Túlio Casagrande Alberto

TubeSpam: Filtragem Automática de Comentários Indesejados Postados no YouTube

Sorocaba, SP

3 de Fevereiro de 2017

Túlio Casagrande Alberto

TubeSpam: Filtragem Automática de Comentários Indesejados Postados no YouTube

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC-So) da Universidade Federal de São Carlos como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação. Linha de pesquisa: Inteligência Artificial e Banco de Dados.

Universidade Federal de São Carlos – UFSCar

Centro de Ciências em Gestão e Tecnologia – CCGT

Programa de Pós-Graduação em Ciência da Computação – PPGCC-So

Orientador: Prof. Dr. Tiago Agostinho de Almeida

Sorocaba, SP

3 de Fevereiro de 2017

Casagrande Alberto, Túlio

TubeSpam: Filtragem Automática de Comentários Indesejados Postados no YouTube / Túlio Casagrande Alberto. -- 2017.
70 f. : 30 cm.

Dissertação (mestrado)-Universidade Federal de São Carlos, campus Sorocaba, Sorocaba
Orientador: Prof. Dr. Tiago Agostinho de Almeida
Banca examinadora: Profa. Dra. Solange Oliveira Rezende, Profa. Dra. Sahudy Montenegro González
Bibliografia

1. Comentários indesejados. 2. Classificação. 3. Aprendizado de máquina. I. Orientador. II. Universidade Federal de São Carlos. III. Título.

Ficha catalográfica elaborada pelo Programa de Geração Automática da Secretaria Geral de Informática (SIn).
DADOS FORNECIDOS PELO(A) AUTOR(A)

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências em Gestão e Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Tulio Casagrande Alberto, realizada em 03/02/2017:

Prof. Dr. Tiago Agostinho de Almeida
UFSCar

Profa. Dra. Solange Oliveira Rezende
USP

Profa. Dra. Sahudy Montenegro González
UFSCar

Agradecimentos

Agradeço,

à minha família, pelo suporte e carinho durante toda a minha vida, por não medirem esforços para que eu pudesse concluir os meus estudos.

aos meus amigos, pelos incontáveis momentos de diversão e descontração, pelos trabalhos em grupo, pela companhia e apoio.

aos meus colegas de pesquisa, Johannes e Renato, pelas inúmeras sessões de ajuda e artigos produzidos.

à UFSCar Sorocaba, pela qualidade de ensino e suporte aos estudantes e pesquisadores.

a todos os professores que participaram da minha formação, pelos ensinamentos valiosos, os quais levarei por toda a minha vida.

aos membros da banca examinadora, pelos comentários, sugestões e contribuições.

e em especial ao Prof. Tiago, pela oportunidade, paciência, motivação e ensinamentos que possibilitaram a conclusão deste trabalho.

"Qualquer tecnologia suficientemente avançada é indistinguível de magia."
Arthur C. Clarke

Resumo

O YouTube tem se tornado uma importante plataforma de compartilhamento de vídeos. Muitos usuários produzem regularmente conteúdo em vídeo e fazem desta tarefa seu principal meio de vida. Contudo, esse sucesso também vem despertando a atenção de usuários mal-intencionados, que propagam comentários e vídeos indesejados para se autopromoverem ou para disseminar *links* maliciosos que podem conter vírus e *malwares*. Visto que o YouTube atualmente oferece recursos limitados para bloquear *spam*, o volume dessas mensagens está impactando muitos usuários e proprietários de canais. Além da característica inerentemente *online* do problema, filtrar *spam* nos comentários do YouTube é uma tarefa que difere-se da tradicional filtragem de *spam* em emails, pois as mensagens costumam ser muito mais curtas e repletas de erros de digitação, gírias, símbolos e abreviações que podem dificultar a tarefa de classificação. Assim, nesta dissertação é apresentada a avaliação de desempenho obtido por métodos tradicionais de classificação *online* auxiliados por técnicas de normalização léxica e indexação semântica, quando aplicados na filtragem automática de comentários indesejados postados no YouTube. Foi avaliado também o desempenho do `MDLText`, um promissor método de classificação de texto baseado no princípio da descrição mais simples. A análise estatística dos resultados indica que os métodos `MDLText`, Passivo-Agressivo, *Naïve* Bayes, MDL e Gradiente Descendente *Online* obtiveram desempenhos equivalentes. Além disso, os resultados também indicam que o uso de técnicas de normalização léxica e indexação semântica são eficazes para atenuar os problemas de representação de texto e, consequentemente, aumentar o poder de predição dos métodos de classificação. Baseado nos resultados dos experimentos, foi proposto e desenvolvido o `TubeSpam`, uma ferramenta *online* para filtrar automaticamente comentários indesejados postados no YouTube.

Palavras-chaves: Comentários indesejados; Classificação; Aprendizado de máquina.

Abstract

YouTube has become an important video sharing platform. Several users regularly produce video content and make this task their main livelihood. However, such success is also drawing the attention of malicious users propagating undesired comments and videos, looking for self-promotion or disseminating malicious links which may have malwares and viruses. Since YouTube offers limited tools for blocking spam, the volume of such messages is shockingly increasing and harming users and channels owners. In addition to the problem being naturally online, comment spam filtering on YouTube is different than the traditional email spam filtering, since the messages are very short and often rife with spelling errors, slangs, symbols and abbreviations. This manuscript presents a performance evaluation of traditional online classification methods, aided by lexical normalization and semantic indexing techniques when applied to automatic filter YouTube comment spam. It was also evaluated the performance of `MDLText`, a promising text classification method based on the minimum description length principle. The statistical analysis of the results indicates that `MDLText`, Passive-Aggressive, *Naïve* Bayes, MDL and Online Gradient Descent obtained statistically equivalent performances. The results also indicate that the lexical normalization and semantic indexing techniques are effective to be applied to the problem. Based on the results, it is proposed and designed `TubeSpam`, an online tool to automatic filter undesired comments posted on YouTube.

Key-words: Undesired comments; Classification; Machine learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ALMA	*Approximate Large Margin Algorithm* (Algoritmo de Margem Larga Aproximada)
B.NB	*Naïve* Bayes Bernoulli
CF	*Confidence Factors* (Fatores de Confidência)
DOM	*Document Object Model* (Modelo de Objeto de Documentos)
MCC	*Matthews Correlation Coefficient* (Coeficiente de Correlação de Matthews)
MDL	*Minimum Description Length Principle* (Princípio da Descrição mais Simples)
M.NB	*Naïve* Bayes Multinomial
OGD	*Online Gradient Descent* (Gradiente Descendente *Online*)
PA	*Passive-Aggressive* (Passivo-Agressivo)
ROMMA	*Relaxed Online Maximum Margin Algorithm* (Algoritmo de Margem Máxima *Online* Relaxada)
SVM	*Support Vector Machines* (Máquinas de Vetores de Suporte)
TF	*Term frequency* (Frequência do termo)
TF-IDF	*Term frequency-inverse document frequency* (Frequência do termo-inverso da frequência nos documentos)

Lista de símbolos

d	Documento ou mensagem de texto
t_i	i-ésimo termo de um documento de texto
$\vec{x}$	Vetor de atributos que representa uma amostra
x_i	i-ésimo atributo de uma amostra
$tf(t_i, d)$	Número de vezes que o termo t_i aparece no documento d (*term frequency*)
$df(t_i)$	Número de documentos no conjunto de treinamento em que t_i aparece (*document frequency*)
$\mathcal{D}$	Conjunto de treinamento
$\|\mathcal{D}\|$	Quantidade de documentos no conjunto de treinamento
s	Passo do treinamento *online*
$\vec{w}$	Vetor de pesos utilizados pelos algoritmos de classificação
y	Classe ou rótulo verdadeiro
$\hat{y}$	Predição da classe
$\mathcal{Y}$	Conjunto de possíveis rótulos
ℓ	Valor do erro da predição
$\|d\|$	Quantidade de termos no documento d
f_n	Número de falsos negativos
f_p	Número de falsos positivos
v_n	Número de verdadeiros negativos
v_p	Número de verdadeiros positivos

Sumário

Prefácio

Combater *spam* é um problema importante do mundo *online*. Desde a primeira vez que a palavra *spam* foi empregada para descrever uma mensagem não-solicitada enviada em massa, esta praga infectou todos os meios populares de comunicação eletrônica por texto. Embora o *spam* em email seja a forma mais conhecida, ele também está presente em diversas outras aplicações, como blogs, SMS e mídias sociais.

A popularização da banda larga permitiu o crescimento da Internet e tornou populares os serviços de compartilhamento e hospedagem de vídeos. Segundo relatório recentemente publicado pela Sandvine[1], empresa que produz equipamentos para provedores de acesso à Internet, aproximadamente 70% do tráfego de *downstream* na América do Norte é proveniente da visualização de vídeos em plataformas como Netflix, YouTube e Amazon Video.

O YouTube surgiu como uma plataforma de compartilhamento e divulgação de vídeos, que também oferece recursos de mídias sociais, possibilitando a interação entre produtores e espectadores, por meio da seção de comentários. O sucesso do serviço pode ser mensurado por estatísticas publicadas pela própria empresa[2]: 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. Além disso, segundo o *ranking* Alexa[3], o YouTube é o segundo site mais acessado do mundo.

Recentemente, o YouTube adotou um sistema de monetização que gratifica os produtores, estimulando a produção de conteúdo original e aumentando o número de visualizações. No entanto, após a adoção do modelo, a plataforma foi inundada por conteúdo indesejado, geralmente constituído por informação de baixa qualidade, o *spam*.

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 1 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.

Segundo relatório da empresa de segurança computacional Nexgate[4], para cada

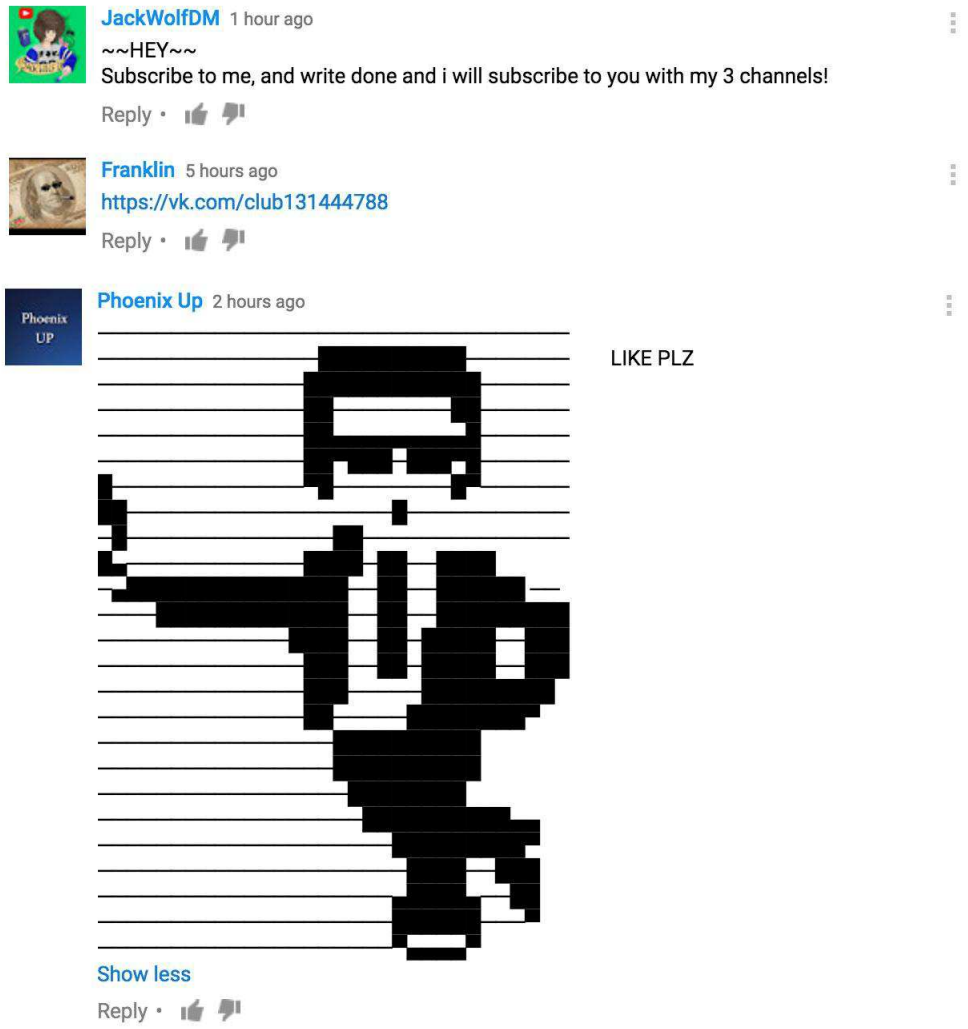
[1] *2016 Global Internet Phenomena Report*. Disponível em <https://www.sandvine.com/trends/global-internet-phenomena>, acessado em 22/10/2016.

[2] *YouTube Statistics*. Disponível em <https://www.youtube.com/yt/press/statistics.html>, acessado em 22/10/2016.

[3] *Top 500 Global Sites*. Disponível em <http://www.alexa.com/topsites>, acessado em 22/10/2016.

[4] *2013 State of Social Media Spam*. Disponível em <http://nexgate.com/solutions/nexgate-social-media-security-stat-center>, acessado em 22/10/2016.

Figura 1 – Exemplo de comentários *spam* comumente disseminados no YouTube.



Fonte: YouTube.

spam encontrado em qualquer mídia social, há outros 100 presentes no Facebook ou YouTube. Além disso, o YouTube possui cinco vezes mais conteúdo perigoso, insultos, ameaças e discursos de ódio do que outras mídias sociais.

O incômodo gerado é tão grande, que o usuário *"PewDiePie"*, proprietário do canal com maior número de inscritos do YouTube (mais de 50 milhões de assinantes), se pronunciou diversas vezes sobre a seção de comentários em seus vídeos. Em uma das ocasiões, o autor chegou a desabilitar completamente o recurso, alegando que a maior parte é representada por mensagens indesejadas e que não há ferramentas para facilitar a moderação[5].

[5] *PewDiePie switches off YouTube comments: 'It's mainly spam'*. Disponível em <https://www.theguardian.com/technology/2014/sep/03/pewdiepie-switches-off-youtube-comments-its-mainly-spam>, acessado em 22/10/2016.

Embora ações como esta evidenciem o problema do *spam* nos comentários do YouTube, existem relativamente poucos trabalhos na literatura que abordaram o assunto, havendo inclusive a escassez de bases de dados para serem usadas para avaliar o problema.

Outra característica importante do processo de filtragem de *spam* é que os padrões das mensagens de *spam* são alterados com frequência. Consequentemente, este é um problema de classificação que exige treinamento *online* e incremental. Por isso, muitos dos métodos de classificação considerados o estado da arte podem não ser apropriados na aplicação em filtros reais de *spam*, pois geralmente demandam um alto custo computacional para o treinamento.

Quando o escopo se restringe a mensagens de mídias sociais, incluindo o YouTube, técnicas consolidadas de filtragem de *spam* podem apresentar desempenho muito insatisfatório, devido ao fato das mensagens normalmente serem curtas e repletas de erros de digitação, gírias, símbolos, *emoticons* e abreviações, que podem dificultar a tarefa de classificação.

Objetivos e contribuições

O principal objetivo desta dissertação é oferecer uma análise de métodos de classificação e técnicas de processamento de linguagem natural, apresentando configurações promissoras para a filtragem automática de comentários indesejados publicados no YouTube. Também é proposta a ferramenta *online* `TubeSpam`, que emprega as melhores configurações encontradas.

