{d}\|$	$\|\tilde{d}\|$	$\|d\|$	$\|\bar{d}\|$	$\|\tilde{d}\|$
Base original	1601	19,04	23,67	1755	17,75	19,64	954	9,57	12,21	1429	14,29	17,18	1357	18,68	28,33
Expansão 1	4953	85,78	107,56	4977	68,54	76,14	3174	43,72	51,14	4120	61,65	73,83	4211	75,74	117,44
Expansão 2	1997	23,65	30,00	2129	21,84	24,09	1192	11,76	15,20	1734	18,01	21,71	1692	23,12	35,47
Expansão 3	1773	25,16	30,86	1896	23,36	25,22	1068	12,46	16,26	1574	19,75	23,97	1501	25,05	38,35
Expansão 4	5038	91,91	114,48	5035	74,15	81,77	3227	46,60	55,22	4195	67,11	80,25	4283	82,11	127,00
Expansão 5	2155	29,77	37,15	2260	27,45	29,83	1298	14,65	19,36	1871	23,47	28,32	1823	29,49	45,40
Expansão 6	4564	79,51	99,82	4631	63,59	70,80	2973	40,83	47,42	3834	57,66	69,21	3863	69,98	108,34
Expansão 7	1461	20,01	24,88	1666	19,12	20,89	932	10,09	12,76	1354	15,82	19,45	1233	19,62	29,56
Expansão 8	1424	19,65	24,44	1604	18,91	20,58	889	10,00	12,61	1307	15,63	19,30	1223	19,31	29,07
Expansão 9	4721	86,40	108,28	4768	69,71	77,03	3061	44,15	51,60	3953	62,99	75,78	4025	76,38	118,23
Expansão 10	1810	24,26	30,79	1974	23,01	25,03	1121	12,20	15,64	1610	19,36	23,61	1551	23,76	36,24

Conforme esperado, pode ser observado na Tabela 3 que a regra de expansão 4, que combina os termos originais, os termos normalizados e os conceitos obtidos pelo processo de indexação semântica, foi a que gerou o maior número de atributos e obteve a maior média de termos por comentário para todas as bases de dados. Por outro lado, a expansão 8, que aplica apenas normalização léxica, foi a que gerou o menor número de atributos e a menor média de termos por comentário.

6.4. Medidas de desempenho

Para comparar os resultados, foram empregadas as seguintes medidas de desempenho tradicionais na literatura de classificação de *spam*:

- *coeficiente de correlação de Matthews (MCC – Matthews correlation coefficient)*: tradicionalmente usado para avaliar e comparar filtros de *spam* [Almeida et al. 2011]. Essa medida tem como principal característica a avaliação que leva em conta o desbalanceamento entre as classes.
- *spam caught ou sensitividade*: proporção de mensagens *spam* corretamente identificadas;
- *blocked ham ou taxa de falsos positivos*: proporção de mensagens *ham* incorretamente classificadas como *spam*. Quanto menor, melhor é o desempenho;
- *F-medida*: média harmônica entre precisão (proporção de amostras classificadas como pertencentes à classe *spam* e que realmente pertencem à classe *spam*) e sensitividade (*spam caught*). Ela pode variar entre zero e um, sendo que valores próximos de um indicam que o classificador obteve bons resultados tanto na precisão, quanto na sensitividade.

7. Resultados

As Tabelas 4, 5, 6, 7 e 8 apresentam a média dos resultados obtidos em 50 rodadas dos experimentos mostrados no Algoritmo 3, sendo que em cada rodada os comentários foram aleatoriamente amostrados. Para cada método e base de dados, são apresentados os desempenhos obtidos com (1) os comentários originais e (2) a técnica *ensemble*. Os resultados estão ordenados pela F-medida. Os valores destacados em negrito indicam o melhor resultado para cada base de dados. Ainda, os valores são apresentados usando um mapa de calor em tons de cinza onde, para cada base de dados, quanto mais escura é a célula da tabela, melhor é o desempenho.

Tabela 4. Resultados obtidos nos experimentos com a base de dados Eminem.

