**DHENY FERNANDES**

# Classificação de Conteúdo Malicioso Baseado em Floresta de Caminhos Ótimos

**Universidade Estadual Paulista "Júlio de Mesquita Filho"**
**2016**

**DHENY FERNANDES**

# Classificação de Conteúdo Malicioso Baseado em Floresta de Caminhos Ótimos

**Dissertação** apresentada ao Programa de Pós-Graduação em Ciências da Computação como parte dos requisitos para obtenção do título de **MESTRE EM CIÊNCIA DA COMPUTAÇÃO.**

**Área de Concentração:** Computação Aplicada.

**Linha de Pesquisa:** Processamento de Imagens e Visão Computacional.

**Orientador:** Prof. Dr. João Paulo Papa.

**Coorientador:** Prof. Dr. Kelton Augusto Pontara da Costa

**Bauru - SP**
**2016**

Fernandes, Dheny.
Classificação de conteúdo malicioso baseado em floresta de caminhos ótimos / Dheny Fernandes. -- São José do Rio Preto, 2016
56 f. : il., tabs.

Orientador: João Paulo Papa
Coorientador: Kelton Augusto Pontara da Costa
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação - Matemática. 2. Redes de computadores - Medidas de segurança. 3. Sistemas de detecção de intrusão (Medidas de segurança) 4. Floresta de caminhos ótimos. 5. Aprendizado do computador. 6. Spam (Mensagens eletrônicas) I. Papa, João Paulo. II. Costa, Kelton Augusto Pontara da. III. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. IV. Título.
CDU – 681.3.025

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

**Universidade Estadual Paulista "Júlio de Mesquita Filho"**
**Instituto de Biociências, Letras e Ciências exatas**
**Programa de Pós-graduação em Ciência da Computação**

**ALUNO:** Dheny Fernandes .

**NÚMERO DE MATRÍCULA:** 135599-1.

**ÁREA DE CONCENTRAÇÃO:** Computação Aplicada.

**LINHA DE PESQUISA:** Processamento de Imagens e Visão Computacional.

**PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO:** Nível Mestrado.

**TÍTULO DA DISSERTAÇÃO:** Classificação de Conteúdo Malicioso Baseado em Floresta de Caminhos Ótimos.

**ORIENTADOR:** Prof. Dr. João Paulo Papa.

**COORIENTADOR:** Prof. Dr. Kelton Augusto Pontara da Costa
Esta dissertação foi Aprovada em reunião pública realizada na Sala de videoconferência, Campus de Bauru, em 19 de Maior de 2016, às 10 horas, pela seguinte Banca Examinadora:

| **NOME** | **ASSINATURA** |
|---|---|
| Prof. Dr. João Paulo Papa<br>UNESP - Universidade Estadual Paulista "Júlio de Mesquita Filho" | _____ |
| Prof(a). Dr(a). Aparecido Nilceu Marana<br>UNESP - Universidade Estadual Paulista "Júlio de Mesquita Filho" | _____ |
| Prof(a). Dr(a). Jurandy Gomes Almeida Jr.<br>UNIFESP - Universidade Federal de São Paulo | _____ |

Bauru-SP, 19 de Maio de 2016.

# Dedicatória

Dedico este trabalho aos meus pais, Jerônimo e Clizeide, pelo apoio incondicional dado a mim em todo este período de estudo. Dedico-o, também, à minha noiva, Isabelle, pela compreensão, ajuda e companheirismo. Amo vocês!

# Agradecimentos

Agradeço a Deus, porque Dele, por Ele e para Ele são todas as coisas. Agradeço a meus pais, que se dedicam mais à mim do que à eles próprios. Agradeço à minha noiva, que foi um ombro amigo em todos os momentos. Agradeço aos meus orientadores que me ensinaram, me corrigiram e me fizeram crescer. Agradeço à CAPES pelo apoio financeiro e à UNESP pela infraestrutura e apoio. Agradeço à todos meus amigos e companheiros que tornaram possível a realização deste projeto.

## Resumo

O advento da Internet trouxe amplos benefícios nas áreas de comunicação, entretenimento, compras, relações sociais, entre outras. Entretanto, várias ameaças começaram a surgir nesse cenário, levando pesquisadores a criar ferramentas para lidar com elas. Spam, *malwares*, conteúdos maliciosos, *pishing*, fraudes e falsas URLs são exemplos de ameaças. Em contrapartida, sistemas antivírus, *firewalls* e sistemas de detecção e prevenção de intrusão são exemplos de ferramentas de combate às tais ameaças. Principalmente a partir de 2010, encabeçado pelo *malware Stuxnet*, as ameaças tornaram-se muito mais complexas e persistentes, fazendo com que as ferramentas até então utilizadas se tornassem obsoletas. O motivo é que tais ferramentas, baseadas em assinaturas e anomalias, não conseguem acompanhar tanto a velocidade de desenvolvimento das ameaças quanto sua complexidade. Desde então, pesquisadores têm voltado suas atenções a métodos mais eficazes para se combater ciberameaças. Nesse contexto, algoritmos de aprendizagem de máquina estão sendo explorados na busca por soluções que analisem em tempo real ameaças provenientes da internet. Assim sendo, este trabalho tem como objetivo analisar o desempenho dos classificadores baseados em Floresta de Caminhos Ótimos, do inglês *Optimum-path Forest* (OPF), comparando-os com os demais classificadores do estado-da-arte. Para tanto, serão analisados dois métodos de extração de características: um baseado em tokens e o outro baseado em Ngrams, sendo N igual a 3. De maneira geral, o OPF mais se destacou no não bloqueio de mensagens legítimas e no tempo de treinamento. Em algumas bases a quantidade de spam corretamente classificada também foi alta. A versão do OPF que utiliza grafo completo foi melhor, apesar de que em alguns casos a versão com grafo *knn* se sobressaiu. Devido às exigências atuais em questões de segurança, o OPF, pelo seu rápido tempo de treinamento, pode ser melhorado em sua eficácia visando uma aplicação real. Em relação aos métodos de extração de características, 3gram foi superior, melhorando os resultados obtidos pelo OPF.

*Palavras-chave*: Segurança em Redes de Computadores, Floresta de Caminhos Ótimos, Spam, Classificação, Aprendizado de Máquina.

## Abstract

The advent of Internet has brought widespread benefits in the areas of communication, entertainment, shopping, social relations, among others. However, several threats began to emerge in this scenario, leading researchers to create tools to deal with them. Spam, malware, malicious content, phishing, fraud and false URLs are some examples of these threats. In contrast, anti-virus systems, firewalls and intrusion detection and prevention systems are examples of tools to combat such threats. Especially since 2010, headed by the Stuxnet malware, threats have become more complex and persistent, making the tools previously used became obsolete. The reason is that such tools based on signatures and anomalies can not follow both the speed of development of the threats and their complexity. Since then, researchers have turned their attention to more effective methods to combat cyber threats. In this context, machine learning algorithms are being exploited in the search for solutions to analyze real-time threats from the internet. Therefore, this study aims to analyze the performance of classifiers based on Optimum-path Forest, OPF, comparing them with the other state-of-the-art classifiers. To do so, two features extraction methods will be analyzed: one based on tokens and other based on Ngrams, considering N equal 3. Overall, OPF stood out in not blocking legitimate messages and training time. In some bases the amount of spam classified correctly was high as well. The version that uses complete graph was better, although in some cases the version that makes use of *knn* graph outperformed it. Due to the current demands on security issues, OPF, considering its fast training time, can be improved in its effectiveness aiming at a real application. In relation to feature extraction methods, 3gram was better, improving OPF's results.