Dentre as contribuições oferecidas neste trabalho, destacam-se:

1. Introdução à classificação de *spam* no contexto de mensagens curtas e ruidosas;
2. Estudo relacionado ao tratamento de atributos textuais, tais como o emprego de normalização léxica e indexação semântica;
3. Análise de diferentes métodos de classificação *online*;
4. Criação de novas bases de dados rotuladas com cometários do YouTube para futuras pesquisas; e
5. Desenvolvimento de uma ferramenta *online* de filtragem automática de comentários indesejados no YouTube, o `TubeSpam`.

Organização

Este manuscrito está estruturado da seguinte forma:

- No Capítulo 1, é introduzido o problema do *spam* nos comentários do YouTube, bem como é apresentada a revisão dos principais trabalhos existentes na literatura;
- No Capítulo 2, são introduzidos conceitos da representação computacional de amostras textuais, assim como técnicas de normalização léxica e indexação semântica;
- No Capítulo 3, são brevemente apresentados os métodos de classificação *online* avaliados neste trabalho;
- No Capítulo 4, são detalhados a proposta, a metodologia experimental e os resultados obtidos nesta pesquisa.
- No Capítulo 5, é apresentada a ferramenta *online* `TubeSpam` resultante das pesquisas realizadas ao longo deste trabalho.
- Finalmente, no último capítulo, são oferecidas as principais conclusões e direcionamentos para trabalhos futuros.

1 O Spam

Com a popularização da Internet e o crescimento no uso do email como um meio de comunicação popular, rápido e barato, surgiu um dos principais problemas da comunicação eletrônica em geral: o envio em massa de mensagens não-solicitadas (SAHAMI et al., 1998). Tal fenômeno ficou conhecido como *spamming*, as mensagens em si como *spam* e os autores como *spammers* (GYÖNGYI; GARCIA-MOLINA, 2005).

Entre as consequências do *spam* estão o consumo de recursos da infraestrutura da rede para trafegar as mensagens, assim como a atenção gasta pelo usuário, para selecionar e filtrar as mensagens indesejadas (BRATKO et al., 2006). Além disso, existe também o custo gerado para as vítimas dos crimes que geralmente acompanham o *spam*, tais como roubo financeiro, roubo de dados e propriedade intelectual, vírus e *malwares*, exposição à fraudes, pornografia e propaganda enganosa (GOODMAN; HECKERMAN; ROUNTHWAITE, 2005).

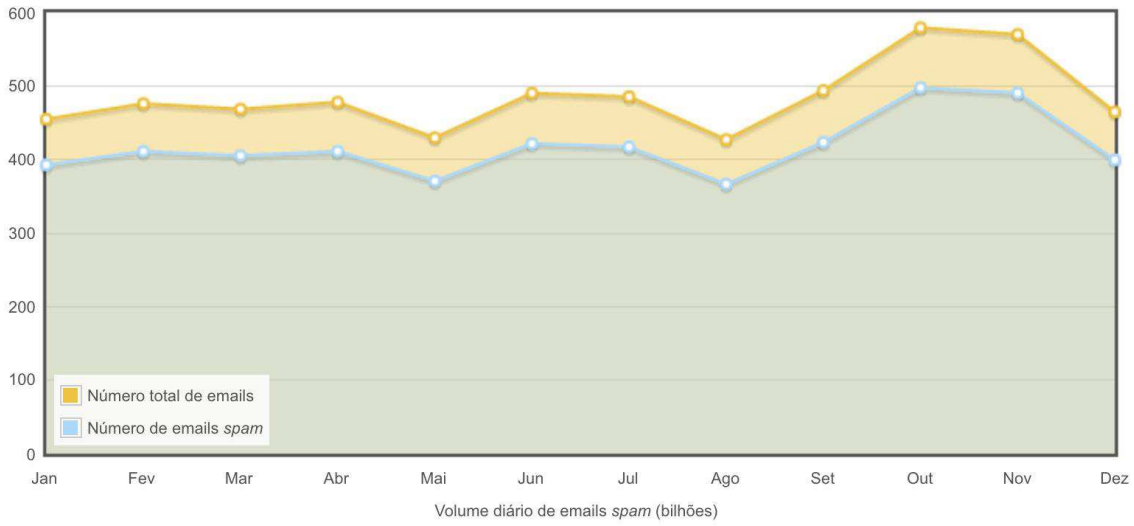
Segundo dados da Cisco, empresa que fornece soluções computacionais, cerca de 86% do volume diário de emails é *spam*, podendo ultrapassar a quantidade de 500 bilhões de mensagens *spam* por dia[1]. Os principais assuntos são notificações falsas de pagamentos e depósitos bancários, compras de produtos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

Desde o seu surgimento, diversas abordagens foram propostas para combater o *spam*, desde instrumentos jurídicos para punir os *spammers*, até sistemas de identificação e de reputação, em que os remetentes podem se certificar como usuários legítimos (GOODMAN; HECKERMAN; ROUNTHWAITE, 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 aprendizado de máquina são particularmente eficazes nesta tarefa, pois são capazes de se ajustarem à base de treinamento e aprenderem as características das amostras (SEBASTIANI, 2002).

As primeiras tentativas de filtrar *spam* em emails foram inicialmente bem sucedidas (SAHAMI et al., 1998), porém não demorou muito para que os *spammers* aprendessem a burlar os primeiros filtros manipulando as mensagens de diversas formas (FAWCETT, 2003). Isso acabaria gerando uma corrida entre os especialistas por trás dos filtros de *spam*, constantemente propondo novas regras para a filtragem, e os *spammers*, encontrando novas formas de evadir a detecção (DALVI et al., 2004). Esse fenômeno é conhecido como

[1] *Spam Overview*. Disponível em <https://www.senderbase.org/static/spam>, acessado em 02/01/2017.
[2] *Cisco 2014 Annual Security Report*. Disponível em <https://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf>, acessado em 24/10/2016.

Figura 2 – Volume diário de emails *spam* enviados no ano de 2016 (em bilhões).



Fonte: SenderBase, Cisco (adaptado).

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).

Possuir um adversário ativo confere à filtragem de *spam* um problema especial para a categorização automática de texto. Como o *spam* evolui constantemente 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, tais como: blogs, mídias sociais, fóruns de discussão, sites de notícias, wikis, sites de *e-commerce*, sites de relacionamento, jogos eletrônicos, mensageiros instantâneos e muitos outros (CHAKRABORTY et al., 2016). Independente do meio empregado, o *spam* está associado à informação indesejada, de baixa qualidade, que atrapalha a visualização do conteúdo que realmente interessa ao usuário e pode ser apresentado na forma de imagem, texto ou vídeo.

O YouTube, embora possua a presença marcante desta praga, ainda foi pouco abordado na literatura. Chaudhary e Sureka (2013) e Chowdury et al. (2013) propuseram métodos para detectar vídeos *spam*, uma prática que era bastante comum no YouTube, visto que era possível publicar vídeos-resposta a outros vídeos. Algumas publicações populares chegavam a receber milhares de vídeos-resposta, abrindo espaço para a disseminação de vídeos não relacionados com a postagem original, além de conteúdo publicitário e pornográfico.

Como a funcionalidade de publicar vídeos-resposta foi removida do YouTube[3], a maior parte de conteúdo indesejado ainda presente na plataforma está nos comentários. Em um dos trabalhos pioneiros sobre comentários *spam*, Mishne, Carmel e Lempel (2005) propuseram uma abordagem para detectar comentários *spam* em blogs, comparando a similaridade entre o modelo de linguagem utilizado na postagem original, nos comentários e nas páginas apontadas pelos *links* dos comentários. No entanto, essa estratégia não pode ser aplicada no YouTube, uma vez que os vídeos são postados com pouca ou nenhuma descrição textual, dificultando o processo de geração de modelos de linguagem para mapear a publicação original (ALBERTO; LOCHTER; ALMEIDA, 2015).

Rădulescu, Dinsoreanu e Potolea (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. Os autores extraíram nove atributos numéricos dos comentários e aplicaram técnicas de processamento de linguagem natural e de aprendizado de máquina para identificar os comentários *spam*. Contudo, a abordagem não obteve bons resultados no contexto do YouTube e os autores atribuíram isso ao fato de que as mensagens na plataforma normalmente são curtas e repletas de erros de digitação.

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 são muito curtas e repletas de erros de digitação, gírias, símbolos, *emoticons* e abreviações (AMMARI; DIMITROVA; DESPOTAKIS, 2012; EISENSTEIN, 2013). De forma mais objetiva, Baldwin et al. (2013) realizaram análises linguísticas e estatísticas em diversas bases de dados de mensagens de mídias sociais para medir o nível de ruído existente. As estatísticas apontam que um comentário do YouTube possui em média 16 palavras e 31,9% delas falharam em ser interpretadas pelo analisador gramatical. Os autores concluíram que, embora o texto encontrado em mídias sociais seja ruidoso, é possível aplicar técnicas de processamento de linguagem natural para diminuir o ruído.

O'Callaghan et al. (2012) propuseram aplicar redes funcionais (*network motifs*) para identificar campanhas de *spam* no YouTube. O trabalho apresentou bons resultados, porém limita-se a identificar mensagens muito semelhantes propagadas por *bots*. Conforme apontado por Alberto, Lochter e Almeida (2015) em um estudo mais recente, a maior parte do *spam* disseminado no YouTube costuma não ser enviado por *bots*, mas por usuários reais que tentam se autopromover através de vídeos populares produzidos por outros usuários. Outra contribuição de O'Callaghan et al. (2012) é a publicação de uma base de dados com comentários extraídos do YouTube, rotulados como *spam* ou comentários legítimos. Embora a base possua mais de 6 milhões de amostras, os rótulos foram atribuídos de

[3] *So long, video responses... Next up: better ways to connect.* Disponível em <https://youtube-creators.googleblog.com/2013/08/so-long-video-responsesnext-up-better.html>, acessado em 24/10/2016.

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.

Em um trabalho mais recente, Alsaleh et al. (2015) desenvolveram um mecanismo para detectar comentários *spam* em blogs que pode ser instalado como *plugin* nos navegadores. O sistema proposto pelos autores possui as etapas de: manipulação e inspeção do DOM (do inglês *Document Object Model*) das páginas Web para coletar as mensagens; pré-processamento do texto, aplicando lematização e *tokenização*; extração de atributos, tais como número de palavras e caracteres do comentário; e aplicação dos algoritmos de classificação.

Outra abordagem bastante comum é detectar e bloquear os *spammers* ao invés das mensagens. Benevenuto et al. (2009) e Sureka (2011) obtiveram bons resultados ao utilizar atributos baseados nos hábitos dos *spammers*, como o tempo médio entre as publicações e a quantidade de comentários idênticos em diferentes vídeos. No entanto, conforme apontado pelos autores, o comportamento dos *spammers* tende a mudar com o tempo, podendo tornar os atributos propostos obsoletos. Além disso, diversas aplicações permitem que usuários postem mensagens de forma anônima, inviabilizando a aplicação de técnicas de detecção de *spammers*.

A filtragem automática de *spam* também é relevante na execução de outras tarefas. Severyn et al. (2014) relataram que a detecção de opinião nos comentários do YouTube apresenta um desafio adicional, pois a plataforma possui muitas mensagens *spam* ou com conteúdo irrelevante. A primeira etapa do sistema de detecção de opinião proposta pelos autores é justamente a remoção de tais comentários.

Enquanto plataformas como o Twitter e o Facebook estão tomando providências para combater o *social spam*, movendo ações judiciais contra *spammers* e aplicando técnicas de aprendizado de máquina para detectar atividades suspeitas[4], o YouTube ainda fornece pouquíssimas funcionalidades para facilitar a moderação dos comentários em seus vídeos. Além de recursos limitados, como o reconhecimento de arte feita de caracteres ASCII [5], uma mensagem é removida se possuir uma palavra proibida pelo autor do vídeo (*blacklist*) ou se for sinalizada como *spam* por outros usuários (O'CALLAGHAN et al., 2012; RĂDULESCU; DINSOREANU; POTOLEA, 2014). Assim sendo, todos esses fatos evidenciam que, para manter viável a comunicação entre donos de canais e seus seguidores, há uma forte demanda por um filtro automático de *spam* no YouTube, antes que esse canal de comunicação seja totalmente corrompido pelos *spammers* e caia em desuso pelos usuários, que poderão buscar por melhores alternativas.

[4] *Keeping Facebook Activity Authentic*. Disponível em <https://www.facebook.com/business/news/authentic-activity-on-facebook>, acessado em 25/10/2016.

[5] *An update on YouTube comments*. Disponível em <https://youtube-creators.googleblog.com/2013/11/an-update-on-youtube-comments.html>, acessado em 25/10/2016.

O primeiro passo para realizar a filtragem automática de *spam* é a conversão do texto em uma representação compatível com os algoritmos de classificação. Para entender por que técnicas tradicionais de filtragem de *spam* podem apresentar desempenho insatisfatório quando aplicadas nos comentários do YouTube, é preciso primeiro entender como o computador interpreta essas amostras de texto.

2 Representação computacional

Como o texto não pode ser diretamente interpretado pelos algoritmos de classificação, é necessário um procedimento de indexação e representação das amostras em atributos numéricos (SEBASTIANI, 2002).

O primeiro passo é a segmentação (ou *tokenização*) das mensagens em termos (ou *tokens*), que são considerados a menor unidade de interesse de uma amostra e podem corresponder uma palavra ("carro"), um conjunto de palavras ("carro vermelho") ou um único símbolo ("c"), dependendo da aplicação. A regra mais simples de segmentação é utilizar o espaço em branco para dividir a amostra original, mas também é comum utilizar pontos, vírgulas, hífen e outros caracteres.

A regra de segmentação pode ser aplicada com diferentes valores de granularidade (*n-gram*), alterando quantas palavras compõem um termo. Para granularidade 2, por exemplo, modelo conhecido como *2-gram* ou bigrama, o termo é composto por duas palavras. Em seguida, define-se o vocabulário, que é o conjunto de todos os possíveis termos segmentados. Na Tabela 1, são apresentados dois vocabulários obtidos com granularidades diferentes para uma mesma amostra.

Tabela 1 – Diferentes representações de vocabulários para a amostra "*eu comprei um carro vermelho*".

Representação	Vocabulário
1-gram	{carro, comprei, eu, um, vermelho}
2-gram	{carro vermelho, comprei um, eu comprei, um carro}

Dada a segmentação das mensagens em termos e a construção do vocabulário, é possível definir a indexação e representação das amostras em atributos numéricos. Neste sentido, *bag of words* é uma das representações mais utilizadas na literatura de processamento de linguagem natural e categorização de texto (SEBASTIANI, 2002). Em sua forma mais tradicional, o modelo aplica a representação *1-gram* (unigrama), ou seja, cada termo é composto por uma palavra.

Considerando que cada mensagem ou documento de texto d é composto por um conjunto de termos $d = t_1, \ldots, t_n$, sendo que cada termo t_i corresponde a um elemento de um vocabulário conhecido, pode-se representar cada documento como um vetor $\vec{x} = \langle x_1, \ldots, x_n \rangle$, onde $x_1, \ldots, x_n$ são valores dos atributos associados aos termos $t_1, \ldots, t_n$. No caso mais simples, os atributos são binários, indicando $x_i = 1$, se o documento contém t_i, ou $x_i = 0$, caso contrário.