	MCC		*Spam caught*		*Blocked ham*		F-medida	
	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.
MDLText	0.809	0.813	0.968	0.963	0.175	0.165	0.917	**0.918**
B.NB	**0.825**	0.762	0.887	0.948	**0.058**	0.200	0.916	0.897
M.NB	0.794	0.768	0.939	0.945	0.152	0.191	0.910	0.899
SGD	0.787	0.760	0.911	0.944	0.125	0.199	0.904	0.896
ROMMA	0.766	0.777	0.936	0.937	0.180	0.169	0.898	0.902
ALMA	0.766	0.770	0.948	0.944	0.197	0.187	0.898	0.900
Perceptron	0.754	0.746	0.965	0.961	0.237	0.239	0.894	0.890
OGD	0.684	0.674	**0.987**	0.982	0.360	0.364	0.864	0.861

Tabela 5. Resultados obtidos nos experimentos com a base de dados Katy Perry.

	MCC		*Spam caught*		*Blocked ham*		F-medida	
	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.
MDLText	0.811	**0.834**	**0.929**	0.928	0.120	0.095	0.907	**0.917**
SGD	0.752	0.820	0.888	**0.929**	0.136	0.110	0.877	0.911
B.NB	0.824	0.821	0.879	0.925	**0.057**	0.105	0.908	0.911
M.NB	0.793	0.814	0.897	0.922	0.104	0.109	0.897	0.908
Perceptron	0.770	0.795	0.912	0.920	0.144	0.126	0.887	0.899
ALMA	0.768	0.792	0.912	0.916	0.146	0.124	0.886	0.898
ROMMA	0.754	0.785	0.914	0.920	0.163	0.136	0.881	0.895
OGD	0.756	0.778	0.884	0.903	0.131	0.127	0.877	0.890

Tabela 6. Resultados obtidos nos experimentos com a base de dados LMFAO.

	MCC		*Spam caught*		*Blocked ham*		F-medida	
	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.
B.NB	**0.854**	0.822	0.915	0.942	**0.059**	0.125	**0.931**	0.920
SGD	0.813	0.823	0.919	0.943	0.107	0.125	0.914	0.920
M.NB	0.811	0.821	0.940	0.945	0.135	0.130	0.915	0.919
MDLText	0.791	0.798	0.927	0.924	0.140	0.129	0.906	0.908
ALMA	0.736	0.773	0.939	0.948	0.218	0.188	0.884	0.899
ROMMA	0.743	0.767	0.934	0.938	0.202	0.181	0.886	0.896
Perceptron	0.721	0.753	0.952	0.956	0.253	0.223	0.878	0.891
OGD	0.713	0.750	**0.973**	0.968	0.297	0.244	0.874	0.890

O `MDLText` obteve a melhor F-medida nos experimentos com as bases de dados *Eminem*, *Katy Perry*, *Psy* e *Shakira*. Individualmente, usando os comentários originais,

Tabela 7. Resultados obtidos nos experimentos com a base de dados Psy.

	MCC		*Spam caught*		*Blocked ham*		F-medida	
	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.
MDLText	0.870	**0.883**	0.953	0.954	0.084	0.071	0.936	**0.942**
M.NB	0.876	0.882	0.945	0.949	0.069	0.067	0.939	0.941
SGD	0.824	0.879	0.917	0.950	0.093	0.072	0.913	0.940
B.NB	0.870	0.878	0.915	0.950	**0.046**	0.073	0.933	0.939
Perceptron	0.841	0.857	**0.965**	0.955	0.128	0.099	0.922	0.929
ALMA	0.832	0.849	0.953	0.953	0.124	0.107	0.918	0.926
OGD	0.845	0.846	**0.965**	0.947	0.123	0.104	0.924	0.924
ROMMA	0.825	0.840	0.948	0.948	0.126	0.109	0.914	0.922

Tabela 8. Resultados obtidos nos experimentos com a base de dados Shakira.