*Keywords*: Computer Network Security, Optimum-path Forest, Spam, Classification, Machine Learning.

# Lista de Figuras

# Lista de Tabelas

# Sumário

# Capítulo 1

# Introdução

Questões de *cyber* segurança têm atraído muita atenção ultimamente. Depois dos eventos desencadeados pelo *Stuxnet* [1] em 2010, pôde-se observar a criação de uma consciência cibernética na questão de segurança, principalmente por parte de grandes indústrias e governos. O fato é que o *Stuxnet* pode ser considerado um divisor de águas, visto que, até 2010, a grande maioria dos ataques realizados pela da internet envolviam questões financeiras: roubos de senhas de cartão de crédito, explorar alguma falha em sites para desvio de dinheiro e fraudes, por exemplo. No entanto, o *Stuxnet* mudou esse cenário. Ele se espalhou por meio dos sistemas Microsoft Windows e tinha como alvo softwares e equipamentos de automação industrial da empresa Siemens. A novidade não está em atacar sistemas industriais, mas sim em ser o primeiro *malware* que espia e subverte tais sistemas, sendo utilizado para atacar cinco organizações iranianas de enriquecimento de urânio. Este fato levou governos e indústrias a reforçarem suas estratégias de *cyber* segurança a fim de se protegerem.

Franke e Brynielsson [2] descrevem as estratégias de defesa cibernética utilizadas pelos governos de vários países ao redor do mundo. As medidas revelam o que os autores chamam de "consciência de situação cibernética" adotada por cada país em relação às ameaças atuais. É interessante notar as nuances de cada medida, evidenciadas nos exemplos que se seguem. O governo australiano instalou um centro de operações em segurança que visa facilitar respostas operacionais a eventos de segurança, enquanto que a França deseja antecipar e analisar efetivamente o ambiente a fim de tomar a decisão apropriada. A Estônia quer melhorar a eficiência em monitorar o tráfego de rede, e o governo do Reino Unido tem por objetivo continuar melhorando a detecção e análise de sofisticadas ameaças. Por fim, o governo russo almeja o desenvolvimento de um sistema governamental para detecção, advertência e mitigação de *cyber* ataques nos recursos de informação da federação russa. De modo geral, as estratégias adotadas por estes governos apresentam uma semelhança: elas visam rapidez e eficácia no tratamento de ameaças. Isso significa o desenvolvimento de sistemas que possam analisar ameaças num curto espaço de tempo, ou mesmo em tempo real, e preparar uma resposta à elas.

Originado provavelmente em 1994, o spam continua sendo uma ameaça bastante perigosa. Sua história é controversa, mas segundo o site Antispam.br[1], o termo foi originado de uma cena do programa *Monty Python's Flying Circus TV Show*, em que um grupo de *vikings* repete de maneira exaustiva uma frase sempre que alguém consulta o cardápio da taverna onde eles estão, sendo este todo formado com SPAM®, um presunto condimentado fabricado pela *Hormel Foods*. Entretanto, a prática iniciou-se quando dois advogados enviaram uma mensagem sobre uma loteria de *Green Cards* americanos para usuários da USENET[2], e o ato de enviar mensagem de propaganda para um fórum sem foco nesse assunto causou espanto e revolta nos assinantes

[1] antispam.br/historia/ <acesso em 21/03/2016>

[2] USENET é uma coleção de grupos de notícias em que usuários podem postar mensagens e estas podem ser distribuídas pelos servidores da USENET. Ver www.usenet.org <acesso em 21/03/2016>

do grupo. A situação se tornou mais drástica, porém, quando essa mesma mensagem foi enviada para vários grupos de discussão da USENET. Isso foi possível graças a um programa capaz de automatizar o envio em massa de mensagem de propaganda. A partir desse momento, toda mensagem recebida que não foi solicitada foi considerada spam.

A prática tomou contornos mundiais e hoje afeta o desempenho da rede mundial de computadores e toda rede interna de qualquer instituição. As diversas entidades envolvidas no combate à prática de spam frequentemente lançam estatísticas sobre como ela afeta o tráfego de dados. A Figura 1.1 mostra a quantidade de spam enviada por dia no período de 12 maio de 2015 até 12 de maio de 2016, e as estatísticas referem-se à uma pequena fração dos dados de spam computados pela Spamcop no mundo, porém ainda representativas.

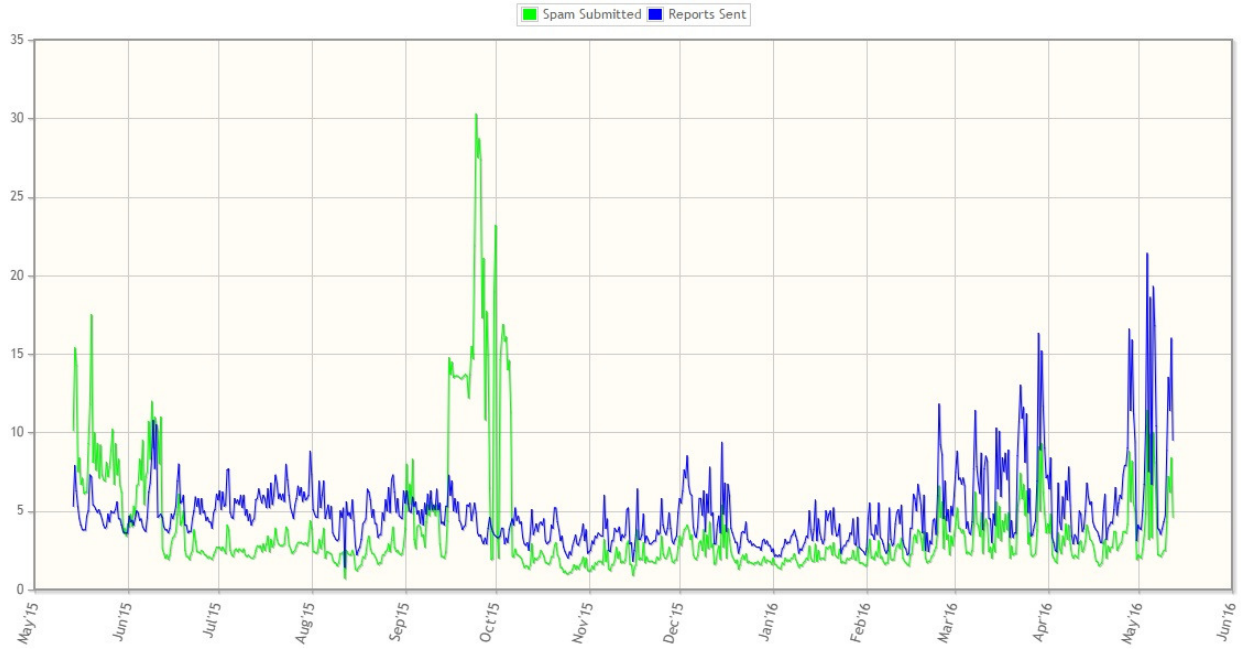


Figura 1.1: Quantidade de spams enviados por dia no período de 1 ano. Adaptado de: www.spamcop.net

Da Figura 1.1 é possível dizer que, em média, 3,9 mensagens de spam são enviadas por segundo, tendo atingido um pico máximo de 30,3 spams por segundo. No período considerado, um total de mais de um bilhão de spams foi enviado. A linha azul do gráfico indica o número de relatórios recebidos pela Spamcop diretamente relacionados aos spams enviados. É importante ressaltar que esses dados não representam o volume total de spams circulando pela internet, mas apenas uma amostra dos dados processados pela *SpamCop* a partir de seus pontos de monitoração de tráfego.