Um exemplo de representação *bag of words* é dado a seguir. Considere as amostras (1) "eu comprei um carro vermelho" e (2) "eu adquiri um carro velho". O vocabulário computado para estas amostras, é dado por {adquiri, carro, comprei, eu, um, velho, vermelho}. A representação de todas as amostras é feita no formato de matriz documento-termo (Tabela 2), onde o número de linhas (m) corresponde à quantidade de amostras e número de colunas (n) corresponde ao tamanho do vocabulário, ou quantidade de atributos. O valor dos atributos indica se determinado termo do vocabulário ocorre em determinada amostra.

Tabela 2 – Exemplo de representação com *bag of words*.

	adquiri	carro	comprei	eu	um	velho	vermelho
(1)	0	1	1	1	1	0	1
(2)	1	1	0	1	1	1	0

Alternativamente à representação binária, os atributos podem representar a frequência do termo (*term frequency – TF*), expressando quantas vezes cada termo ocorre em um dado documento, podendo ser transformada em escala logarítmica (WILBUR; KIM, 2009):

$$TF(t_i, d) = \log(1 + tf(t_i, d)), \tag{2.1}$$

onde $tf(t_i, d)$ é a quantidade de vezes que o termo t_i aparece no documento d.

Outra representação bastante comum é a obtida pela função *TF-IDF (term frequency-inverse document frequency)*, que intuitivamente, expressa que (i) quanto maior a frequência de um termo em um documento, mais representativo ele é, e (ii) quanto em mais documentos um termo ocorre, menos informativo ele é (WILBUR; KIM, 2009):

$$TF\text{-}IDF(t_i, d) = \log(1 + tf(t_i, d)) \log\left(\frac{|\mathcal{D}| + 1}{df(t_i) + 1}\right), \tag{2.2}$$

onde $tf(t_i, d)$ é a quantidade de vezes que o termo t_i aparece no documento d, $|\mathcal{D}|$ é a quantidade de documentos no conjunto de treinamento e $df(t_i)$ é o número de documentos no conjunto de treinamento em que t_i aparece.

Embora o *bag of words* forneça uma representação simplificada e eficaz das amostras de texto, o modelo traz uma série de limitações (GABRILOVICH; MARKOVITCH, 2005). O exemplo dado na Tabela 2 demonstra um problema bastante conhecido, a sinonímia. Os vocábulos "comprei" e "adquiri" têm o mesmo significado, porém são representados em colunas distintas.

Outro fenômeno bastante comum é a polissemia, em que dois vocábulos iguais apresentam significados diferentes dependendo do contexto. Dessa forma, dois termos com

significados diferentes são representados na mesma coluna. Por exemplo, nas sentenças "minha cabeça dói" e "ele é o cabeça da empresa", o termo cabeça possui significados distintos: parte do corpo e pessoa que dirige um grupo de pessoas.

As mensagens curtas e ruidosas presentes no YouTube e outras mídias sociais também geram desafios de representação adicionais. Considere as seguintes amostras em inglês (1) *"dis movie is awsm"*, (2) *"your new car is awesome"* e (3) *"I <3 u"*. A Tabela 3 contém a representação *bag of words* dessas amostras.

Tabela 3 – Representação das amostras (1) *"dis movie is awsm"*, (2) *"your new car is awesome"* e (3) *"I <3 u"*.

	<3	*awesome*	*awsm*	*car*	*dis*	*I*	*is*	*movie*	*new*	*u*	*your*
(1)	0	0	1	0	1	0	1	1	0	0	0
(2)	0	1	0	1	0	0	1	0	1	0	1
(3)	1	0	0	0	0	1	0	0	0	1	0

No exemplo dado, é possível perceber que os vocábulos *"awesome"* e *"awsm"*, apesar de serem a mesma palavra, estão escritas de formas diferentes, sendo a segunda uma abreviação popular da primeira. Dessa forma, o modelo acaba criando dois atributos completamente diferentes, podendo confundir os métodos de classificação, que não observam a relação existente entre essas palavras.

O vocábulo *"dis"* também apresenta o mesmo problema, visto que está escrito da maneira incorreta. Caso existisse no vocabulário a palavra *"this"*, os dois vocábulos também seriam representados em colunas diferentes. A terceira amostra introduz um símbolo (*"<3"*) e uma letra (*"u"*), representando a frase *"I love you"*, e que, mais um vez, não apresentariam relação com os vocábulos *"love"* e *"you"*, caso eles estivessem presentes.

Além destes problemas, as amostras de texto curtas e ruidosas podem tornar o vocabulário muito grande e deixar os vetores de atributos altamente esparsos, o que pode prejudicar os algoritmos de classificação, exigindo maiores recursos computacionais e aumentando excessivamente o tempo de treinamento.

Todos esses problemas de representação podem ser amenizados através do emprego de técnicas de processamento e expansão textual, que podem tornar as amostras de texto mais informativas e adequadas aos métodos de aprendizado de máquina (ALMEIDA et al., 2016).

2.1 Técnicas de processamento e expansão textual

Visto que modelo *bag of words* pode conduzir a resultados insatisfatórios quando aplicado na representação de mensagens curtas e repletas de erros de digitação, gírias,

símbolos e abreviações, é recomendado o uso de técnicas de processamento de linguagem natural para normalizar e enriquecer a informação semântica das amostras de texto. Dentre tais técnicas estão a normalização léxica, a geração de conceitos por indexação semântica e a desambiguação de conceitos (*word-sense disambiguation*) (ALMEIDA et al., 2016).

A normalização léxica está relacionada à correção ortográfica e consiste em substituir variantes léxicas e expressões normalmente ofuscadas para sua forma canônica. Por exemplo, os termos *"dis"* e *"awsm"* seriam substituídos pelas palavras em inglês *"this"* e *"awesome"*. Verbos como *"is"* e *"going"*, por outro lado, são reduzidos para seus radicais, *"be"* e *"go"*, respectivamente. O benefício da normalização léxica é justamente aproximar termos equivalentes e reduzir o número de elementos do vocabulário.

Para realizar a normalização léxica de mensagens ruidosas, é possível combinar o uso de diferentes dicionários, como por exemplo, um de inglês[1], usado para consultar se uma determinada palavra existe no idioma inglês e reduzir para seu radical quando possível; e um de *Lingo*[2], nome dado ao conjunto de gírias e abreviações comumente usados na Internet, que é utilizado para traduzir um termo *Lingo*, por palavras da língua inglesa. Caso não haja uma tradução para o termo de entrada, o termo original pode ser mantido.

A geração de conceitos por indexação semântica envolve analisar termos isolados e encontrar sinônimos, de acordo com os conceitos que essas palavras representam. Um dos conceitos associados à palavra *"car"*, por exemplo, é *"motor vehicle with four wheels"*, que, por sua vez, é formado pelo seguinte conjunto: {*car*, *auto*, *automobile*, *machine*, *motorcar*}. Sendo assim, com esta técnica é possível expandir a amostra original, adicionando os sinônimos encontrados em grandes e modernos dicionários semânticos, como o repositório *BabelNet* (NAVIGLI; PONZETTO, 2012a).

A geração de conceitos, no entanto, pode retornar um grande número de conceitos para cada palavra da mensagem original. Isso pode agregar ruído à mensagem e prejudicar o desempenho da representação de texto, ao invés de melhorá-lo. A técnica de desambiguação é aplicada em combinação com a geração de conceitos para remover ruídos e tornar a amostra mais significativa. O processo de desambiguação consiste em construir um grafo com as informações semânticas da amostra e selecionar apenas os termos mais próximos. Dessa forma, somente as palavras mais relevantes são mantidas, de acordo com o contexto da mensagem (NAVIGLI; PONZETTO, 2012b).

[1] *FreeLing*. Disponível em <http://devel.cpl.upc.edu/freeling/>, acessado em 04/11/2016.

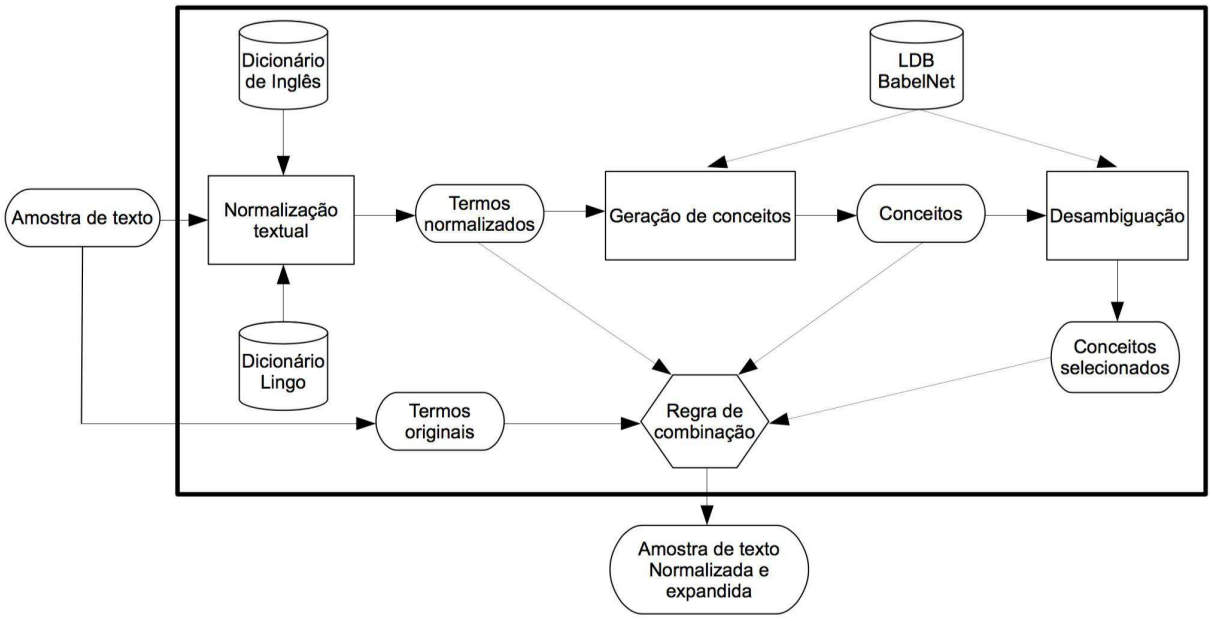
[2] *NoSlang Lingo dictionary*. Disponível em <http://www.noslang.com/dictionary/>, acessado em 04/11/2016.

2.2 TextExpansion

De forma a combinar as técnicas de normalização léxica, de geração de conceitos por indexação semântica e de desambiguação de conceitos, Silva (2016), Almeida et al. (2016) propuseram um sistema para aumentar a informação semântica das amostras de texto e, consequentemente, melhorar o aprendizado e a qualidade das predições, o `TextExpansion`[3].

O `TextExpansion` permite expandir a amostra original, gerando novas amostras normalizadas e expandidas. Em seguida, dada uma regra de combinação predefinida, as amostras são unidas em uma mensagem de texto final, que pode ser aplicada nos métodos de aprendizado de máquina no lugar da amostra original. A Figura 3 ilustra o processo realizado pela ferramenta `TextExpansion`.

Figura 3 – A amostra original é processada pelos dicionários semânticos e técnicas de detecção de contexto. Cada etapa cria uma nova amostra, normalizada ou expandida. Posteriormente, dada uma regra de combinação, as amostras são unidas em uma mensagem de texto final com o mesmo conteúdo semântico da amostra original.



Fonte: Silva (2016).

As possíveis regras de combinação são formadas por quatro parâmetros, que são basicamente respostas para as seguintes perguntas: (1) *O método deve manter os termos originais?*, (2) *O método deve aplicar a normalização léxica?*, (3) *O método deve gerar conceitos?* e (4) *O método deve aplicar a desambiguação?*. Como cada parâmetro é binário, é possível gerar 11 combinações, incluindo a amostra original. A Figura 4 apresenta todas

[3] `TextExpansion`. Disponível em <http://lasid.sor.ufscar.br/expansion/>, acessado em 04/11/2016.

as possíveis regras de combinação que podem ser utilizadas na ferramenta `TextExpansion`. Na Regra 1, a base de dados original é mantida. Porém nas demais, pelo menos uma etapa de normalização ou expansão é aplicada. Como a etapa de *desambiguação* resulta em um subconjunto da etapa de *geração de conceitos*, não é possível combinar as duas saídas.

Tabela 4 – Possíveis regras de combinação utilizadas no `TextExpansion`.

	Manter o termos originais?	Aplicar normalização léxica?	Aplicar geração de conceitos?	Aplicar desambiguação?
Regra 1†	✓			
Regra 2	✓		✓	
Regra 3	✓			✓
Regra 4	✓	✓		
Regra 5	✓	✓	✓	
Regra 6	✓	✓		✓
Regra 7			✓	
Regra 8				✓
Regra 9		✓		
Regra 10		✓	✓	
Regra 11		✓		✓

† base de dados original

A Tabela 5 apresenta um exemplo de normalização e expansão utilizando a amostra *"Psy - two Billions of views! WTF? Plz, u shud c my nu song at dis lnk..."*. Enquanto a etapa de *normalização* substitui as gírias e abreviações por suas palavras correspondentes em inglês, a etapa de *geração de conceitos* insere todos os conceitos encontrados da amostra original. A etapa de *desambiguação* mantém apenas os conceitos semanticamente relevantes à amostra original. Por fim, a regra de combinação escolhida (Regra 11), expande a amostra original, incluindo apenas os termos resultantes das etapas de *normalização* e *desambiguação*.

Para obter informações detalhadas sobre o `TextExpansion`, consulte Almeida et al. (2016).

Após empregar as técnicas de normalização e indexação semântica, espera-se que seja possível eliminar problemas como a sinonímia e a polissemia, resultando em uma amostra mais informativa e adequada aos métodos de aprendizado de máquina. No entanto, nota-se que nesse contexto, a geração de conceitos pode preencher a amostra com muito ruído. Isso pode ser decorrente dos termos pesquisados ou do domínio no qual a mensagem está inserida. Sendo assim, não existe um consenso de qual regra de combinação é a melhor para todos os cenários, podendo variar conforme o domínio ao qual a expansão é aplicada e quais métodos de aprendizado de máquina são empregados.

Tabela 5 – Exemplo de normalização e expansão da amostra *"Psy - two Billions of views! WTF? Plz, u shud c my nu song at dis lnk..."*.

Termos originais	*Psy - two Billions of views! WTF? Plz, u shud c my nu song at dis lnk...*
Normalização	*Psy two billion of view what the fuck please you should see my new song at this link*
Geração de conceitos	*Psy deuce ii two 1000000000000 billion gazillion jillion million one_million_million one_thousand_million trillion zillion of eyeshot horizon opinion panorama perspective persuasion position prospect purview scene sentiment sight survey thought view vista what the fuck fucking nookie nooky piece_of_ass piece_of_tail roll_in_the_hay screw screwing shag shtup please you should see my new birdsong call ming ming_dynasty song song_dynasty strain sung sung_dynasty vocal at this connectedness connection connexion contact data_link inter-group_communication liaison link linkup nexus radio_link tie tie-in*
Desambiguação	*Psy two billion of opinion what the sleep_together please you should see my new Sung at this associate*
Regra de combinação 11	*Psy two billion of view opinion what the fuck sleep_together please you should see my new song Sung at this link associate*

3 Métodos de classificação *online*

Como a filtragem de *spam* trata-se de um problema de classificação adversária, em que os *spammers* constantemente tentam evadir a detecção, métodos tradicionais de aprendizado de máquina, que não possuem aprendizado incremental e geram modelos de predição estáticos, podem não ser apropriados para serem aplicados em filtros reais de *spam*. O comportamento dos usuários também evolui continuamente, exigindo que o modelo de predição seja dinâmico e que o processo de aprendizagem seja realizado de maneira *online* e incremental (BRATKO et al., 2006).