	MCC		*Spam caught*		*Blocked ham*		F-medida	
	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.	Orig.	Ens.
MDLText	**0.840**	0.836	0.947	0.943	0.106	0.106	**0.917**	0.915
Perceptron	0.834	0.814	0.950	0.937	0.116	0.122	0.914	0.904
OGD	0.827	0.813	**0.953**	0.930	0.125	0.116	0.911	0.903
M.NB	0.825	0.800	0.892	0.889	0.069	0.090	0.906	0.894
ALMA	0.817	0.811	0.940	0.931	0.122	0.119	0.905	0.902
ROMMA	0.811	0.814	0.938	0.935	0.126	0.121	0.902	0.904
SGD	0.810	0.809	0.902	0.891	0.092	0.083	0.900	0.898
B.NB	0.758	0.803	0.756	0.892	**0.024**	0.090	0.848	0.895

o `MDLText` obteve a melhor F-medida nas bases de dados *Eminem* e *Shakira*. Nos experimentos com a técnica *ensemble*, ele obteve a melhor F-medida nas bases de dados *Eminem*, *Katy Perry*, *Psy* e *Shakira*.

Os resultados do `MDLText` considerando as outras medidas de desempenho também foram bastante expressivos. Ele obteve o melhor MCC nas bases de dados *Katy Perry*, *Psy* e *Shakira* e a melhor taxa de *spam caught* na base de dados *Katy Perry*. Quanto à taxa de *blocked ham*, o `MDLText` foi um dos quatro melhores métodos em todas as bases de dados.

Para certificar que os resultados não foram obtidos por acaso, foi realizada uma análise estatística usando o teste de Friedman. 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 [Demšar 2006]:

$$\chi_F^2 = \frac{12 \times q}{k \times (k+1)} \times \sum_{j=1}^{k} \left(R_j^2 - \frac{k+1}{2} \right)^2. \tag{13}$$

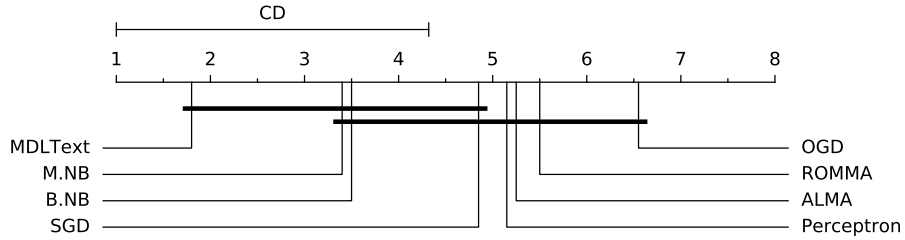
A hipótese nula é rejeitada se o valor crítico na distribuição χ^2, com $k-1$ graus de liberdade, é menor que χ_F^2 [Demšar 2006]. Para um intervalo de confiança $\alpha = 0{,}05$, $k = 8$ e $q = 10$, o valor crítico é $2{,}170$. Portanto, como $\chi_F^2 = 26{,}350$, a hipótese nula pode ser rejeitada.

Uma vez que a hipótese nula pôde ser rejeitada, foi realizado o teste *post-hoc* para comparar o desempenho par-a-par das técnicas. Para isso, foi usado o teste de Nemenyi.

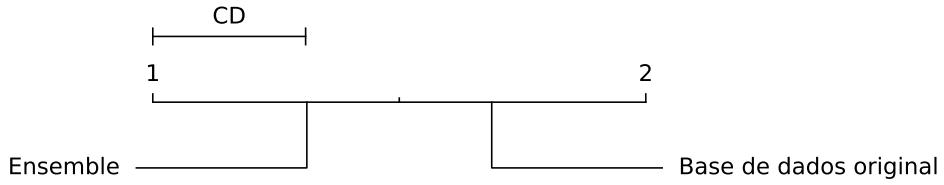
Nele, os desempenhos de dois métodos são significativamente divergentes se a diferença entre as médias dos *rankings* for maior ou igual a uma diferença crítica calculada pela seguinte equação [García et al. 2009, Demšar 2006]:

$$CD = q_\alpha \times \sqrt{\frac{k \times (k+1)}{6 \times q}}, \tag{14}$$

onde q_α corresponde ao valor crítico apresentado em Demšar [Demšar 2006, pág. 12]. Para um intervalo de confiança $\alpha = 0{,}05$, $k = 8$ e $q = 10$, a diferença crítica foi igual a $3{,}320$. A Figura 8(a) ilustra a análise estatística apresentando o *ranking* médio dos métodos (eixo do gráfico) e a diferença crítica (CD). Na figura, os métodos que não estão conectados são estatisticamente diferentes. Portanto, existe evidência estatística para afirmar que o `MDLText` foi superior aos métodos Perceptron, ALMA, ROMMA e OGD. No entanto, apesar do *ranking* médio ter sido o melhor, o `MDLText` foi considerado equivalente aos métodos B.NB, M.NB e SGD.