O site *Secure List*[3], especializado em ameaças cibernéticas, liberou um relatório online analisando a prática de spam no primeiro trimestre de 2015 [3]. Fica claro que spam está se tornando muito complexo em sua análise, detecção e prevenção, isto porque deixou de ser apenas o envio de mensagem não solicitada e passou a ser vetor de ataques mais sofisticados, como por exemplo *pishing*[4], que é baseado na curiosidade das pessoas. A Figura 1.2 apresenta os principais códigos maliciosos responsáveis por ataques de *pishing* oriundos de anexos de e-mails enviados como spam.

A Figura 1.2 apresenta nomes de alguns códigos maliciosos encontrados em anexos de e-mails e o primeiro deles foi projetado para, ao ser instalado ná máquina do usuário, baixar

[3]https://securelist.com/ <acesso em 22/03/2016>

[4]*Pishing* é o ato criminoso de fisgar dados sensitivos de um usuário, como senhas, dados bancários, entre outros.

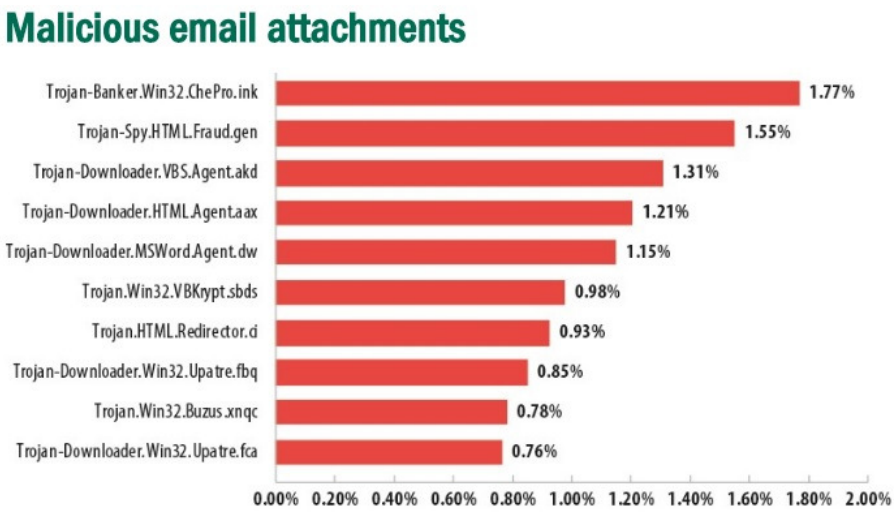


Figura 1.2: Anexos maliciosos de e-mail. Adaptado de: www.securelist.com

programas que roubam informações financeiras confidenciais e, em 2015, tinham como alvo bancos brasileiros e portugueses. O que contribui para o sucesso dessas ameças que trafegam junto dos spams é a sua grande proporção no volume de tráfego de e-mail. A Figura 1.3 apresenta a proporção de spam no tráfego de e-mail entre os meses de Outubro de 2014 a Março de 2015.

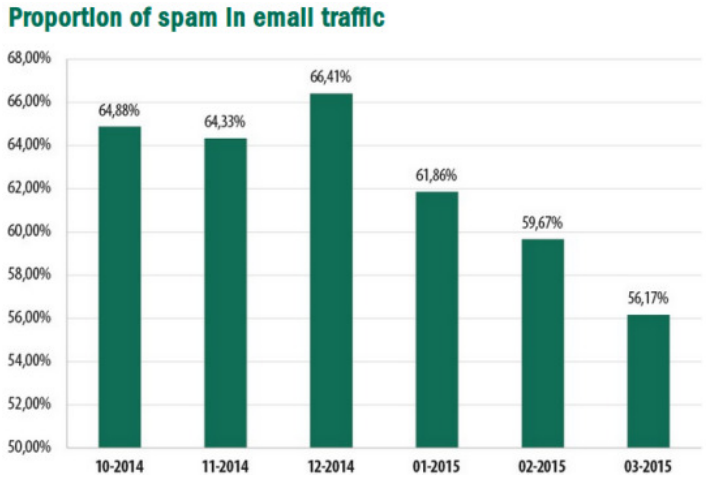


Figura 1.3: Proporção de spam em tráfego de e-mail. Adaptado de: www.securelist.com

É possível verificar que o volume de spam trafegado entre os e-mails legítimos é extremamente alto, superando este em todos os e-mails analisados, o que comprova a real necessidade de desenvolvimento de sistemas mais avançados na análise de mensagens, e o fato das ferramentas clássicas baseadas em assinatura serem efetivas apenas contra 30-50% das atuais ameaças de segurança colabora com isso. Diante dessa situação, torna-se necessário buscar outros métodos que atinjam melhor desempenho contra as ameaças atuais e, nesse contexto, algoritmos de aprendizado de máquina podem ser utilizados, os quais são definidos da seguinte forma, de acordo com Murphy: "um conjunto de métodos que podem detectar automaticamente os padrões nos dados, e então usar os padrões descobertos para prever dados futuros, ou para realizar outros tipos de tomada de decisão sob incerteza" [4]. Devido à complexidade e alta velocidade de mudança que as correntes ameaças apresentam, esse paradigma pode ser adaptado para a criação de ferramentas de combate. A abordagem de spam foi ampliada ao longo dos anos, deixando de ser algo exclusivo de e-mails e passando a utilizar outros vetores, como SMS

(*Short Message Service*), comentários em blogs e em vídeos da plataforma Youtube.

Dentro do cenário de aprendizado de máquina, é possível introduzir o conceito de classificadores baseados em Floresta de Caminhos Ótimos, que utilizam técnicas de aprendizado de máquina. Um classificador baseado em OPF se vale de um subconjunto de amostras, denominadas protótipos, e de uma função de custo de caminho no grafo de treinamento, cujo objetivo é agrupar amostras com atributos similares. O grafo de treinamento é, então, particionado em uma floresta de caminhos de custo mínimo, sendo cada protótipo a raiz de uma árvore e todas as amostras da árvore são rotuladas com o mesmo rótulo da raiz. Uma nova amostra é classificada quando encontra um protótipo que lhe ofereça um caminho ótimo dentre todos os protótipos, ou seja, sua classificação é baseada na força de conexidade entre a amostra e o protótipo mais conexo [5, 6].