O aprendizado *online* representa uma importante família de algoritmos eficientes e escaláveis de aprendizado de máquina. De maneira geral, tais algoritmos são rápidos e robustos, tornando-os adequados a uma grande variedade de aplicações. Existem diversos métodos de classificação na literatura que permitem o aprendizado *online*. Dentre os mais tradicionalmente empregados, destacam-se:

- Perceptron (ROSENBLATT, 1958);
- Algoritmo de Margem Máxima *Online* Relaxada (ROMMA – *Relaxed Online Maximum Margin Algorithm*) (LI; LONG, 2002);
- Algoritmo de Margem Larga Aproximada (ALMA – *Approximate Large Margin Algorithm*) (GENTILE, 2001);
- Passivo-Agressivo (PA – *Passive-Aggressive*) (CRAMMER et al., 2006); e
- Gradiente Descendente *Online* (OGD – *Online Gradient Descent*) (ZINKEVICH, 2003; HOI; WANG; ZHAO, 2014).

Esses algoritmos compartilham algumas características em comum, como a realização da classificação binária (embora existam variantes multiclasses) e o fato de possuírem hipóteses de predição lineares, dividindo o espaço de atributos em duas metades. Além disso, os métodos acima são baseados na primeira ordem do gradiente, exigindo menos recursos computacionais do que abordagens baseadas na segunda ordem.

No escopo da classificação de texto, o *Naïve* Bayes (LANGLEY; IBA; THOMPSON, 1992; MANNING; RAGHAVAN; SCHÜTZE, 2008) é outro método que se destaca, sendo muito utilizado como *baseline*. Além destes, o `MDLText` (SILVA; ALMEIDA; YAMAKAMI, 2017), um novo método de classificação de texto baseado no princípio da descrição mais simples (MDL – *Minimum Description Length*) (RISSANEN, 1978), oferece características promissoras para ser aplicado ao problema de detecção de *spam*. O método é eficiente,

dinâmico e escalável, podendo ser aplicado em cenários reais de classificação *online* de texto.

3.1 Classificação *online*

Na classificação *online*, cada amostra é apresentada individualmente e de maneira sequencial. A cada passo s, o classificador recebe uma amostra de exemplo $\vec{x}_s \in \mathcal{X}$ e realiza a predição:

$$\hat{y}_s = \begin{cases} 1, & \text{se } \vec{w}^T \vec{x}_s \geq 0 \\ -1, & \text{caso contrário} \end{cases}, \tag{3.1}$$

onde $\vec{w}$ é o vetor de pesos utilizados pelo classificador e $\hat{y}_s \in \mathcal{Y}$, sendo $\mathcal{Y} = \{-1, 1\}$ para uma classificação binária. Após realizar a predição, o verdadeiro rótulo $y_s \in \mathcal{Y}$ é revelado, permitindo que o classificador compute o erro ℓ baseado em algum critério para medir a diferença entre a predição $\hat{y}_s$ e o rótulo verdadeiro y_s. Com o resultado do erro, o classificador finalmente decide quando e como atualizar o modelo de classificação ao final de cada passo.

Os diferentes métodos de classificação *online* distinguem-se em como é calculado o erro ℓ e como o modelo de classificação é atualizado a cada passo. Nas seções seguintes, são brevemente apresentados os métodos de aprendizado *online* avaliados neste trabalho e vastamente empregados na literatura.

Para obter mais detalhes sobre classificação *online*, consulte Li e Long (2002), Crammer et al. (2006), Hoi, Wang e Zhao (2014).

3.2 Perceptron

O Perceptron (ROSENBLATT, 1958) é uma das primeiras e mais simples abordagens que permitem a classificação *online*. Desenvolvido para replicar o comportamento do cérebro humano, o Perceptron é considerado um dos precursores das redes neurais artificiais. O método é relativamente simples e não possui parâmetros.

A função de erro utilizada pelo Perceptron é conhecida como *0–1 loss* e é calculada por:

$$\ell = \begin{cases} 0, & \text{se } y_s = \hat{y}_s \\ 1, & \text{se } y_s \neq \hat{y}_s \end{cases}. \tag{3.2}$$

Quando o modelo tiver cometido um erro ($\ell > 0$), o Perceptron atualiza os pesos da seguinte forma:

$$\vec{w} \coloneqq \vec{w} + y_s \vec{x}_s. \tag{3.3}$$

Para saber mais sobre o método Perceptron, consulte Rosenblatt (1958), Li e Long (2002).

3.3 ROMMA

O Algoritmo de Margem Máxima *Online* Relaxada (ROMMA – *Relaxed Online Maximum Margin Algorithm*) (LI; LONG, 2002) é um método baseado no critério da margem máxima do hiperplano e propõe um treinamento rápido e eficiente, com complexidade e simplicidade similares ao Perceptron. O ROMMA pode ser visto como uma aproximação do problema de margem máxima que evita operações computacionalmente custosas, como é o caso da abordagem tradicional das Máquinas de Vetores de Suporte (SVM – *Support Vector Machines*) (CORTES; VAPNIK, 1995).

A função de erro do ROMMA é a mesma do método Perceptron, *0–1 loss*, expressada na Equação (3.2). Alternativamente, existe a versão *aggressive*-ROMMA (LI; LONG, 2002), que utiliza a função de erro conhecida como *hinge loss*, calculada por:

$$\ell = max(0, 1 - y_s \vec{w}^T \vec{x}_s). \tag{3.4}$$

O *aggressive*-ROMMA considera que o modelo cometeu um erro não apenas quando um rótulo errado é atribuído ($y_s \neq \hat{y}_s$), mas sempre que $y_s \vec{w}^T \vec{x}_s < 1$.

Quando o modelo tiver cometido um erro ($\ell > 0$), ambos, ROMMA e *aggressive*-ROMMA, atualizam os pesos da seguinte forma:

$$\vec{w} \coloneqq \frac{(\|\vec{x}_s\|\|\vec{w}\|)^2 - y_s \vec{w}^T \vec{x}_s}{(\|\vec{x}_s\|\|\vec{w}\|)^2 - (\vec{w}^T \vec{x}_s)^2} \vec{w} + \frac{\|\vec{w}\|^2 (y_s - \vec{w}^T \vec{x}_s)}{(\|\vec{x}_s\|\|\vec{w}\|)^2 - (\vec{w}^T \vec{x}_s)^2} \vec{x}_s. \tag{3.5}$$

Para maiores informações sobre o método ROMMA, consulte Li e Long (2002).

3.4 ALMA

Outro método baseado no critério da margem máxima do hiperplano é o Algoritmo de Margem Larga Aproximada (ALMA – *Approximate Large Margin Algorithm*) (GENTILE, 2001), cuja principal diferença é permitir que a margem seja mensurada por meio de uma norma genérica p (*p-norm*), ao passo que algoritmos como o ROMMA foram desenvolvidos para aplicar especificamente a norma euclidiana.

Além do parâmetro norma p, definido com o valor padrão 2 e usado para medir a distância entre o hiperplano e as margens, o ALMA também possui o grau de aproximação para a margem ótima (α), definido com o valor padrão 0,9. Outros dois parâmetros B e C possuem os valores ótimos $1/\alpha$ e $\sqrt{2}$, respectivamente, mas podem ser ajustados de acordo com o problema.

A função de erro do ALMA é calculada por:

$$\gamma = \frac{B\sqrt{p-1}}{\sqrt{k}}, \tag{3.6}$$

$$\ell = max\left(0, (1-\alpha)\gamma - y_s \vec{w}^T \vec{x}_s\right), \tag{3.7}$$

sendo k o número de erros cometidos (ou correções realizadas), iniciado com o valor 1.

Quando o modelo tiver cometido um erro ($\ell > 0$), o ALMA atualiza os pesos da seguinte forma:

$$\vec{w}' = \vec{w} + \frac{C}{\sqrt{p-1}\sqrt{k}} y_s \vec{x}_s, \tag{3.8}$$

$$\vec{w} \coloneqq \vec{w}' / max(1, \|\vec{w}'\|), \tag{3.9}$$

$$k \coloneqq k + 1. \tag{3.10}$$

Para maiores detalhes sobre o método ALMA, consulte Gentile (2001).

3.5 Passivo-Agressivo

O Passivo-Agressivo (PA – *Passive-Aggressive*) (CRAMMER et al., 2006) é outro método desenvolvido sobre a noção de margens do hiperplano de separação das classes. A otimização do PA consiste em manter o novo modelo o mais próximo possível do modelo atual, no entanto, forçando a atualização das margens o tanto quanto necessário para atingir $\ell = 0$ para a amostra de exemplo atual, $\vec{x}_s$. Ou seja, em caso de $\ell = 0$ os pesos *passivamente* se mantêm inalterados, enquanto que em caso de $\ell > 0$, o modelo *agressivamente* atualiza as margens para contemplar a amostra de exemplo atual e anular o erro.

A função de erro do PA é a *hinge loss*, definida na Equação (3.4). Quando o modelo tiver cometido um erro ($\ell > 0$), o PA atualiza os pesos da seguinte forma:

$$\vec{w} \coloneqq \vec{w} + \frac{\ell}{\|\vec{x}_s\|^2} y_s \vec{x}_s. \tag{3.11}$$

Como a estratégia de atualização do PA pode ser bastante agressiva na presença de ruídos, foram propostas duas variantes do algoritmo original que empregam um *parâmetro de agressividade* ($C > 0$), cujo intuito é equilibrar a margem desejada e a proximidade com o modelo atual, melhorando a habilidade do algoritmo lidar com ruídos. Estes dois métodos foram nomeados PA-I e PA-II.

O PA-I atualiza os pesos da seguinte forma:

$$\vec{w} \coloneqq \vec{w} + min\left(C, \frac{\ell}{\|\vec{x}_s\|^2}\right) y_s \vec{x}_s. \tag{3.12}$$

O PA-II atualiza os pesos da seguinte forma:

$$\vec{w} \coloneqq \vec{w} + \frac{\ell}{\|\vec{x}_s\|^2 + 1/2C} y_s \vec{x}_s. \tag{3.13}$$

Para saber mais sobre o método PA, consulte Crammer et al. (2006).

3.6 OGD

O Gradiente Descendente *Online* (OGD – *Online Gradient Descent*) (ZINKEVICH, 2003; HOI; WANG; ZHAO, 2014) é outro método largamente utilizado, que envolve a otimização convexa e a atualização do gradiente descendente por meio de diferentes funções de erro.

O método emprega uma taxa de aprendizagem η_s, definida por:

$$\eta_s = \frac{C}{\sqrt{s}}, \tag{3.14}$$

onde $C > 0$ é a constante de aprendizagem, informada como parâmetro, e s é o número de passos realizados.

O OGD oferece quatro diferentes funções de erro para a atualização do gradiente (HOI; WANG; ZHAO, 2014). Além das funções *0–1 loss* e *hinge loss*, definidas anteriormente nas Equações (3.2) e (3.4), respectivamente, o OGD propõe a utilização da função de erro logístico (*logistic loss*), definida por:

$$\ell = log(1 + exp(-y_s \vec{w}^T \vec{x}_s)), \tag{3.15}$$

ou da função de erro quadrático (*square loss*), definida por:

$$\ell = 0{,}5(\vec{w}^T \vec{x}_s - y_s)^2. \tag{3.16}$$

Quando o modelo tiver cometido um erro ($\ell > 0$), o OGD também propõe diferentes formas de atualizar os pesos. Quando a função de erro *0–1 loss* ou a função de erro *hinge loss* são empregadas, o OGD atualiza os pesos da seguinte forma:

$$\vec{w} := \vec{w} + \eta_s y_s \vec{x}_s. \tag{3.17}$$

Quando a função de erro logístico é empregada, os pesos são atualizados da seguinte forma:

$$\vec{w} := \vec{w} + \eta_s y_s \vec{x}_s \frac{1}{1 + exp(y_s \vec{w}^T \vec{x}_s)}. \tag{3.18}$$

E por fim, quando a função de erro quadrático é empregada, o OGD atualiza os pesos da seguinte maneira:

$$\vec{w} := \vec{w} + \eta_s (y_s - \vec{w}^T \vec{x}_s) \vec{x}_s. \tag{3.19}$$

Para obter mais informações sobre o método OGD, consulte Zinkevich (2003), Hoi, Wang e Zhao (2014).

3.7 *Naïve* Bayes

O *Naïve* Bayes (LANGLEY; IBA; THOMPSON, 1992; MANNING; RAGHAVAN; SCHÜTZE, 2008) é um método de classificação probabilístico baseado no teorema de Bayes, cujo treinamento é considerado rápido, eficiente e robusto, podendo ser aplicado a uma grande variedade de problemas. O *Naïve* Bayes é excepcionalmente popular na tarefa de categorização de texto e detecção de *spam*, sendo bastante utilizado como *baseline*.

O método analisa cada atributo individualmente e coleta probabilidades atributo–classe, assumindo que o valor dos atributos contribuem de forma independente para a probabilidade de uma determinada amostra pertencer a uma classe.

Seja a amostra de exemplo $\vec{x}$, com n atributos, representada pelo vetor $\vec{x} = \langle x_1, \ldots, x_n \rangle$, a probabilidade desta amostra pertencer a uma determinada classe $c \in \mathcal{Y}$ é dada por:

$$P(c \mid \vec{x}) \propto P(c)\ P(\vec{x} \mid c), \tag{3.20}$$

onde $P(c)$ é a probabilidade a *priori* de uma amostra qualquer ocorrer na classe c e $P(\vec{x} \mid c)$ é a probabilidade condicional da amostra $\vec{x}$ ocorrer na classe c.

Na abordagem mais tradicional do classificador *Naïve* Bayes, a probabilidade $P(\vec{x} \mid c)$ é calculada por:

$$P(\vec{x} \mid c) = \prod_{i=1}^{n} P(x_i \mid c), \tag{3.21}$$

onde $P(x_i \mid c)$ é a probabilidade condicional do atributo x_i ocorrer em uma amostra da classe c. Sendo assim, para realizar a predição, o *Naïve* Bayes escolhe a classe com maior probabilidade a *posteriori* (MAP – *maximum a posteriori*):

$$\hat{y}_s = c_{map} = \arg\max_{c \in \mathcal{Y}} P(c \mid \vec{x}_s). \tag{3.22}$$

Como o processo de aprendizagem do *Naïve* Bayes pode ser realizado de maneira incremental, o método é naturalmente *online*, bastando que as probabilidades $P(c)$ e $P(x_i \mid c)$ sejam atualizadas a cada nova amostra de exemplo.

Duas variações bastante empregadas na categorização de texto são os métodos *Naïve* Bayes Multinomial (M.NB) e *Naïve* Bayes Bernoulli (B.NB) (ALMEIDA; YAMAKAMI; ALMEIDA, 2011).

Seja o documento $d = \{t_1, \ldots, t_n\}$ representado pelo vetor $\vec{x} = \langle x_1, \ldots, x_n \rangle$, onde cada x_i corresponde à frequência de t_i em d, o *Naïve* Bayes Multinomial calcula a probabilidade $P(\vec{x} \mid c)$ por:

$$P(\vec{x} \mid c) = P(|d|)\ |d|!\ \prod_{i=1}^{n} \frac{P(t_i \mid c)^{x_i}}{x_i!}. \tag{3.23}$$

O *Naïve* Bayes Bernoulli, por outro lado, emprega atributos binários e leva em consideração não só a presença de um termo, como também sua ausência. Dado o documento $d = \{t_1, \ldots, t_n\}$ representado pelo vetor $\vec{x} = \langle x_1, \ldots, x_n \rangle$, onde $x_1, \ldots, x_n$ são valores binários, a probabilidade $P(\vec{x} \mid c)$ é calculada por:

$$P(\vec{x} \mid c) = \prod_{i=1}^{n} P(t_i \mid c)^{x_i}\ (1 - P(t_i \mid c))^{(1-x_i)}. \tag{3.24}$$

Para maiores detalhes sobre o método *Naïve* Bayes, consulte Langley, Iba e Thompson (1992), Manning, Raghavan e Schütze (2008), Almeida, Yamakami e Almeida (2011).