(a)



(b)

Figura 8. *Rankings* médios usados nas análises estatísticas e diferenças críticas calculadas usando o teste de Nemenyi. 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 8(a) apresenta o *ranking* médio obtido por cada método. A Figura 8(b) apresenta o *ranking* médio obtido nos experimentos com as bases de dados originais e com a técnica *ensemble*.

Comparando-se os resultados obtidos com as bases de dados originais e com a técnica *ensemble*, as Tabelas 4, 5, 6, 7 e 8 mostram que, na maioria das bases de dados, a melhor F-medida foi obtida nos experimentos com a técnica *ensemble*. Além disso, em

geral, para todas as medidas de desempenho, os métodos de classificação obtiveram os melhores resultados quando está técnica foi aplicada.

Para certificar-se que tais diferenças foram significativas, também foi realizada uma análise estatística usando o teste não-paramétrico de Friedman. Para um intervalo de confiança $\alpha = 0{,}05$, a hipótese nula de igualdade entre os resultados usando as bases originais e a técnica *ensemble* pode ser descartada. De acordo com a comparação par-a-par usando o teste *post-hoc* de Nemenyi, para um intervalo de confiança $\alpha = 0{,}05$, $k = 2$ e $q = 40$, a diferença crítica entre os métodos foi $0{,}310$ (Figura 8(b)). Portanto, pode-se afirmar que o desempenho geral obtido usando a técnica *ensemble* foi estatisticamente superior ao desempenho obtido usando os comentários originais.

8. Conclusões e trabalhos futuros

Neste trabalho, `MDLText`, um método *online* de classificação de texto baseado no princípio da descrição mais simples, foi aplicado na detecção automática de *spam* em comentário do YouTube. Esse método possui características desejáveis, tais como alta eficiência, aprendizado incremental e robustez a *overfitting*, o que habilita sua aplicação em problemas reais, dinâmicos e que envolvem classificação adversária, tais como a classificação de *spam*.

Para avaliar o desempenho do método, foram conduzidos experimentos em um cenário de classificação com cinco bases de dados públicas, reais e não-codificadas. O desempenho do `MDLText` foi comparado com os de métodos tradicionais de aprendizado *online*: M.NB, B.NB, SGD, OGD, ROMMA, ALMA e Perceptron. A análise estatística dos resultados mostrou que o `MDLText` foi significativamente superior à maioria dos métodos comparados.

Como as amostras originais são normalmente muito curtas e ruidosas, também foi proposta uma técnica *ensemble* que combina as predições dos métodos de classificação obtidas com comentários originais e com comentários gerados através da aplicação de técnicas de normalização léxica, indexação semântica e desambiguação. A técnica proposta obteve resultados promissores na classificação de comentários do YouTube, sendo que a análise estatística dos resultados confirmou que o desempenho geral da técnica *ensemble* foi superior ao obtido nos experimentos que usaram apenas os comentários originais.

Trabalhos futuros envolvem a aplicação do `MDLText` e da técnica *ensemble* em outros problemas de classificação de textos curtos e ruidosos, além da adaptação da `TextExpansion` para processar textos escritos em outras línguas. Também, serão estudadas adaptações na técnica *ensemble*, procurando novas formas de combinações dos atributos que possam produzir melhores resultados, ao mesmo tempo que mantenha sua natureza incremental.

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

Referências

Alberto, T. C., Lochter, J. V., and Almeida, T. A. (2015). Tubespam: Comment spam filtering on YouTube. In *Proceedings of the 14th International Conference on Machine Learning and Applications (ICMLA'15)*, pages 138–143, Miami, FL, USA. IEEE.

Almeida, T. A., Silva, T. P., Santos, I., and Hidalgo, J. M. G. (2016). Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering. *Knowledge-Based Systems*, 108:25–32.