## 1.1 Objetivos

O objetivo principal desta dissertação é avaliar os classificadores baseados em floresta de caminhos ótimos na detecção de mensagens de spam nos ambientes anteriormente mencionados, compará-los com outros classificadores no quesito tempo de treinamento, um fator diferencial hoje, e verificar qual o melhor método, seja por token ou por 3gram, para descrever uma mensagem como sendo spam ou ham, isto é, uma mensagem legítima.

## 1.2 Organização do Texto

O presente trabalho está organizado como segue. O Capítulo 2 apresenta a revisão bibliográfica de spam em suas abordagens aqui tratadas, ou seja, e-mail, SMS, comentários em blogs e em vídeos da plataforma Youtube. O Capítulo 3 apresenta a teoria a respeito do classificador OPF. O Capítulo 4 apresenta a metodologia que foi utilizada no desenvolvimento de cada um dos ambientes aqui estudados. O Capítulo 5 mostra os resultados obtidos. Por fim, o Capítulo 6 apresenta a conclusão.

# Capítulo 2

# Revisão Bibliográfica

Este capítulo apresenta as teorias descritas na literatura sobre spam em alguns de seus ambientes de propagação. Apesar da definição geral de spam ser a mesma para todos, algumas situações se diferem e, por isso, é necessária a apresentação das singularidades concernentes à cada ambiente.

## 2.1 Análise Contextual

Yang et al. [7] definem spam ou mensagem em massa não solicitada, ou ainda mensagem comercial não solicitada, como a prática de enviar mensagens, seja qualquer o vetor de envio utilizado, não desejadas, geralmente com conteúdo comercial, em grande quantidade, para um número indiscriminado de pessoas. Neste contexto, vários problemas podem ser relacionados à prática de spam, dentre os quais: desperdício de largura de banda e espaço de armazenamento, sobrecarga do servidor de e-mail, má fama associada ao e-mail como forma confiável de comunicação, *pishing* e conteúdo ofensivo [7, 8].

Faz-se necessário, antes de serem analisadas as técnicas para detecção de spam em seus variados vetores aqui estudados, analisar alguns detalhes econômicos relacionados à essa prática. Em 2012, Rao e Riley [9] publicaram um detalhado artigo analisando o impacto da prática de spam na economia mundial, tanto do lado para quem pratica spam quanto do lado de quem se defende dele, seja utilizando as próprias ferramentas ou a de terceiros. Os autores informaram que em 2010, 88% dos e-mails enviados no mundo eram spam, e quando é considerado que a estimativa de e-mails enviados diariamente seja da ordem de 90 bilhões, pode-se imaginar o custo envolvido nessa prática. Os autores foram além e, baseando-se nos preços de ferramentas de proteção, gastos com pessoal e todo processo envolvido com segurança, estimaram que, anualmente, as empresas americanas gastam cerca de 20 bilhões de dólares com spam, um valor alarmante. Por outro lado, os spammers, apenas considerando suas operações nos Estados Unidos da América, recebem um valor bruto estimado em 200 milhões de dólares.

Porém, isso não seria possível contando somente com o envio manual de spam por um spammer. A utilização de *botnets*, ou seja, uma rede de computadores "zumbi" infectada por um *malware* projetado para escravizar esses computadores à um computador mestre, é o maior aliado dos spammers. Como exemplo, a *"Rustock"*, a maior *botnet* conhecida, possuía a capacidade de enviar 30 bilhões de spam por dia antes de ser desabilitada em março de 2011. Essas informações mostram a urgente necessidade de desenvolvimento de técnicas para detecção de spam.

## 2.2 E-mail Spam

Ao longo dos anos, várias técnicas para filtragem de e-mail spam foram desenvolvidas. Kakade et al. [10] apresentam um levantamento das principais delas, as quais foram divididas em quatro principais classes: técnicas baseadas em aprendizado de máquina; técnicas baseadas em conteúdo; técnicas baseadas em listas e, por fim, aquelas classificadas como híbridas das anteriores ou que apresentam outro tipo de solução.

As técnicas para filtragem de spam baseadas em aprendizado de máquina são as mais comumente utilizadas e as que apresentam os melhores resultados, sendo as fortes bases matemáticas e alta exatidão na classificação os principais motivos para sua adoção. Dentre elas, podem ser citadas: K-vizinhos mais próximos, do inglês *K-nearest neighbors* (Knn), Redes Neurais Artificiais, do inglês *Artificial Neural Networks* (ANN), Máquinas de Vetores de Suporte, do inglês *Support Vector Machines* (SVM), e o classificador Naïve Bayes.

Knn é uma técnica caracterizada por sua natureza não paramétrica, no sentido de não fazer nenhuma consideração acerca da distribuição original dos dados, e o algoritmo de aprendizado é preguiçoso, pois sua função é aproximada localmente apenas e toda computação é adiada até a classificação. Todo novo e-mail é classificado como spam ou *ham*, baseado numa medida de distância entre as classes spam e *ham*.

Já as Redes Neurais são inspiradas no sistema nervoso biológico. Seu modelo de classificação é construído com base em aprendizado de exemplos similares. Uma rede neural irá convergir quando, durante a fase de treinamento, todas as instâncias no exemplo de entrada forem corretamente classificadas como spam ou *ham*, quando possível.

SVM é uma das técnicas mais utilizadas atualmente. Sua amostra de treinamento, assim como para os demais classificadores, é definida como um conjunto de vetores de $n$ atributos. Ao final da fase de treinamento, haverá um hiperespaço contendo um número de dimensões igual a $n' \gg n$, em que $n'$ denota o número de amostras de treinamento. No processo de filtragem de spam, são criados dois modelos, spam e *ham*. Essas duas classes são separadas por um hiperplano, sendo que toda instância de e-mail é tratada como um único ponto com dimensões no hiperespaço. O classificador Naïve Bayes, por sua vez, calcula a "spamicidade"[1] de uma palavra em e-mails autênticos e spams. Em seguida, é utilizada inferência Bayesiana para calcular a probabilidade de um e-mail ser ou não spam.

As técnicas baseadas em conteúdo avaliam um e-mail por meio de suas palavras, frases ou anexos na intenção de determinar se o mesmo é spam ou ham. Existem dois filtros que se valem dessa técnica: o baseado em palavras e o heurístico. O filtro baseado em palavras bloqueia e-mails baseado na "spamicidade" de algumas palavras, isto é, se um e-mail possui alguma palavra que frequentemente é utilizada em spam, ele é bloqueado. O problema dessa técnica é que se um filtro é configurado com palavras comuns, a taxa de falsos positivos aumenta consideravelmente. Por outro lado, os filtros heurísticos utilizam algum critério mais intuitivo do que simplesmente métricas técnicas. Pontuação ou *score* é, geralmente, um critério utilizado para classificar e-mail como spam ou *ham*. Para isso, mais pontos são dados a termos que mais frequentemente se encontram em spams e termos geralmente usados em e-mails legítimos possuem baixa pontuação. Depois de avaliar as palavras presentes em um e-mail, uma pontuação é calculada e atribuída a ele e, caso ultrapasse um limite preestabelecido, a mensagem é classificada como spam.