3.8 MDL

O princípio da descrição mais simples (MDL – *Minimum Description Length*) (RISSANEN, 1978) 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 a

descrição mais compacta dos dados. Para isso, o método busca um bom equilíbrio entre a complexidade do modelo e a sua capacidade de representação (ALMEIDA; YAMAKAMI, 2012).

Matematicamente, dado um conjunto de potenciais modelos $\{m_1, \ldots, m_n\} \in M$, o MDL seleciona aquele, tal que:

$$m_{mdl} = \arg\min_{m \in M} L(m) + L(x \mid m), \tag{3.25}$$

onde $L(m)$ é o tamanho da descrição do modelo m, representando a sua complexidade, e $L(x \mid m)$ é o tamanho da descrição do dado x quando codificado com m, representando quão bem m descreve x.

No contexto da classificação de texto, os modelos são as possíveis classes do problema. Dado um documento de texto d com rótulo desconhecido, deseja-se associar a este documento o rótulo da classe c que tiver o menor tamanho de descrição em relação a ele, ou seja, o menor $L(d \mid c)$. O tamanho da descrição das potenciais classes $L(c)$ pode ser ignorado, uma vez que existe um risco de tornar-se arbitrário e difícil de determinar (SILVA; ALMEIDA; YAMAKAMI, 2017). Assim, para um determinado documento d, o MDL prediz a sua classe com base na seguinte equação:

$$\hat{y} = \arg\min_{c \in \mathcal{Y}} L(d \mid c). \tag{3.26}$$

O tamanho da descrição do documento d quando codificado com c pode ser calculado pela soma dos tamanhos da descrição de todos os termos t que aparecem em d:

$$L(d \mid c) = \sum_{i=1}^{|d|} L(t_i \mid c), \tag{3.27}$$

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

O tamanho da descrição de cada termo t_i quando codificado com c, por sua vez, é calculado por:

$$L(t_i \mid c) = \lceil -\log_2 P(t_i \mid c) \rceil, \tag{3.28}$$

onde $P(t_i \mid c)$ é a probabilidade condicional do termo t_i dados os documentos da classe c, calculada com base no peso *TF-IDF* (Equação (2.2)) deste termo para todos os documentos do conjunto de treinamento $\mathcal{D}$, conforme definido na seguinte equação:

$$P(t_i \mid c) = \frac{n_c(t_i) + \frac{1}{|\Omega|}}{\hat{n}_c + 1}, \tag{3.29}$$

onde $n_c(t_i) = \sum_{\forall d} \hat{w}(t_i, d \mid c)$, ou seja, é igual a soma dos pesos normalizados do termo t_i para todos os documentos d da classe c, obtidos por *TF-IDF*, seguido de normalização euclidiana:

$$\hat{w}(t_i, d) = \frac{TF\text{-}IDF(t_i, d)}{\|TF\text{-}IDF(:, d)\|_2}. \tag{3.30}$$

De forma semelhante, $\hat{n}_c$ corresponde à soma de $n_c(t_i)$ para todos os termos que aparecem em documentos da classe c e, por fim, o parâmetro $|\Omega|$ é usado para preservar uma porção do tamanho da descrição para termos que nunca apareceram em $\mathcal{D}$ e que possuem, portanto, $n_c(t_i) = 0$. Neste trabalho, foi adotado $|\Omega| = 2^{10}$, conforme originalmente proposto por (SILVA; ALMEIDA; YAMAKAMI, 2017).

Para saber mais sobre o método MDL, consulte Rissanen (1978), Almeida e Yamakami (2012), Silva, Almeida e Yamakami (2017).

3.9 MDLText

O `MDLText` (SILVA; YAMAKAMI; ALMEIDA, 2015; SILVA; ALMEIDA; YAMAKAMI, 2017) expande a capacidade preditiva do MDL, adicionando dois componentes ao cálculo do modelo: (1) uma função de penalidade $\hat{S}(d, c)$ para cada classe e (2) uma função de penalidade $K(t_i)$ para cada termo. Assim, a Equação (3.27) é adaptada para:

$$L(d \mid c) = \hat{S}(d, c) \sum_{i=1}^{|d|} L(t_i \mid c) K(t_i). \tag{3.31}$$

O primeiro componente adicionado, a função de penalidade $\hat{S}(d, c) \geq 1$, tem a finalidade de penalizar a classe com menor similaridade em relação ao documento. Quanto menor a similaridade, maior é o valor obtido em $\hat{S}(d, c)$ e, portanto, maior é o tamanho da descrição do documento d dada a classe c. A função $\hat{S}(d, c)$ é definida por:

$$\hat{S}(d, c) = -\log_2 \left(\frac{1}{2} S(d, \overline{c}) \right), \tag{3.32}$$

onde $0 \leq S(d, \overline{c}) \leq 1$ é a similaridade de cosseno (*cosine similarity*) entre o documento d e um vetor protótipo da classe $\overline{c}$, sendo que valores próximos a 1 indicam maior similaridade.

Para calcular a similaridade de cosseno, o `MDLText` aplica a seguinte equação:

$$S(d, \overline{c}) = \frac{\sum_{i=1}^{|d|} \hat{w}(t_i, d \mid c) \overline{c}(t_i)}{\|\hat{w}(:, d)\|_2 \|\overline{c}\|_2}, \tag{3.33}$$

onde $\hat{w}$ são os pesos normalizados (Equação (3.30)) e o vetor protótipo $\overline{c}$ é formado pela média dos pesos dos termos. Portanto, para cada termo t_i, $\overline{c}(t_i)$ é:

$$\overline{c}(t_i) = \frac{n_c(t_i)}{|\mathcal{D}_c|}, \tag{3.34}$$

onde $|\mathcal{D}_c|$ é o número de documentos de treinamento que pertencem à classe c.

Já o segundo componente, a função de penalidade $K(t_i)$, tem a finalidade de fazer com que termos mais representativos contribuam mais com $L(d \mid c)$ e, portanto, exerçam maior influência na separação entre as classes. A função $K(t_i)$ é definida por:

$$K(t_i) = \frac{1}{(1+\alpha) - F(t_i)}, \tag{3.35}$$

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 igual a 0. Neste trabalho, foi adotado $\alpha = 10^{-3}$, conforme originalmente proposto por (SILVA; ALMEIDA; YAMAKAMI, 2017).

Para calcular $F(t_i)$, pode ser usada qualquer função que retorne uma pontuação de contribuição do termo entre 0 e 1, sendo que o `MDLText` propõe o uso da técnica de fatores de confidência (CF – *Confidence Factors*), cujo intuito é reduzir o impacto de ruídos introduzidos por termos com baixa significância (SILVA; YAMAKAMI; ALMEIDA, 2015; SILVA; ALMEIDA; YAMAKAMI, 2017).

Para obter informações detalhadas sobre o método `MDLText`, consulte Almeida e Yamakami (2012), Silva, Yamakami e Almeida (2015), Silva, Almeida e Yamakami (2017).

4 Experimentos e resultados

Com a finalidade de comparar o desempenho de métodos tradicionais de classificação *online*, assim como avaliar o emprego de técnicas de processamento de linguagem natural e encontrar configurações eficazes para realizar a filtragem automática de comentários postados no YouTube, este capítulo apresenta a proposta de pesquisa, a metodologia experimental empregada e, por fim, os resultados obtidos pelos experimentos.

4.1 Proposta

Identificados os dois maiores problemas para a filtragem automática de *spam* nos comentários do YouTube: (1) as mensagens costumam ser muito curtas e mal redigidas; e (2) o problema de classificação é naturalmente *online*; foram realizados experimentos para validar a hipótese de que a aplicação de técnicas de normalização léxica e indexação semântica, combinadas com a característica dinâmica e incremental de métodos de classificação *online*, pode obter resultados promissores e ser efetivamente empregada na filtragem automática de comentários indesejados postados no YouTube.

Para tornar os experimentos reprodutíveis, na próxima seção é apresentada a metodologia experimental empregada.

4.2 Metodologia experimental

Devido à escassez de bases públicas sobre o tema desta pesquisa, a primeira etapa consistiu em rotular manualmente amostras reais que foram extraídas do YouTube por meio de sua API[1] no primeiro semestre de 2015. Como resultado desse processo, foram criadas e disponibilizadas publicamente cinco bases de dados[2], que representam cinco dentre os dez vídeos com maior número de visualizações do YouTube no período da coleta.

O processo de rotulação das bases foi realizado através de uma ferramenta colaborativa chamada *Labeling*[3], desenvolvida para essa finalidade durante o período do mestrado. Cada amostra representa um comentário postado em língua inglesa, no espaço de comentários de um vídeo, rotulado como *spam* ou mensagem legítima (*ham*). As amostras também possuem metadados associados, como o nome do autor e a data de publicação, que foram preservados.

[1] *YouTube Data API*. Disponível em <https://developers.google.com/youtube/v3>, acessado em 05/11/2016.

[2] *YouTube Spam Collection*. Disponível em <http://dcomp.sor.ufscar.br/talmeida/youtubespamcollection>, acessado em 05/11/2016.

[3] *Labeling*. Disponível em <http://lasid.sor.ufscar.br/ml-tools/>, acessado em 05/11/2016.

Na Tabela 6, são apresentados os nomes das bases de dados coletadas e utilizadas nos experimentos, o *ID* do vídeo no YouTube, a quantidade de amostras em cada classe e o total de amostras por base.

Tabela 6 – Bases de dados criadas e utilizadas nos experimentos.

Base de dados	YouTube ID	# *Spam*	# *Ham*	Total
Eminem	uelHwf8o7_U	245	203	448
Katy Perry	CevxZvSJLk8	175	175	350
LMFAO	KQ6zr6kCPj8	236	202	438
Psy	9bZkp7q19f0	175	175	350
Shakira	pRpeEdMmmQ0	174	196	370

Em seguida, foi empregada a ferramenta `TextExpansion` para gerar novas bases expandidas, uma para cada possível combinação de parâmetros, apresentadas na Tabela 4. Em todos os experimentos, tanto com as bases de dados originais, quanto com as bases expandidas, os textos foram convertidos para letras minúsculas e segmentados, utilizando quaisquer caracteres não alfanuméricos (exceto *underline*) como delimitadores. Foi adotada a representação *bag of words*, *1-gram*, sendo utilizada a frequência dos termos (TF) como atributos, que são então convertidos para TF-IDF ao longo do treinamento. Essa conversão é feita de forma incremental para reproduzir um cenário real e dinâmico, visto que o TF-IDF carrega informações do conjunto de treinamento na sua representação, como a quantidade de documentos no conjunto de treinamento em que cada termo aparece. A escolha do TF-IDF baseia-se no fato de que essa representação foi a que mais beneficiou os métodos mais tradicionais de categorização de texto, de acordo com **??**). Para o B.NB, foi utilizada a representação binária, pois é intrínseca para este método.

A Tabela 7 apresenta as estatísticas básicas sobre cada base de dados gerada através da ferramenta `TextExpansion`, de acordo com cada regra de combinação utilizada, onde $|d|$ corresponde ao número de atributos (tamanho do vocabulário), $\mathcal{M}$ é a mediana do número de termos por documento e IQR é a amplitude interquartil (do inglês, *interquartile range*) do número de termos por documento.

Conforme esperado, pode ser observado na Tabela 7 que a regra de combinação que gerou maior número de atributos para todas as bases de dados foi a Regra 5, que combina os termos originais, os normalizados e os obtidos pelo processo de geração de conceitos. Por outro lado, a Regra 9, que aplica apenas normalização léxica, foi a que resultou no menor número de atributos.

Foram avaliados os desempenhos dos métodos de classificação *online* descritos no capítulo anterior: ALMA, B.NB, MDL, `MDLText`, M.NB, OGD, PA, Perceptron e ROMMA. Os métodos B.NB e M.NB foram implementados em `Python` usando a biblioteca `scikit-learn` (PEDREGOSA et al., 2011). Os métodos ALMA, OGD, PA, Perceptron e

Tabela 7 – Estatísticas das bases de dados de comentários do YouTube.

	Eminem			Katy Perry			LMFAO			Psy			Shakira		
	$\|d\|$	$\mathcal{M}$	IQR	$\|d\|$	$\mathcal{M}$	IQR	$\|d\|$	$\mathcal{M}$	IQR	$\|d\|$	$\mathcal{M}$	IQR	$\|d\|$	$\mathcal{M}$	IQR
Regra 1[†]	1601	7,00	13,50	1755	11,00	12,00	954	6,00	5,00	1429	9,00	9,00	1357	6,00	12,00
Regra 2	4953	37,00	72,00	4977	39,00	57,00	3174	33,50	25,00	4120	38,50	50,00	4211	23,00	54,00
Regra 3	1997	9,00	16,50	2129	13,00	14,00	1192	7,00	6,00	1734	12,00	13,00	1692	7,00	14,00
Regra 4	1773	9,00	16,00	1896	13,00	14,00	1068	6,00	6,00	1574	11,50	13,00	1501	7,00	12,00
Regra 5	5038	37,00	74,00	5035	40,00	57,00	3227	34,00	25,00	4195	41,00	51,00	4283	23,50	56,00
Regra 6	2155	10,00	21,00	2260	15,00	17,00	1298	7,00	7,00	1871	14,00	16,00	1823	8,00	16,00
Regra 7	4564	37,00	72,00	4631	38,00	55,00	2973	33,00	24,00	3834	38,00	49,00	3863	22,00	52,00
Regra 8	1461	8,00	15,00	1666	12,00	13,00	932	6,00	5,00	1354	10,50	11,00	1233	6,00	12,00
Regra 9	1424	8,00	14,00	1604	11,00	12,00	889	6,00	5,00	1307	10,00	11,00	1223	6,00	12,00
Regra 10	4721	37,00	73,00	4768	38,00	55,00	3061	33,00	25,00	3953	38,00	49,00	4025	23,00	53,00
Regra 11	1810	9,00	17,50	1974	14,00	16,00	1121	7,00	7,00	1610	13,00	14,00	1551	7,50	14,00

[†] base de dados original

ROMMA foram implementados em `MATLAB` usando biblioteca `LIBOL` (HOI; WANG; ZHAO, 2014). O MDL e o `MDLText` foram implementados em `C++`[4]. Todos os métodos foram testados com os parâmetros padrões definidos em suas respectivas bibliotecas.

Para avaliar os resultados, foi empregado o Coeficiente de Correlação de Matthews (MCC – *Matthews Correlation Coefficient*), tradicionalmente empregado em tarefas de classificação de *spam* como medida de qualidade de classificadores binários. Ele retorna valores reais entre -1 e $+1$, sendo que: $+1$ indica uma predição perfeita; 0, uma predição aleatória média; e -1, uma predição inversa (ALMEIDA; YAMAKAMI; ALMEIDA, 2011).

O MCC pode ser calculado da seguinte forma:

$$MCC = \frac{(v_p v_n) - (f_p f_n)}{\sqrt{(v_p + f_p)(v_p + f_n)(v_n + f_p)(v_n + f_n)}}, \tag{4.1}$$

onde v_p (verdadeiros positivos) é o número de comentários do YouTube corretamente classificados como *spam*, f_p (falsos positivos) é o número de comentários incorretamente classificados com *spam*, v_n (verdadeiros negativos) refere-se ao número de comentários corretamente classificados como *ham* e f_n (falsos negativos) refere-se ao número de comentários incorretamente classificados como *ham*.