Almeida, T. A., Yamakami, A., and Almeida, J. (2011). Spam filtering: how the dimensionality reduction affects the accuracy of naive Bayes classifiers. *Journal of Internet Services and Applications*, 1(3):183–200.

Ammari, A., Dimitrova, V., and Despotakis, D. (2012). Identifying relevant YouTube comments to derive socially augmented user models: A semantically enriched machine learning approach. In Ardissono, L. and Kuflik, T., editors, *Advances in User Modeling: UMAP 2011 Workshops, Girona, Spain, July 11-15, 2011, Revised Selected Papers*, pages 71–85, Berlin, Heidelberg. Springer Berlin Heidelberg.

Assis, F., Yerazunis, W., Siefkes, C., and 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.

Baldwin, T., Cook, P., Lui, M., MacKinlay, A., and Wang, L. (2013). How noisy social media text, how diffrnt social media sources? In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP'13)*, pages 356–364, Nagoya, Japan.

Barron, A., Rissanen, J., and Yu, B. (1998). The minimum description length principle in coding and modeling. *IEEE Transaction on Information Theory*, 44(6):2743–2760.

Bratko, A., Filipič, B., Cormack, G. V., Lynam, T. R., and Zupan, B. (2006). Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, 7:2673–2698.

Chaitin, G. J. (1969). On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the Association for Computing Machinery (JACM)*, 16(1):145–159.

Dalvi, N., Domingos, P., Mausam, Sanghai, S., and Verma, D. (2004). Adversarial classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 99–108, Seattle, WA, USA. ACM.

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

Eisenstein, J. (2013). What to do about bad language on the internet. In *Proceedings of NAACL-HLT 2013*, pages 359–369, Atlanta, GA, USA. Association for Computational Linguistics.

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

García, S., Fernández, A., Luengo, J., and 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.

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

Goodman, J., Heckerman, D., and Rounthwaite, R. (2005). Stopping spam. *Scientific American*, 292:42–49.

Grünwald, P., Kontkanen, P., Myllymäki, P., Silander, T., and Tirri, H. (1998). Minimum encoding approaches for predictive modeling. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*, pages 183–192, Madison, Wisconsin, USA. Morgan Kaufmann.

Grünwald, P. D. (2005). A tutorial introduction to the minimum description length principle. In *Advances in Minimum Description Length: Theory and Applications*. MIT Press.

Grünwald, P. D. (2007). *The Minimum Description Length Principle*. The MIT Press.

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

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

Kolmogorov, A. N. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–7.

Li, M. and Vitányi, P. M. (2008). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, New York, NY, 3rd edition.

Li, Y. and Long, P. M. (2002). The relaxed online maximum margin algorithm. *Machine Learning*, 46(1-3):361–387.

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

O'Callaghan, D., Harrigan, M., Carthy, J., and Cunningham, P. (2012). Network analysis of recurring YouTube spam campaigns. *CoRR*, abs/1201.3783.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rădulescu, C., Dinsoreanu, M., and Potolea, R. (2014). Identification of spam comments using natural language processing techniques. In *Proceedings of the 10th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP'14)*, pages 29–35, Cluj-Napoca, Romania. IEEE.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471.

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431.

Rissanen, J. (1996). Fisher information and stochastic complexity. *IEEE Transaction on Information Theory*, 42(1):40–47.

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.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656.

Silva, R. M., Alberto, T. C., Almeida, T. A., and 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.

Silva, R. M., Almeida, T. A., and 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.

Silva, R. M., Almeida, T. A., and 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.

Silva, R. M., Almeida, T. A., and Yamakami, A. (2017). MDLText: An efficient and lightweight text classifier. *Knowledge-Based Systems*, 118:152–164.

Solomonoff, R. J. (1964). A formal theory of inductive inference: Parts 1 & 2. *Information and Control*, 7(1, 2):1–22, 224–254.

Vo, D. and Ock, C. (2015). Learning to classify short text from scientific documents using topic models with various types of knowledge. *Expert Systems with Applications*, 42(3):1684–1698.

Wilbur, W. J. and Kim, W. (2009). The ineffectiveness of within-document term frequency in text classification. *Information Retrieval*, 12(5):509–525.

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.

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.