Técnicas baseadas em lista utilizam listas que mantém os nomes de emissores legítimos e emissores de spams, conhecido como *spammers*. Uma mensagem é bloqueada, então, de acordo com seu emissor. *Blacklist*, *Real Time BlackHole list*, *Whitelist* e *Greylist* são exemplos de técnicas baseadas em lista. Uma *Blacklist* contém uma lista de *spammers* e os spams são bloqueados pelo do nome de usuário ou IP *(Internet Protocol)* presente nessa lista. A

[1] Spamicidade de uma palavra é a porcentagem de ocorrências de uma palavra em mensagens de spams.

desvantagem é a taxa de falsos positivos, em que um *spammer* utiliza um IP ou usuário válido para mandar spam.

*Real Time Blackhole list* é semelhante à citada acima, com a diferença que sua lista é mantida por terceiros, o que garante uma frequência maior de atualização. No entanto, uma organização pode ter menor controle sobre uma lista e casos de falsos positivos. *Whitelist* é a técnica exatamente oposta à *Blacklist*, isto é, ela possui uma lista de apenas emissores legítimos sendo adequada quando o número de emissores de e-mail é fixo ou controlado. O sério prejuízo dessa técnica é um novo emissor, que não é um *spammer*, não poder enviar um e-mail, o que torna necessário, para sanar esse problema, a constante atualização da lista.

Por fim, uma *Greylist* é uma técnica nova que se baseia na ideia de que *spammers* enviam spam apenas uma vez. Assim, cada novo e-mail, inicialmente, é retido numa *Greylist* no servidor receptor e uma mensagem de falha é enviada ao emissor. Se não for um *spammer*, o servidor de e-mail emissor enviará a mensagem novamente e essa não será considerada spam. Caso ela não seja reenviada, será considerada como spam. A desvantagem é no caso de um e-mail ser uma emergência.

Finalmente, há ainda a classe de técnicas híbridas, que utilizam uma união daquelas acima apresentadas, ou aquelas outras que utilizam algum outro meio para filtragem de spam. Sistema desafio/resposta, filtros colaborativos, filtros baseados em imagens, filtros baseados em servidores SMTP *(Simple Mail Transfer Protocol)*, checagem de lista *DNS (Domain Name System) Blackhole*, sistemas *DNS Lookup* e o modelo *Bag-of-Words* se enquadram nessa classe. O sistema desafio/resposta trabalha da seguinte maneira: o servidor que gera o e-mail o envia para o servidor receptor. Então, este envia um desafio àquele. Se for um *spammer* não responderá, porque não será possível ou será um trabalho muito tedioso. Entretanto, se o servidor emissor resolver o desafio, este e-mail e todos os demais serão tratados como legítimos.

O filtro colaborativo é uma abordagem baseada em comunidade e os seus usuários decidem a natureza de um e-mail, podendo marcá-lo como spam ou *ham* configurando uma *flag* para isso. Quando a contagem de *flags* de um e-mail ultrapassa um limite estabelecido, o e-mail é então marcado como spam e bloqueado para outros usuários. Uma desvantagem é que cada nova comunidade é criada ignorando as demais. Caso uma delas abrigue um *spammer*, o contador do *flag* não será incrementado, resultando em spam sendo marcado como e-mail legítimo.

Os filtros baseados em imagens suprem uma falha dos anteriormente mencionados. Tanto o Sistema desafio/resposta quanto o filtro colaborativo analisam apenas palavras ou frases. Assim, *spammers* podem facilmente circundar essas defesas utilizando mensagens com imagens em spams. Para remediar essa falha, *Optical Character Recognition (OCR)* é utilizado para extrair texto de uma imagem e, então, podem ser utilizadas as técnicas já citadas.

Filtros baseados em servidores SMTP implementam ferramentas que combinam várias abordagens no processo de filtragem de spam. ASSP *(Anti-Spam STMP Proxy)* é uma ferramenta utilizada em servidores SMTP que combina as abordagens *Blackhole list* e filtros Bayesianos. Na técnica de checagem de lista *DNS Blackhole*, o servidor de e-mail receptor mantém uma lista baseada em DNS de servidores presente numa *blackhole list*. Assim, o endereço IP de um emissário é checado nessa lista. Se ele pertencer a ela, o e-mail é marcado como spam e não será entregue a seu destinatário.[2]

Sistemas *DNS Lookup* utilizam um recurso do DNS, o *Mail Exchange record (MX)*, que especifica um servidor de e-mail responsável por aceitar mensagens de e-mail de um destinatário em nome de um domínio para efetivamente filtrar spam. Baseados no registro MX, servidores DNS verificam o nome de um servidor de e-mail como válido ou não. Caso ele não possua um registro MX válido, os e-mails enviados desse domínio serão tratados como spams e não serão enviados a seus destinatários.

[2]A lista de servidores de e-mail está disponível online no site: http://www.dnsbl.info/ <acesso em 24/07/2015>

Por fim, no modelo *Bag-of-Words*, cada palavra no e-mail é listada em um documento e então associada com um índice. Assim, a frequência de ocorrência de cada palavra é utilizada como uma característica para treinar um classificador. As palavras mais comumente encontradas em spam vão para uma *bag*, que se comporta como uma classe, denominada spam. As outras palavras ficam em uma *bag* de e-mails legítimos. Dessa forma, um classificador pode classificar um e-mail verificando em qual *bag* cada palavra desse e-mail se encontra.

Reforçando o uso de técnicas de aprendizado de máquina como aquelas que obtém os melhores resultados na filtragem de spam, Almeida e Yamakami [11] apresentaram os avanços em técnicas de filtragem de spam. Os autores apresentaram e compararam sete versões do classificador Naïve Bayes, Máquinas de Vetores de Suporte e um novo método por eles proposto baseado no princípio *Minimum Description Lenght (MDL)*. Os experimentos foram conduzidos sobre bases de dados públicas e não codificadas, e os resultados mostraram que o método proposto se sobrepôs aos filtros de spam do estado da arte.

O princípio MDL é um poderoso método de inferência indutiva, que é a base para modelagem estatística, reconhecimento de padrões e aprendizado de máquina. Os métodos MDL são particularmente adequados para lidar com seleção de modelo, predição e problemas de estimação em situações em que os modelos sob consideração podem ser arbitrariamente complexos e o ajuste dos dados é uma preocupação séria [12].

Um filtro de spam baseado no princípio MDL trabalha de acordo com os seguintes passos:

- Tokenização: o classificador extrai todos os $m$ termos de uma nova mensagem $M = \{t_1, \cdots, t_m\}$, em que $t_i$ denota o termo $i$;

- Computar o aumento do comprimento da descrição quando $M$ é atribuído para cada classe $c \in \{spam, ham\}$:

$$L_M(spam) = \sum_{i=1}^{m} \left\lceil -log_2 \left( \frac{n_{spam}(t_i) + \frac{1}{|\phi|}}{n_{spam} + 1} \right) \right\rceil \tag{2.1}$$

$$L_M(ham) = \sum_{i=1}^{m} \left\lceil -log_2 \left( \frac{n_{ham}(t_i) + \frac{1}{|\phi|}}{n_{ham} + 1} \right) \right\rceil \tag{2.2}$$

  em que $n_{spam}(t_i)$ é o número de vezes que o termo $t_i$ aparece em mensagens da classe spam, $n_{ham}(t_i)$ é o número de vezes que o termo $t_i$ aparece em mensagens da classe *ham* e $|\phi|$ é o tamanho do vocabulário, ou seja, cada termo é representado por um símbolo de 32 bits, ou $2^{32}$, no caso.