Para simular um cenário real, foi considerado que um pequeno número de amostras é disponibilizado para treinar o método de classificação (20% de cada base de dados) e, em seguida, uma mensagem por vez é apresentada ao classificador. O classificador faz uma predição da nova amostra e recebe um *feedback*. Caso a predição esteja errada, o modelo de treinamento é atualizado com a classe correta da amostra. Esse processo é detalhado no Algoritmo 1.

[4] `MDLText`. Disponível em <https://github.com/renatoms88/MDLText>, acessado em 04/01/2017.

Algoritmo 1 *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 as demais mensagens em $\mathcal{D}$

  $df \leftarrow$ quantidade de mensagens em $\mathcal{D}_{treino}$ em que cada termo aparece
  $n_{treino} \leftarrow$ quantidade de mensagens em $\mathcal{D}_{treino}$
  $\mathcal{D}_{treino} \leftarrow$ converte as mensagens de $\mathcal{D}_{treino}$ de *TF* para *TF-IDF* usando df e n_{treino}

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

  $i \leftarrow 1$
  para cada mensagem d em $\mathcal{D}_{teste}$ **faça**
    $\hat{d} \leftarrow$ cópia de d antes de convertê-la para *TF-IDF*
    $d \leftarrow$ converte d de *TF* para *TF-IDF* usando df e n_{treino}

    $\hat{c}_i \leftarrow$ o classificador faz a predição da classe da mensagem
    $c_i \leftarrow$ classe verdadeira da mensagem
    $\ell \leftarrow$ o classificador calcula o erro de predição
    se $\ell > 0$ **então**
      $df \leftarrow$ atualiza df
      $n_{treino} \leftarrow n_{treino} + 1$
      $d \leftarrow$ converte $\hat{d}$ de *TF* para *TF-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

4.3 Resultados

A Tabela 8 apresenta os resultados obtidos por cada método de classificação aplicado sobre cada base de dados. Cada valor representa a média do MCC obtido em 10 rodadas dos experimentos executados de acordo com o Algoritmo 1, sendo que em cada rodada os comentários foram aleatoriamente ordenados. Os valores em negrito indicam a melhor regra de combinação para cada método. Os valores em negrito precedidos pelo símbolo "*" indicam o melhor resultado considerando todos os métodos e todas as regras de combinação.

Tabela 8 – Resultados obtidos por cada método de classificação aplicado sobre cada base de dados processada pelas regras de combinação.

	ALMA	B.NB	M.NB	MDL	MDLText	OGD	PA	Perceptron	ROMMA
					Eminem				
Regra 1†	0,761	0,813	0,785	0,782	0,806	**0,703**	**0,816**	0,748	0,759
Regra 2	0,724	0,756	0,755	0,744	0,753	0,647	0,764	0,684	0,730
Regra 3	0,745	0,808	0,786	0,777	**0,810**	0,681	0,808	0,752	0,761
Regra 4	**0,770**	0,799	0,786	0,792	0,808	0,680	0,813	**0,761**	0,763
Regra 5	0,723	0,754	0,755	0,758	0,768	0,656	0,755	0,705	0,725

Continua na próxima página

Tabela 8 – *Continuação*

	ALMA	B.NB	M.NB	MDL	MDLText	OGD	PA	Perceptron	ROMMA
Regra 6	0,752	0,804	**0,787**	0,788	**0,810**	0,682	0,803	0,744	**0,770**
Regra 7	0,694	0,769	0,746	0,741	0,763	0,666	0,757	0,686	0,706
Regra 8	0,754	0,794	0,777	**0,793**	0,775	0,626	0,780	0,712	0,740
Regra 9	0,750	***0,824**	0,774	0,777	0,800	0,626	0,795	0,729	0,753
Regra 10	0,700	0,763	0,728	0,755	0,756	0,627	0,743	0,673	0,717
Regra 11	0,753	0,812	0,778	0,788	0,798	0,652	0,794	0,718	0,754
					Katy Perry				
Regra 1†	0,764	0,818	0,780	0,748	0,795	0,766	0,801	0,766	0,764
Regra 2	0,724	0,721	0,744	0,700	0,761	0,697	0,769	0,715	0,692
Regra 3	0,743	0,793	0,765	0,733	0,819	0,742	0,778	0,767	0,745
Regra 4	**0,783**	**0,832**	0,790	0,778	***0,839**	**0,788**	**0,814**	0,773	**0,778**
Regra 5	0,734	0,746	0,735	0,722	0,778	0,704	0,756	0,713	0,721
Regra 6	0,781	0,800	0,779	0,767	0,828	0,774	0,790	0,773	0,765
Regra 7	0,720	0,709	0,735	0,723	0,752	0,689	0,750	0,726	0,702
Regra 8	0,756	0,793	0,787	**0,782**	0,816	0,786	0,802	**0,780**	0,764
Regra 9	0,764	0,810	**0,804**	0,765	0,809	0,745	0,800	0,760	0,763
Regra 10	0,712	0,732	0,737	0,716	0,761	0,692	0,745	0,737	0,716
Regra 11	0,757	0,826	0,785	0,778	0,813	0,752	0,802	0,777	0,768
					LMFAO				
Regra 1†	0,733	***0,849**	0,806	**0,815**	0,771	0,721	0,791	0,734	0,723
Regra 2	**0,754**	0,774	0,788	0,771	0,758	0,728	0,761	0,738	0,724
Regra 3	0,740	0,816	0,797	0,794	0,781	0,728	0,777	0,734	0,715
Regra 4	0,752	0,840	0,804	0,797	0,786	0,737	**0,803**	0,751	0,748
Regra 5	0,739	0,780	0,759	0,782	0,757	0,749	0,781	0,739	0,728
Regra 6	0,746	0,819	**0,808**	0,774	0,791	**0,752**	0,799	**0,756**	**0,758**
Regra 7	0,725	0,767	0,750	0,779	0,756	0,738	0,765	0,730	0,731
Regra 8	0,731	0,840	0,805	0,785	**0,807**	0,678	0,774	0,719	0,717
Regra 9	0,737	0,841	0,800	0,801	0,781	0,721	0,792	0,720	0,747
Regra 10	0,718	0,779	0,758	0,754	0,743	0,733	0,764	0,733	0,715
Regra 11	0,738	0,836	0,798	0,767	0,784	0,751	0,788	0,736	0,742
					Psy				
Regra 1†	0,823	0,848	***0,878**	0,814	0,859	0,847	**0,866**	0,835	0,826
Regra 2	0,761	0,745	0,784	0,744	0,795	0,775	0,794	0,771	0,755
Regra 3	**0,839**	0,852	0,859	**0,836**	0,873	**0,858**	0,864	0,843	**0,837**
Regra 4	0,837	0,844	0,853	0,793	***0,878**	0,839	0,863	**0,846**	0,823
Regra 5	0,765	0,753	0,813	0,774	0,825	0,769	0,791	0,790	0,772
Regra 6	0,831	0,841	0,854	0,823	0,853	0,847	**0,866**	0,843	0,823
Regra 7	0,738	0,726	0,795	0,742	0,802	0,770	0,782	0,752	0,740
Regra 8	0,825	0,859	0,852	0,803	0,843	0,804	0,852	0,825	0,826
Regra 9	0,812	**0,863**	0,848	0,797	0,848	0,802	0,862	0,837	0,836
Regra 10	0,747	0,744	0,777	0,740	0,792	0,776	0,787	0,753	0,735
Regra 11	0,820	0,840	0,825	0,818	0,851	0,836	0,860	0,835	0,797
					Shakira				
Regra 1†	0,803	0,764	0,815	**0,815**	***0,842**	0,823	0,839	0,827	0,816
Regra 2	0,755	0,661	0,724	0,774	0,778	0,756	0,788	0,757	0,756

Continua na próxima página

Tabela 8 – *Continuação*

	ALMA	B.NB	M.NB	MDL	MDLText	OGD	PA	Perceptron	ROMMA
Regra 3	**0,814**	**0,773**	**0,827**	0,811	0,837	**0,835**	***0,842**	**0,831**	**0,819**
Regra 4	0,805	0,764	0,799	0,800	0,824	0,811	0,827	0,803	0,806
Regra 5	0,748	0,657	0,723	0,751	0,790	0,774	0,789	0,767	0,770
Regra 6	0,813	0,747	0,821	0,802	0,827	0,813	0,834	0,819	0,813
Regra 7	0,751	0,662	0,738	0,750	0,775	0,761	0,784	0,745	0,745
Regra 8	0,805	0,740	0,803	0,786	0,812	0,771	0,828	0,791	0,797
Regra 9	0,799	0,749	0,797	0,772	0,822	0,800	0,838	0,815	0,797
Regra 10	0,746	0,658	0,727	0,755	0,788	0,771	0,777	0,749	0,741
Regra 11	0,799	0,727	0,808	0,793	0,812	0,791	0,826	0,816	0,789

† base de dados original

Individualmente, o `MDLText` obteve os melhores resultados nas bases de dados *Katy Perry*, *Psy* (empatado com o M.NB) e *Shakira* (empatado com o PA), enquanto que o B.NB obteve os melhores resultados nas bases *Eminem* e *LMFAO*. O melhor resultado geral foi obtido pelos métodos M.NB e `MDLText` na base *Psy*, com MCC de 0,878. Já o pior resultado geral foi obtido pelo método OGD na base *Eminem*, com MCC de 0,626.

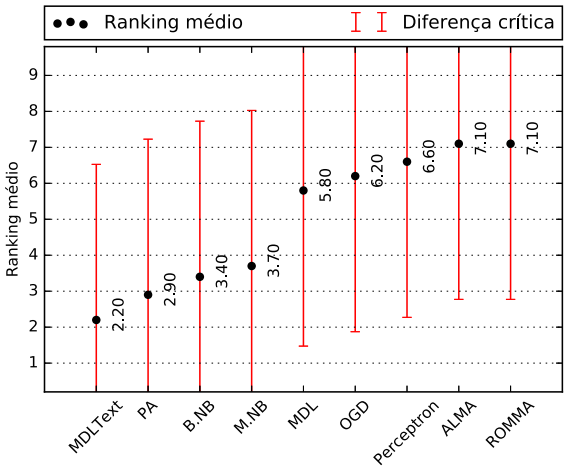
Para certificar-se que os resultados não foram obtidos por acaso, foi realizada uma análise estatística usando o método de Friedman (DEMŠAR, 2006), que usa as informações sobre o *ranking* médio dos métodos em cada base de dados para checar se a hipótese nula, que afirma que todos os métodos são equivalentes, pode ser descartada.

O *ranking* médio de cada método, considerando os melhores resultados para cada base de dados, é apresentado na Figura 4(a). Conforme pode ser observado, o melhor *ranking* médio foi o obtido pelo `MDLText`, enquanto o *ranking* dos métodos ALMA e ROMMA foram os piores. Com um intervalo de confiança $\alpha = 0{,}05$, a hipótese nula pode ser seguramente descartada. Assim, é possível afirmar que existe uma diferença estatística entre os resultados apresentados. Em seguida, foi feita a comparação par-a-par entre os resultados dos métodos usando o teste *post-hoc* de Bonferroni–Dunn (DEMŠAR, 2006). Este teste afirma que dois modelos ou grupos são estatisticamente diferentes se a diferença entre seus *rankings* for superior a uma diferença crítica. Para um intervalo de confiança $\alpha = 0{,}05$, $k = 9$ (número de modelos ou grupos) e $n = 5$ (número de observações), a diferença crítica é 4,327. Portanto, apesar do `MDLText` ter obtido o melhor *ranking* médio, o teste *post-hoc* indica que o resultado obtido por ele foi equivalente aos métodos B.NB, MDL, M.NB, OGD e PA e estatisticamente superior aos métodos ALMA, Perceptron e ROMMA.

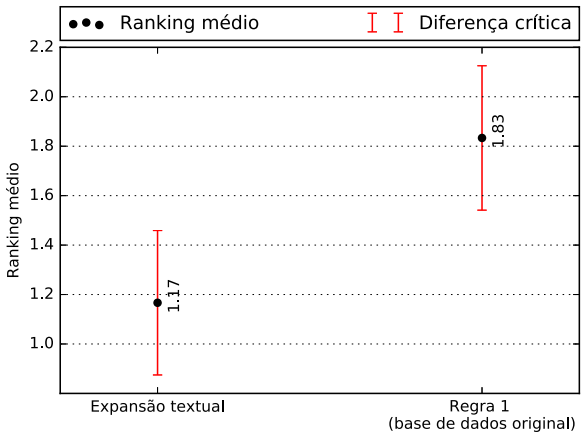
Avaliando o impacto da expansão textual sobre os resultados, pode-se notar que conforme apresentado na Tabela 8, na maioria dos experimentos, os resultados obtidos foram superiores quando pelo menos uma das regras de expansão textual (normalização léxica, geração de conceitos ou desambiguação) foi utilizada.

Figura 4 – *Rankings* médios usados nas análises estatísticas e diferenças críticas calculadas usando o teste de Bonferroni–Dunn. A Figura 4(a) apresenta o *ranking* médio obtido por cada método de classificação avaliado. A Figura 4(b) apresenta o *ranking* médio obtido com a base de dados original e usando expansão textual. A Figura 4(c) apresenta o *ranking* médio obtido pelos métodos de classificação com cada regra de combinação.

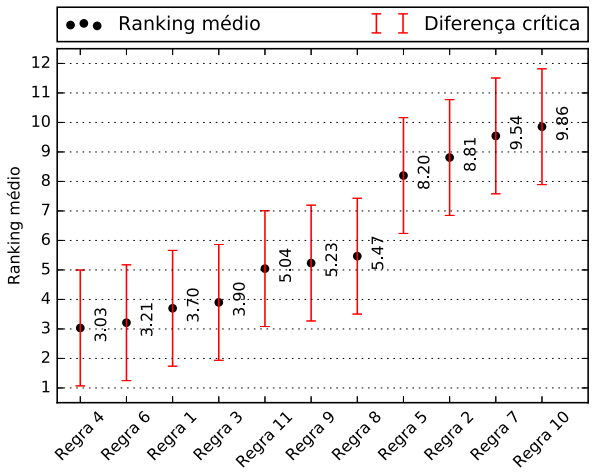
(a)



(b)



(c)



Para verificar se os resultados obtidos nos experimentos com expansão textual foram estatisticamente superiores aos resultados adquiridos nos experimentos com as bases de dados originais, também foi utilizado o teste *post-hoc* de Bonferroni–Dunn. Para um intervalo de confiança $\alpha = 0{,}05$, $k = 2$ e $n = 45$, a diferença crítica é 0,292. Dados os *rankings* médios dos experimentos com as bases originais e com expansão textual, ilustrados na Figura 4(b), pode-se afirmar que existe evidência estatística que confirma que os métodos de classificação obtiveram melhores resultados quando foi aplicada expansão textual.

Também, foi avaliado se alguma das regras de combinação ofereceu resultados estatisticamente superiores às demais. O teste de Friedman descartou a hipótese nula de igualdade entre elas. Conforme apresentado na Figura 4(c), a Regra 4 obteve o melhor *ranking* médio e as Regras 4 e 6 obtiveram *ranking* médio superior ao obtido pelos experimentos com as bases de dados originais (Regra 1). Porém, segundo o teste de Bonferroni–Dunn, com um intervalo de confiança $\alpha = 0{,}05$, $k = 11$ e $n = 45$, para que tais regras fossem, individualmente, estatisticamente superiores à Regra 1, a diferença crítica entre elas deveria ser superior à 1,963, o que não ocorreu. Sendo assim, não é possível afirmar que alguma das regras de expansão, individualmente, obteve resultados estatisticamente superiores à base de dados original.