- Se $L_M(spam) > L_M(ham)$, então $m$ é classificado como spam; caso contrário, $m$ será rotulado como ham.

- Método de treinamento: é empregado o TONE (*train on or near error*), o qual é iniciado com um modelo vazio, classifica cada nova amostra e treina novamente, mesmo se a classificação foi correta. Isso acelera o processo de aprendizado expondo o filtro a adicionais amostras difíceis de serem classificadas no mesmo período de treinamento.

As bases de dados utilizadas por Almeida e Yamakami [11] foram obtidas do conjunto *Enron* [13]. O modelo proposto superou todos os comparados no artigo, atingindo uma precisão maior do que 95% para todas as bases.

## 2.3 SMS Spam

Acerca de SMS, fazem-se necessárias algumas considerações antes de serem apresentados os trabalhos já desenvolvidos nessa área. O serviço, iniciado em 1992, ganhou grande apreço público e hoje é um dos mais utilizados na categoria de troca de mensagens. Seu crescimento é sustentado pelos altos retornos financeiros obtidos pois, de acordo com a empresa Portio Research [14], o faturamento mundial com SMS atingiu a marca de 128 bilhões de dólares em 2011, e estimaram a receita para 2016 em mais de 153 bilhões de dólares. Além disso, o mesmo relatório indica que em 2009 foram enviados 9,5 trilhões de SMS no mundo.

Este cenário de grande uso e rentabilidade tornou-se propício à prática de spam via SMS e, o fato de o valor cobrado por SMS ser muito baixo, chegando até mesmo ser gratuito em alguns lugares, contribuiu para isso. Soma-se à este quadro, também, a falta de software para filtragem de spam para lidar com tal ameaça por parte das operadoras, o que torna essa prática viável. Incipiente nas Américas e Europa, o volume de spams transmitidos via SMS na Ásia é elevado, chegando a corresponder à 30% do volume de tráfego [15].

Para lidar com essa ameaça, pesquisadores têm empregado esforços aplicando diversas técnicas no combate a SMS spam. Um método baseado em CAPTCHA é apresentado por Shirali-Shahreza e Shirali-Shahreza [16]. Neste método, quando uma mensagem de SMS é recebida, um teste CAPTCHA é preparado e enviado ao emissor como um SMS, em que é enviada uma imagem e uma questão de múltipla escolha, na qual o emissor deverá escolher a resposta que represente a imagem, enviando-a de volta. Caso a resposta esteja correta, o emissor é considerado legítimo e a mensagem SMS original é entregue. O grande problema desse método é falta de praticidade. Dada a quantidade supracitada de SMS trafegados por ano, esse método é impraticável. Assim, soluções baseadas em aprendizado de máquina são empregadas para automaticamente detectar SMS spam.

Uma solução que utiliza os classificadores Bayesianos foi apresentada por Gunal et al. [17]. Os autores empregaram duas abordagens de extração de características, a saber: *Information Gain* e Métricas Qui-quadrado, e então, utilizaram dois classificadores Bayesianos diferentes para classificar mensagens SMS. Já Nuruzzaman et al. [18] criaram um algoritmo para treinamento, teste e atualização direto no celular. Os autores utilizaram Naive Bayes e SVM para realizar a classificação das mensagens. Em ambos os trabalhos, os resultados obtiveram boa acurácia, atingindo mais de 90% de taxa de reconhecimento de SMS spam.

## 2.4 Blog Spam

Com relação aos Blogs, importa dizer que a popularidade dos mesmos cresceu muito nos últimos anos. Um Blog nada mais é que um local na internet onde seu autor escreve um artigo e seus leitores podem fazer comentários acerca dele. Dado esse aumento de popularidade, os *spammers* viram uma nova possibilidade de realizarem suas práticas. Para termos noção da nocividade dessa prática, o *Akismet*[3] liberou um relatório sobre o mês de abril de 2015 que informa que, do total de comentários publicados em blogs que são monitorados por essa ferramenta, mais de 4 bilhões são spam, sendo apenas 3,4% comentários legítimos. Esse fato é alarmante, e demonstra a real necessidade de ferramentas capazes de lidarem com essa situação em um curto período de tempo.

Entretanto, o serviço oferecido pelo *Akismet*, assim como *LinkSleeve*[4] e *Defensio*[5], é um método que depende de técnicas colaborativas, enviando os comentários a um servidor central

[3]Akismet é um serviço para filtrar comentários em blogs. Disponível em:https://akismet.com/ <acesso em 23/02/2016>

[4]Disponível em: http://www.linksleeve.org/ <acesso em 23/02/2016>

[5]Disponível em http://www.websense.com/content/facebook.aspx <acesso em 23/02/2016>

que realiza determinados testes para aferir se os comentários são spam. Essa abordagem é eficiente em relação à spams enviados por uma botnet, porém há muitos usuários reais que desejam manualmente espalhar spam. Essa situação exige um novo paradigma no tratamento dos comentários que são spams e, novamente, os conceitos relacionados à aprendizado de máquina surgem como a opção mais viável, apesar de não serem as únicas. Teraguchi et al. [19] adaptaram um filtro Bayesiano utilizado no tratamento de mensagens de e-mail para utilizá-lo em comentários de postagem de blogs. Inicialmente, eles criaram um dicionário manualmente, chamado de dicionário de probabilidade de spam, composto por um par, uma palavra e sua probabilidade de aparecer em mensagem de spam. A partir desse dicionário, o filtro Bayesiano foi utilizado para classificar os comentários e, quando surgia um erro, proveniente de mensagens de spam que não utilizam palavras até então ausentes no dicionário, um processo de extração dessas palavras, adição no dicionário e retreinamento eram realizados, mantendo o dicionário atualizado com o que está sendo utilizado no mundo real. Eles criaram inicialmente 4 diferentes dicionários, denominados A, B, C e D, e calcularam a porcentagem de detecção de spam utilizando cada um deles, sendo que o dicionário D foi o que atingiu o melhor resultado, com 80% de acerto.

Seguindo uma linha semelhante, Almeida e Alberto [20] compararam os seguintes classificadores: Naïve Bayes, SVM, usando kernel Linear, kernel RBF e kernel Polinomial, *Decision Trees*; *Randon Forest*; KNN; *Boosted* Naïve Bayes; *Boosted Decision Trees* e *Multinomial Logistic Regression*. Para tanto, utilizaram uma base de dados disponibilizada por Mishne et al. [21] contendo 1.024 mensagens, das quais 697 são spams e 327 são hams. No processo, os autores realizaram três experimentos, sendo o primeiro com os atributos extraídos das mensagens de texto, o segundo com os atributos extraídos dos metadados da postagem, e o terceiro como sendo a união dos dois primeiros. A conclusão foi de que a combinação dos dois experimentos obteve os melhores resultados, sendo *Multinomial Logistic Regression* e SVM com kernel polinomial os classificadores que obtiveram os melhores resultados.