O teste estatístico mostrou também que as Regras 2, 5, 7 e 10 foram significativamente inferiores às demais, o que confirma a hipótese de que, no contexto dos comentários do YouTube, a técnica de geração de conceitos, quando não sucedida pela técnica de desambiguação, agregou muito ruído às amostras originais. A Regra 4, que obteve o melhor *ranking*, foi considerada estatisticamente equivalente às Regras 1, 3 e 6; e superior às demais. Já, a Regra 6 foi estatisticamente equivalente às Regras 1, 3, 4 e 11; e superior às demais

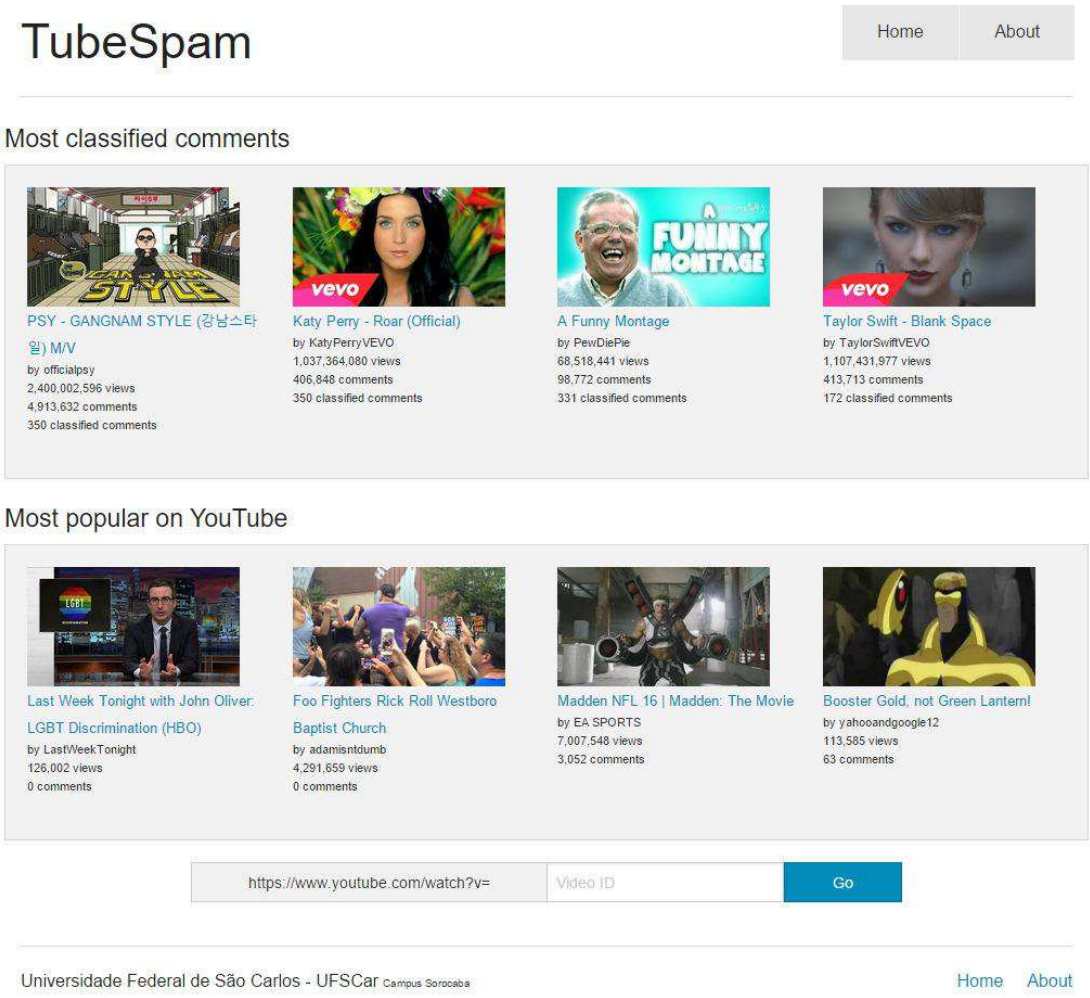
De maneira geral, após relacionar todas as informações dos experimentos e das análises estatísticas, é possível concluir que a melhor configuração para realizar a filtragem automática de comentários indesejados postados no YouTube foi obtida quando foram empregados os métodos `MDLText`, PA e *Naïve* Bayes, combinados com a expansão das amostras de texto pelas Regras 4, 6, 1 ou 3.

5 TubeSpam

Com base nos resultados obtidos nos experimentos, foi desenvolvida uma ferramenta *online* chamada `TubeSpam`[1], que filtra automaticamente comentários indesejados postados no YouTube.

A Figura 5 ilustra a página inicial da aplicação, onde é possível selecionar um vídeo dentre duas listas: os vídeos com comentários já classificados no `TubeSpam` e os vídeos mais populares no YouTube. Também é possível escolher um vídeo arbitrário usando o seu *ID* no YouTube.

Figura 5 – Página inicial do `TubeSpam`. O usuário pode escolher um vídeo pré-selecionado ou informar um *ID* de um vídeo específico na parte inferior da página.



Fonte: `TubeSpam`.

[1] `TubeSpam`. Disponível em <http://lasid.sor.ufscar.br/tubespam/>. Acessado em 30/12/2016.

Dentre as funcionalidades que a ferramenta oferece, destacam-se:

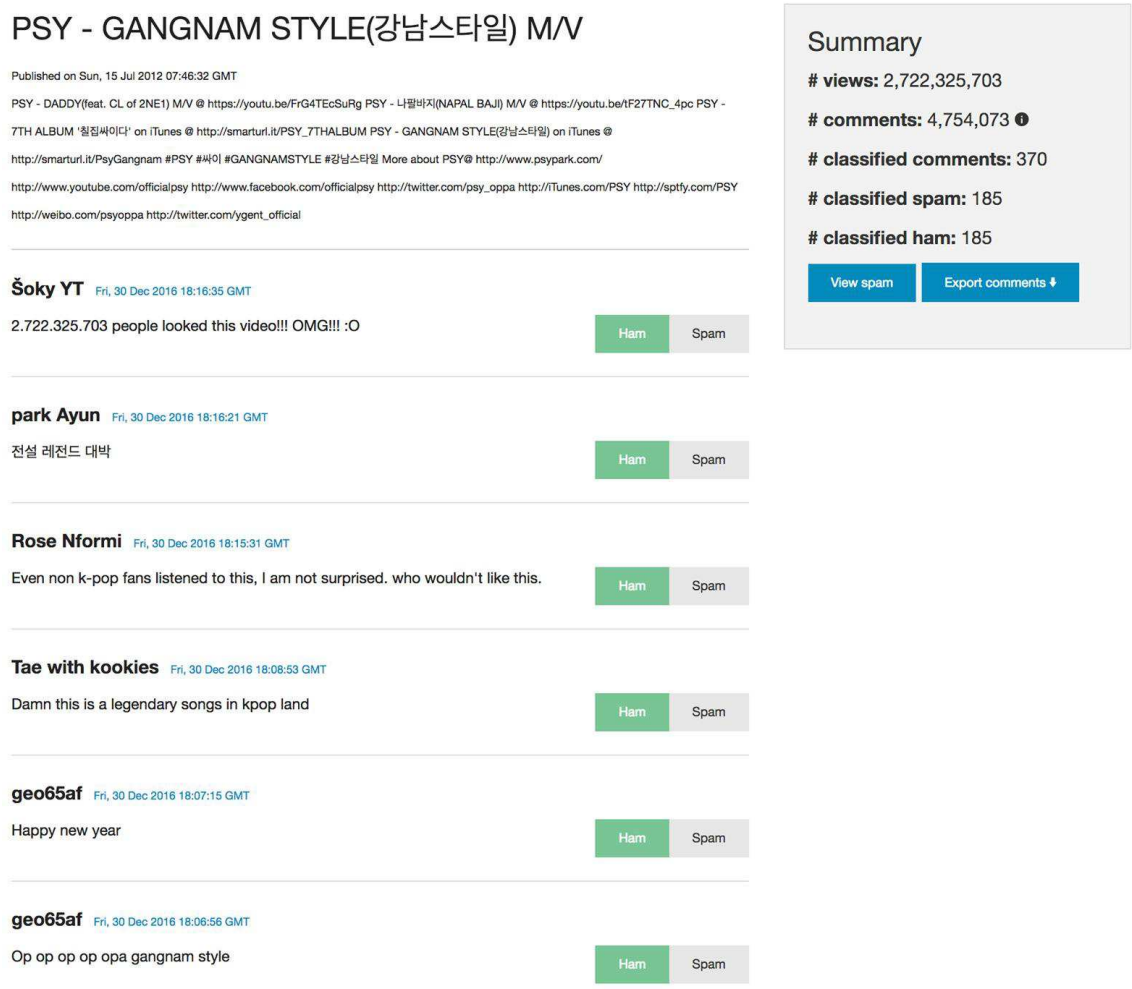
- filtragem automática de comentários indesejados publicados em vídeos do YouTube;
- classificador genérico pronto para ser usado em qualquer vídeo do YouTube;
- classificador específico para vídeos de um mesmo canal, usado em casos quando o classificador genérico já não apresenta bons resultados;
- correção pelo usuário dos rótulos de mensagens classificadas incorretamente;
- modo de visualização em que apenas os comentários legítimos são mostrados, permitindo ao usuário visualizar apenas o conteúdo que realmente importa;
- modo de visualização em que apenas os comentários indesejados são mostrados, permitindo ao usuário revisar quais mensagens foram bloqueadas pela ferramenta; e
- possibilidade de exportar as mensagens em formato CSV.

Inicialmente, quando um vídeo é carregado no `TubeSpam` pela primeira vez, é empregado um classificador genérico, que foi treinado com as mensagens das cinco bases de dados utilizadas pelos experimentos. Quando um usuário informar comentários que foram classificados erroneamente, essas correções são usadas para gerar um novo classificador específico para aquele vídeo e todos os vídeos daquele canal, que passa a ser utilizado, no lugar do classificador genérico. Este classificador específico também aprende incrementalmente por correções feitas em outros vídeos daquele mesmo canal.

Todos os classificadores são gerados utilizando o método `MDLText`. A escolha deste algoritmo reside no fato de que, nos experimentos realizados, o `MDLText` apresentou o melhor *ranking* médio, além de oferecer bom balanço entre robustez e esforço computacional. Antes de serem submetidos ao classificador, os comentários são expandidos com a ferramenta `TextExpansion`, utilizando a regra de combinação 4, que mantém os termos originais e aplica normalização léxica. A escolha dessa regra de combinação também é baseada nos resultados obtidos pelos experimentos, no qual a Regra 4 obteve o melhor *ranking* médio. Além disso, a normalização léxica é uma técnica bastante eficiente e rápida, sendo adequada para uma ferramenta *online* como o `TubeSpam`.

A Figura 6 apresenta um exemplo do `TubeSpam` em ação, com comentários postados no vídeo *Psy – Gangnam Style* sendo filtrados automaticamente. Nesta página, todas as mensagens rotuladas como *spam* são escondidas, permitindo ao usuário visualizar apenas os comentários legítimos. A qualquer momento, o usuário pode informar um comentário que foi classificado incorretamente e corrigir o seu rótulo. Essas correções são usadas no treinamento incremental do classificador.

Figura 6 – `TubeSpam` filtrando automaticamente comentários *spam* postados no vídeo *Psy – Gangnam Style*.

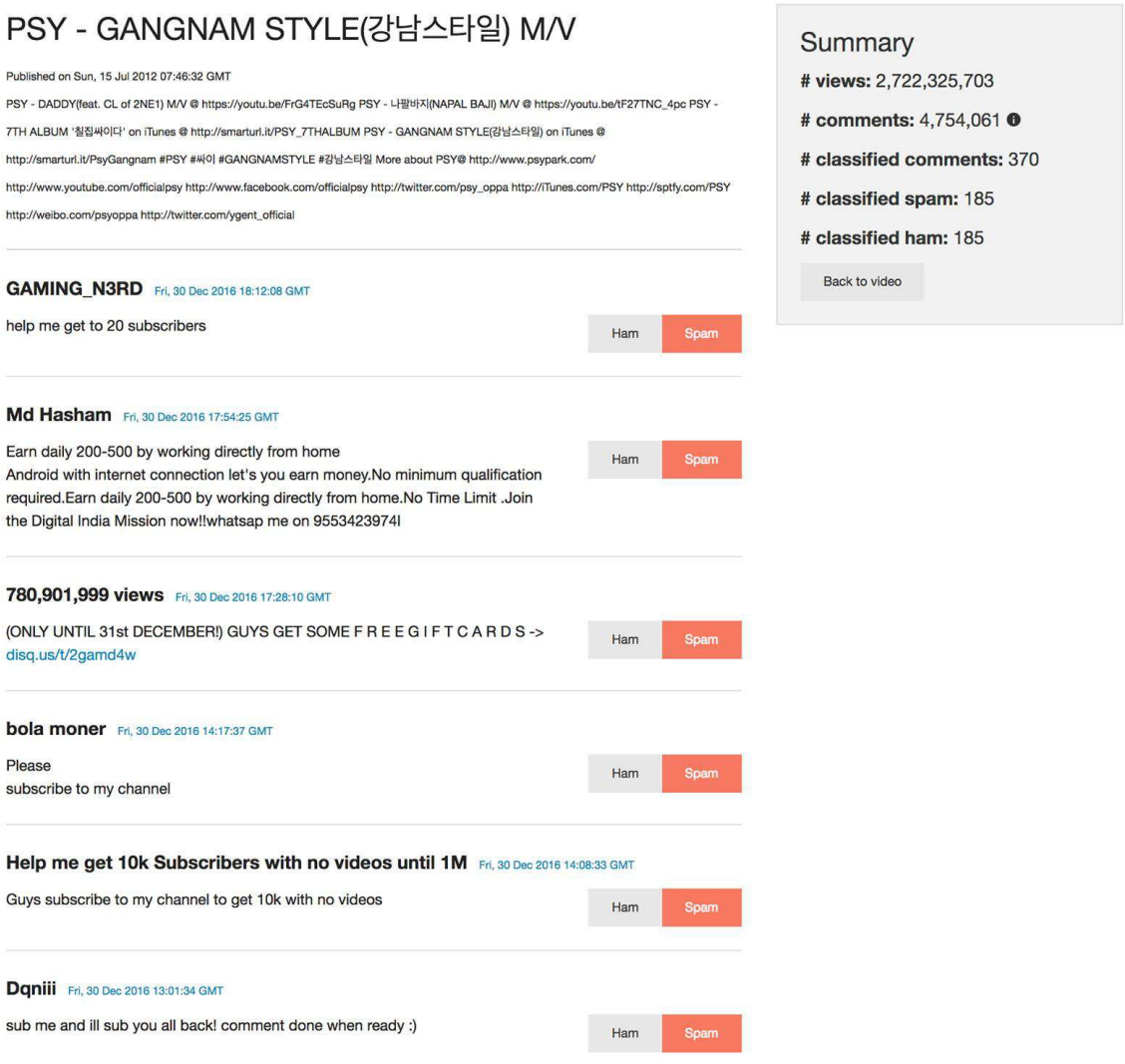


Fonte: `TubeSpam`.