Uma outra abordagem foi seguida por Radulescu et al. [22], em que os autores propuseram um método para detecção de spam em comentários de posts em blogs utilizando processamento de linguagem natural, buscando, para tanto, fluxo descontínuo de texto, linguagem vulgar e inadequada ou não relacionada a um contexto específico, e comentários incoerentes com um acentuado número de pontuação, novas linhas, espaços em branco e caracteres que não são ASCII. A arquitetura do sistema proposto é composta por três módulos: extração de características, extração de tópico e similaridade de comentário em post. No primeiro módulo, é realizada a extração de todas as palavras presentes em um conjunto de dados rotulado. O módulo de extração de tópico indica o quanto um tópico é citado nos comentários de um post. Por fim, o último módulo detecta se um comentário e um post contêm tópicos similares. Os experimentos foram conduzidos utilizando-se comentários extraídos do site de notícias *Daily Telegraph* e o classificador *Decision Trees* foi utilizado. Sua avaliação se deu por meio das métricas *precision* e *recall*, sendo a primeira da ordem de 95.83% e a segunda da ordem de 83.66%.

## 2.5 Social Spam

Por fim, será analisado o último vetor explorado neste trabalho: as mídias sociais, mais precisamente a plataforma Youtube. A prática de spam nesse meio é conhecida como social spam. O Youtube, um serviço de vídeos lançado em maio de 2005, é o maior do mundo em seu ramo de atuação. Segundo reportado pelo Google[6], o Youtube possui mais de 1 bilhão de usuários, 80% de sua audiência vêm de fora dos EUA e mais de 300 horas de vídeo são enviadas por

[6]https://www.youtube.com/yt/press/statistics.html <acesso em 10/03/2016>

minuto, gerando bilhões de visualizações. Todas essas estatísticas se configuram como um forte apelo aos spammers na prática de spam, tanto que a empresa de segurança computacional Nextgate informou em seu relatório "*2013 State of Social Media Spam*" [7] que somente no primeiro semestre de 2013 o volume de *social spam*, prática de spam realizada em mídias sociais como Facebook, Twitter e Youtube, apresentou um crescimento de 355%. Além disso, o número de usuários *fakes* aumentaram consideravelmente, sendo que, para cada 7 novos usuários cadastrados nas redes sociais, 5 eram *spammers*.

Dessa maneira, há a urgente necessidade de se filtrar os comentários realizados nessa plataforma. Várias abordagens estão sendo apresentadas por diversos pesquisadores ao redor do mundo. Radulescu et al. [22], além do trabalho feito com comentários em blogs, também utilizam a mesma abordagem para comentários em vídeos do Youtube. Utilizando a mesma metodologia já explicada, os autores obtiveram resultados menos satisfatórios, sendo que a precisão ficou com 60% e a revocação com 54.55%, evidenciando que o processamento de linguagem natural, para esse caso, não é adequado.

Buscando soluções mais aprimoradas, Madden et al. [23] propuseram um esquema de classificação para categorizar comentários no Youtube. Este esquema é baseado na análise qualitativa de conteúdo. A análise de conteúdo é uma metodologia de pesquisa que envolve fazer o conteúdo de mensagens se manifestar popr meio da identificação de características de uma forma tão objetiva quanto possível. De posse dessa metodologia, os autores criaram 10 categorias e 58 subcategorias. Dentre as categorias, cabe destacar a "comentários que não são respostas", visto que uma de suas subcategorias é "spam". A ideia foi criar esse esquema e apresentá-lo como possível solução para o problema de spam em comentários.

De forma mais objetiva, Alberto et al. [24] criaram um conjunto de dados composto por comentários aleatoriamente extraídos dos 5 vídeos mais visualizados de todos os tempos do Youtube, denominado pelos autores da seguinte forma: Psy, Shakira, LMFAO, Katy Perry e Eminem. Os autores descartaram os metadados e criaram uma *bag-of-words*, ou seja, todas as palavras extraídas dos comentários dos vídeos. Eles utilizaram a frequência com que cada termo aparece em um determinado comentário. Os autores compararam 10 versões de diferentes classificadores, a saber: *Decision Trees*, KNN, Regressão Logística, *Bernoulli* Naïve Bayes, *Gaussian* Naïve Bayes, *Multinomial* Naïve Bayes, *Randon Forest*, SVM com kernel Linear, SVM com kernel Polinomial e SVM com kernel Gaussiano. Para Psy, SVM com kernel Gaussiano apresentou os melhores resultados; para Katy Perry *Randon Forest* se apresentou como o melhor classificador; já para LMFAO, *Bernoulli* Naïve Bayes atingiu os melhores resultados; *Decision Trees* foi o classificador que obteve os melhores resultados para Eminem, e *Multinomial* Naïve Bayes apresentou os melhores resultados para Shakira.

[7] Disponível em: http://goo.gl/wSNTaZ <acesso em 10/03/2016>

# Capítulo 3

# Floresta de Caminhos Ótimos

Este Capítulo apresenta a teoria a respeito do OPF em suas versões utilizando grafo completo e grafo $k$-nn.

O classificador OPF modela o problema de reconhecimento de padrões como uma tarefa de partição de um grafo, no qual o vetor de características extraído de cada amostra é considerado um nó desse grafo. Os arcos são definidos por alguma relação de adjacência predefinida e ponderados por uma métrica de distância aplicada à seus correspondentes vetores de características. A principal ideia dos classificadores baseados em OPF é comandar um processo de competição entre algumas amostras-chave (protótipos) no intuito de conquistar os demais nós. Tal processo particiona o grafo em árvores de caminhos ótimos, do inglês *optimum-path trees (OPTs)*, enraizadas em cada protótipo, definindo, assim, regiões discretas de influência para cada OPT. O processo de competição é realizado por meio de funções de custo de caminho oferecidas à cada amostra pelos protótipos. É importante observar que diferentes funções de custo de caminho levam a diferentes partições no grafo.

Papa et al. [5, 6, 25] apresentaram duas diferentes versões supervisionadas do classificador OPF, sendo que a primeira faz uso de um grafo completo ($\text{OPF}_{cpl}$) [5, 6] e a segunda emprega um grafo $k$-nn ($\text{OPF}_{knn}$) [25]. Ambas versões operam similarmente, isto é, empregam o mesmo algoritmo OPF, porém com as seguintes modificações: (i) relação de adjacência, (ii) metodologia para estimar os protótipos, e (iii) função de custo de caminho. As Seções 3.1 e 3.2 apresentam em maiores detalhes as versões com grafo completo e com grafo $k$-nn, respectivamente.