Também é possível revisar quais comentários foram bloqueados pelo `TubeSpam`. A Figura 7 mostra as mensagens rotuladas como *spam* para o mesmo vídeo. Nesta página, o usuário também pode informar comentários classificados erroneamente, que são então inseridos ao treinamento incremental.

Dentre possíveis extensões que a ferramenta oferece estão a adaptação dos classificadores para filtrar mensagens consideradas ofensivas e a utilização dos comentários e seus rótulos em outras plataformas, como *plugins* em navegadores e dispositivos móveis.

Figura 7 – Comentários postados no vídeo *Psy – Gangnam Style* identificados como *spam* pelo `TubeSpam`.



Fonte: `TubeSpam`.

Conclusão

O YouTube é uma plataforma de distribuição e compartilhamento de vídeos, que também possui recursos de redes sociais. Recentemente, devido à alta popularidade e ao novo sistema de monetização nos vídeos, a plataforma foi inundada com conteúdo indesejado, publicado por usuários mal-intencionados que tentam se promover ou disseminar *links* maliciosos que podem infectar os computadores dos outros usuários com vírus e *malwares*.

A filtragem automática de *spam* nos comentários do YouTube ainda é um problema pouco explorado, evidenciado pela escassez de ferramentas disponíveis para auxiliar na moderação das postagens. Desta forma, proprietários de canais famosos têm desabilitado a seção de comentários em seus vídeos, passando a utilizar outros meios para interagir com os espectadores.

O problema de detecção de *spam* é inerentemente *online*, visto que o padrão das mensagens de *spam* muda com o tempo, exigindo que os métodos empregados possuam treinamento *online* e incremental. Outro agravante, é que as mensagens postadas no YouTube costumam ser curtas e repletas de erros de digitação, gírias, símbolos e abreviações, fazendo com que métodos consolidados de filtragem de *spam* em emails possam apresentar desempenho insatisfatório.

Nesse cenário, o principal objetivo desse trabalho foi encontrar métodos e configurações promissores que podem ser efetivamente empregados para minimizar os efeitos deste problema, além de oferecer uma ferramenta *online* para detectar e filtrar automaticamente comentários *spam* publicados em vídeos do YouTube, o `TubeSpam`.

Inicialmente, foram coletadas e criadas cinco bases de dados reais, públicas e não-codificadas extraídas diretamente do YouTube, que foram rotuladas e utilizadas para avaliar o desempenho de diversos métodos tradicionais de classificação *online*. Além destes, foi avaliado o desempenho de um método promissor de classificação de texto baseado no princípio da descrição mais simples, o `MDLText`.

Os resultados obtidos evidenciaram o bom desempenho da maioria dos métodos de classificação avaliados na filtragem automática de *spam* em comentários do YouTube. Individualmente, dentre as cinco bases de dados avaliadas, o `MDLText` obteve o melhor resultado em três e o B.NB obteve o melhor resultado na outras duas. O melhor desempenho, na média, foi obtido pelo método `MDLText` e os piores desempenhos foram obtidos pelos métodos ALMA e ROMMA. A análise estatística realizada com o teste de Friedman garantiu que os resultados não foram obtidos por acaso e, posteriormente, através do teste de Bonferroni–Dunn, foi possível concluir que há similaridade estatística entre os métodos

`MDLText`, PA, B.NB, M.NB, MDL e OGD.

Também, foi avaliado se a aplicação de técnicas de normalização léxica e indexação semântica podem trazer benefícios à classificação, uma vez que as mensagens originais normalmente são curtas e repletas de gírias, símbolos e abreviações. Neste caso, a análise estatística demonstrou que, de maneira geral, o desempenho dos métodos de classificação *online* foram superiores quando alguma técnica de expansão textual foi utilizada. O melhor resultado, na média, foi obtido quando os termos originais foram mantidos e normalização léxica foi aplicada.

Em virtude da complexidade dos experimentos ao lidar com 11 variações para cada base de dados, não foi realizada *grid-search* ou qualquer ajuste de parâmetros para os métodos avaliados. Além disso, foram avaliados os métodos em suas versões mais estáveis e utilizadas na literatura, desconsiderando variações como PA-I, PA-II e *aggressive*-ROMMA, ou o OGD com diferentes funções de erro.

Dentre as limitações da ferramenta desenvolvida, o `TubeSpam`, está a dependência da API do YouTube, usada para acessar as informações e comentários dos vídeos, e a dependência do `TextExpansion` e suas particularidades, como a restrição à língua inglesa e dependência dos dicionários utilizados.

Trabalhos futuros envolvem o desenvolvimento de *plugins* para os navegadores mais utilizados, como *Chrome* e *Firefox*, para que o usuário possa visualizar os comentários filtrados diretamente no YouTube, sem precisar acessar a página do `TubeSpam`.

Publicações

Durante o período do mestrado, os seguintes trabalhos foram produzidos em colaboração com outros pesquisadores ou de maneira independente:

Periódicos

1. R.M. SILVA; T.C. ALBERTO; T.A. ALMEIDA; A. YAMAKAMI. Towards filtering undesired short text messages using an online learning approach with semantic indexing. *Expert Systems with Applications*, Elsevier, 2016. (Em avaliação)

Congressos

1. T.C. ALBERTO; J. VON LOCHTER; T.A. ALMEIDA. Filtragem Automática de Spam nos Comentários do YouTube. In: *Anais do XII Encontro Nacional de Inteligência Artificial e Computacional (ENIAC'15)*, Natal, Brasil, Novembro, 2015.

2. T.C. ALBERTO; J. VON LOCHTER; T.A. ALMEIDA. TubeSpam: Comment Spam Filtering on YouTube. In: *Proceedings of the 14th IEEE International Conference on Machine Learning and Applications (ICMLA'15)*, 138–143, Miami, FL, USA, Dezembro, 2015.

3. R.M. SILVA; T.C. ALBERTO; T.A. ALMEIDA; A. YAMAKAMI. Filtrando Comentários do YouTube através de Classificação Online baseada no Princípio MDL e Indexação Semântica. In: *Anais do XIII Encontro Nacional de Inteligência Artificial e Computacional (ENIAC'16)*, Recife, Brasil, Outubro, 2016.

Referências

ALBERTO, T. C.; LOCHTER, J. V.; ALMEIDA, T. A. TubeSpam: Comment spam filtering on YouTube. In: *Proceedings of the 14th IEEE International Conference on Machine Learning and Applications (ICMLA'15)*. Miami, FL, USA: [s.n.], 2015. p. 138–143. Citado na página 29.

ALMEIDA, T. A. et al. Text normalization and semantic indexing to enhance Instant Messaging and SMS spam filtering. *Knowledge-Based Systems*, Elsevier, v. 108, p. 25–32, 2016. Citado 4 vezes nas páginas 35, 36, 37 e 38.

ALMEIDA, T. A.; YAMAKAMI, A. Occam's razor-based spam filter. *Journal of Internet Services and Applications*, v. 3, n. 3, p. 245–253, 2012. Citado 3 vezes nas páginas 48, 49 e 50.

ALMEIDA, T. A.; YAMAKAMI, A.; ALMEIDA, J. Spam filtering: how the dimensionality reduction affects the accuracy of naive Bayes classifiers. *Journal of Internet Services and Applications, JISA'11*, Springer-Verlag, v. 1, n. 3, p. 183–200, 2011. Citado 2 vezes nas páginas 47 e 53.

ALSALEH, M. et al. Combating comment spam with machine learning approaches. In: *Proceedings of the 14th IEEE International Conference on Machine Learning and Applications (ICMLA'15)*. Miami, FL, USA: [s.n.], 2015. p. 295–300. Citado na página 30.

AMMARI, A.; DIMITROVA, V.; DESPOTAKIS, D. Identifying relevant youtube comments to derive socially augmented user models: A semantically enriched machine learning approach. In: ARDISSONO, L.; KUFLIK, T. (Ed.). *Advances in User Modeling: UMAP 2011 Workshops, Girona, Spain, July 11-15, 2011, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 71–85. ISBN 978-3-642-28509-7. Disponível em: <http://dx.doi.org/10.1007/978-3-642-28509-7_8>. Citado na página 29.

BALDWIN, T. et al. How noisy social media text, how diffrnt social media sources? In: *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP'13)*. Nagoya, Japan: [s.n.], 2013. p. 356–364. Disponível em: <http://aclweb.org/anthology/I/I13/I13-1041.pdf>. Citado na página 29.

BENEVENUTO, F. et al. Detecting spammers and content promoters in online video social networks. In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2009. (SIGIR '09), p. 620–627. ISBN 978-1-60558-483-6. Citado na página 30.

BRATKO, A. et al. Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, JMLR.org, v. 7, p. 2673–2698, 2006. Citado 3 vezes nas páginas 27, 28 e 41.

CHAKRABORTY, M. et al. Recent developments in social spam detection and combating techniques: A survey. *Information Processing & Management*, v. 52, n. 6, p. 1053–1073, 2016. ISSN 0306-4573. Citado na página 28.

CHAUDHARY, V.; SUREKA, A. Contextual feature based one-class classifier approach for detecting video response spam on youtube. In: *Proceedings of the 11th Annual International Conference on Privacy, Security and Trust (PST'13)*. Tarragona, Spain: IEEE, 2013. p. 195–204. Citado na página 28.

CHOWDURY, R. et al. A data mining based spam detection system for youtube. In: *Proceedings of the 8th International Conference on Digital Information Management (ICDIM'13)*. Islamabad, Pakistan: IEEE, 2013. p. 373–378. Citado na página 28.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297, 1995. ISSN 1573-0565. Disponível em: <http://dx.doi.org/10.1007/BF00994018>. Citado na página 43.

CRAMMER, K. et al. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, JMLR.org, v. 7, p. 551–585, dez. 2006. Citado 4 vezes nas páginas 41, 42, 44 e 45.

DALVI, N. et al. Adversarial classification. In: *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. Seattle, WA, USA: ACM, 2004. p. 99–108. ISBN 1-58113-888-1. Citado 2 vezes nas páginas 27 e 28.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, JMLR.org, v. 7, p. 1–30, 2006. ISSN 1532-4435. Citado na página 56.

EISENSTEIN, J. What to do about bad language on the internet. In: *Proceedings of NAACL-HLT 2013*. Atlanta, GA, USA: Association for Computational Linguistics, 2013. p. 359–369. Citado na página 29.

FAWCETT, T. "In vivo" spam filtering: A challenge problem for KDD. *ACM SIGKDD Explorations Newsletter*, ACM, New York, NY, USA, v. 5, n. 2, p. 140–148, 2003. ISSN 1931-0145. Citado na página 27.

GABRILOVICH, E.; MARKOVITCH, S. Feature generation for text categorization using world knowledge. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. (IJCAI'05), p. 1048–1053. Citado na página 34.

GENTILE, C. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, JMLR.org, v. 2, p. 213–242, mar. 2001. Citado 3 vezes nas páginas 41, 43 e 44.

GOODMAN, J.; HECKERMAN, D.; ROUNTHWAITE, R. Stopping spam. *Scientific American*, Nature Publishing Group, v. 292, p. 42–49, 2005. Citado na página 27.

GYÖNGYI, Z.; GARCIA-MOLINA, H. Web spam taxonomy. In: *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web*. Chiba, Japan: [s.n.], 2005. p. 39–47. Citado na página 27.

HOI, S. C. H.; WANG, J.; ZHAO, P. Libol: A library for online learning algorithms. *Journal of Machine Learning Research*, JMLR.org, v. 15, n. 1, p. 495–499, jan. 2014. Citado 5 vezes nas páginas 41, 42, 45, 46 e 53.

LANGLEY, P.; IBA, W.; THOMPSON, K. An analysis of bayesian classifiers. In: *Proceedings of the 10th National Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 1992. (AAAI'92), p. 223–228. Citado 3 vezes nas páginas 41, 46 e 47.

LI, Y.; LONG, P. M. The relaxed online maximum margin algorithm. *Machine Learning*, v. 46, n. 1-3, p. 361–387, jan. 2002. Citado 3 vezes nas páginas 41, 42 e 43.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. Citado 3 vezes nas páginas 41, 46 e 47.

MISHNE, G.; CARMEL, D.; LEMPEL, R. Blocking blog spam with language model disagreement. In: *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb'05)*. Chiba, Japan: [s.n.], 2005. p. 1–6. Citado na página 29.

NAVIGLI, R.; PONZETTO, S. P. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, v. 193, p. 217–250, 2012. Citado na página 36.

NAVIGLI, R.; PONZETTO, S. P. Multilingual WSD with just a few lines of code: the BabelNet API. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*. Jeju, Korea: [s.n.], 2012. Citado na página 36.

O'CALLAGHAN, D. et al. Network analysis of recurring youtube spam campaigns. *CoRR*, abs/1201.3783, 2012. Disponível em: <http://arxiv.org/abs/1201.3783>. Citado 2 vezes nas páginas 29 e 30.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 52.

RĂDULESCU, C.; DINSOREANU, M.; POTOLEA, R. 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)*. Cluj-Napoca, Romania: IEEE, 2014. p. 29–35. Citado 2 vezes nas páginas 29 e 30.

RISSANEN, J. Modeling by shortest data description. *Automatica*, v. 14, p. 465–471, 1978. Citado 3 vezes nas páginas 41, 47 e 49.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958. Citado 3 vezes nas páginas 41, 42 e 43.

SAHAMI, M. et al. A bayesian approach to filtering junk e-mail. In: *Proceedings of the 15th National Conference on Artificial Intelligence*. Madison, WI, USA: [s.n.], 1998. p. 55–62. Citado na página 27.

SEBASTIANI, F. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, ACM, New York, NY, USA, v. 34, n. 1, p. 1–47, 2002. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/505282.505283>. Citado 2 vezes nas páginas 27 e 33.

SEVERYN, A. et al. Opinion mining on youtube. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, MD, USA: Association for Computational Linguistics, 2014. p. 1252–1261. Citado na página 30.

SILVA, R. M.; ALMEIDA, T. A.; YAMAKAMI, A. MDLText: An efficient and lightweight text classifier. *Knowledge-Based Systems*, Elsevier, v. 118, p. 152–164, 2017. ISSN 0950-7051. Citado 4 vezes nas páginas 41, 48, 49 e 50.

SILVA, R. M.; YAMAKAMI, A.; ALMEIDA, T. A. Quanto mais simples, melhor! categorização de textos baseada na navalha de Occam. In: *Anais do XII Encontro Nacional de Inteligência Artificial e Computacional (ENIAC'15)*. Natal, Brasil: [s.n.], 2015. p. 1–7. Citado 2 vezes nas páginas 49 e 50.

SILVA, T. P. da. *Normalização Textual e Indexação Semântica Aplicadas na Filtragem de SMS Spam*. Dissertação (Mestrado) — Universidade Federal de São Carlos, UFSCar Sorocaba, 2016. Citado na página 37.

SUREKA, A. Mining user comment activity for detecting forum spammers in youtube. *CoRR*, abs/1103.5044, 2011. Disponível em: <http://arxiv.org/abs/1103.5044>. Citado na página 30.

WILBUR, W. J.; KIM, W. The ineffectiveness of within-document term frequency in text classification. *Information Retrieval*, Kluwer Academic Publishers, Hingham, MA, USA, v. 12, n. 5, p. 509–525, 2009. Citado na página 34.

ZINKEVICH, M. Online convex programming and generalized infinitesimal gradient ascent. In: *Proceedings of the 20th International Conference on Machine Learning*. Washington, DC, USA: [s.n.], 2003. p. 928–936. Citado 3 vezes nas páginas 41, 45 e 46.