## 3.1 OPF com Grafo Completo

Seja $\mathcal{Z}$ uma base de dados tal que $\mathcal{Z} = \mathcal{Z}_1 \bigcup \mathcal{Z}_2$, em que $\mathcal{Z}_1$ e $\mathcal{Z}_2$ representam o conjunto de treinamento e teste, respectivamente. Cada amostra $s \in \mathcal{Z}$ pode ser representada pelo seu vetor de características $\overrightarrow{v}(s) \in \Re^n$ e mapeado para um nó de um grafo. Seja $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ tal grafo no qual $\mathcal{A}$ é uma relação de adjacência que conecta todos os pares de nós, isto é, um grafo completamente conectado, e $\mathcal{V}$ contém os vetores de características $\overrightarrow{v}(s)$, $\forall s \in \mathcal{Z}$. Para fins de explicação, $\overrightarrow{v}(s)$ será referido como sendo $\mathbf{s}$. Além disso, seja $\lambda(\cdot)$ uma função que atribui um rótulo verdadeiro para cada amostra em $\mathcal{Z}$.

Considere um caminho $\pi_s$ em $\mathcal{G}$ com término em $\mathbf{s}$, e uma função $f(\pi_s)$ que associa um valor a esse caminho. O algoritmo $\text{OPF}_{cpl}$ visa minimizar $f(\pi_s)$ para cada amostra $\mathbf{s}$ usando uma função que computa o maior peso de um arco ao longo do caminho, isto é:

$$\begin{aligned} f_{max}(\langle \mathbf{s} \rangle) &= \begin{cases} 0 & \text{se } \mathbf{s} \in S, \\ +\infty & \text{caso contrário} \end{cases} \\ f_{max}(\pi \cdot \langle \mathbf{s}, \mathbf{t} \rangle) &= \max\{f_{max}(\pi), d(\mathbf{s}, \mathbf{t})\}, \end{aligned} \tag{3.1}$$

em que $\pi \cdot \langle \mathbf{s}, \mathbf{t} \rangle$ denota a concatenação do caminho $\pi_s$ e o arco $\langle \mathbf{s}, \mathbf{t} \rangle$, e $d(\mathbf{s}, \mathbf{t})$ mede a dissimilaridade entre nós adjacentes. O conjunto $\mathcal{S} \subseteq \mathcal{V}$ se refere aos protótipos. Dado que usualmente tem-se o conjunto de treinamento e o conjunto de testes, o OPF$_{cpl}$ também possui uma fase de treino e uma fase de teste, sendo a primeira encarregada de construir uma floresta de caminhos ótimos por meio de um processo de competição usando $f_{max}$, e a segunda é usada para avaliar seu desempenho.

### 3.1.1 Treinamento

Seja $\mathcal{S}^* \subset \mathcal{S}$ um conjunto de protótipos que iniciam um processo de competição. Papa et al. [5] propuseram selecionar tais amostras como sendo os elementos mais próximos de classes diferentes, isto é, posicionar tais amostras-chave nas regiões mais propensas a erros de classificação (fronteiras das classes). Com o objetivo de atingir esse propósito, pode-se computar uma árvore geradora mínima, do inglês *minimum spanning tree* (MST), no grafo derivado do conjunto de treinamento, isto é, $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A})$, em que $\mathcal{V}_1$ contém todos os vetores de características extraídos das amostras de treinamento. Além disso, pode-se simplesmente marcar os elementos conectados de diferentes classes na MST como nós protótipos, compondo, dessa maneira, o final $\mathcal{S}^*$. Note que este conjunto é composto de amostras de treinamento apenas. A fase de treinamento, então, tem como saída uma floresta de caminho ótimo sobre $\mathcal{G}_1$ como implementado pelo *Algoritmo* 1.

**Algoritmo 1** – OPF$_{cpl}$ Fase de Treinamento

| | |
|---|---|
| Entrada: | Um grafo de treinamento $\mathcal{V}_1$ $\lambda$-rotulado, protótipos $\mathcal{S}^* \subset \mathcal{V}_1$ e o par $(v, d)$ para vetor de característica e cálculo das distâncias. |
| Saída: | Floresta de Caminhos Ótimos $P$, mapa de custo de valor $V$ e mapa de rótulos $L$ |
| Auxiliares: | Fila de prioridade $Q$, e variável *cst*. |

1. **Para toda** $s \in \mathcal{V}_1$, **seja** $P(s) \leftarrow nil$ *e* $V(s) \leftarrow +\infty$.
2. **Para todo** $s \in \mathcal{S}^*$, **seja** $V(s) \leftarrow 0$, $P(s) \leftarrow nil$, $L(s) = \lambda(s)$ *e adicione* $s$ *em* $Q$.
3. **Enquanto** $Q$ *é não vazia,* **faça**
4. $\quad$ *Remova de* $Q$ *uma amostra* $\boldsymbol{s}$ *tal que* $V(\boldsymbol{s})$ *é mínimo.*
5. $\quad$ **For cada** $\boldsymbol{t} \in \mathcal{V}_1$ *tal que* $\boldsymbol{s} \neq \boldsymbol{t}$ *e* $V(\boldsymbol{t}) > V(\boldsymbol{s})$, **faça**
6. $\quad\quad$ *Compute* $cst \leftarrow \max\{V(\boldsymbol{s}), d(\boldsymbol{s}, \boldsymbol{t})\}$.
7. $\quad\quad$ **Se** $cst < V(\boldsymbol{t})$, **então**
8. $\quad\quad\quad$ **Se** $V(\boldsymbol{t}) \neq +\infty$, **então** *remova* $\boldsymbol{t}$ *de* $Q$.
9. $\quad\quad\quad$ $P(\boldsymbol{t}) \leftarrow \boldsymbol{s}$, $L(\boldsymbol{t}) \leftarrow L(\boldsymbol{s})$ *e* $V(\boldsymbol{t}) \leftarrow cst$.
10. $\quad\quad\quad$ *Insira* $t$ *em* $Q$.

As linhas 1 - 2 inicializam todas as amostras $\mathbf{s} \in \mathcal{V}_1$ e protótipos $\mathbf{s} \in \mathcal{S}^*$, os mapas de valores de custo de caminho e de rótulos. Os precursores são inicializados como *nil*. Das linhas 5 - 10, o algoritmo verifica se o caminho que atinge uma amostra adjacente $\mathbf{t}$ por meio de $\mathbf{s}$ possui um custo menor em relação ao caminho que termina em $\mathbf{t}$. Se for o caso, $\mathbf{t}$ é conquistado por $\mathbf{s}$, o que torna $\mathbf{s}$ o precursor de $\mathbf{t}$. Portanto, a amostra $\mathbf{t}$ recebe o rótulo da amostra $\mathbf{s}$ e seu valor do custo é atualizado. Este mesmo processo é repetido para todas as amostras de treinamento nas linhas 3 - 10.

### 3.1.2 Classificação

Seja $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{A})$ o grafo derivado do conjunto de teste $\mathcal{Z}_2$. Para cada amostra $\mathbf{t} \in \mathcal{V}_2$, consideram-se todos os arcos conectando $\mathbf{t}$ com amostras $\mathbf{s} \in \mathcal{V}_1$, tornando $\mathbf{t}$, assim, parte do grafo original. Considerando todos os caminhos possíveis entre $\mathcal{S}^*$ e $\mathbf{t}$ deseja-se encontrar

um caminho ótimo $P^*(\mathbf{t})$ de $\mathcal{S}^*$ até $\mathbf{t}$ com a classe $\lambda(R(\mathbf{t}))$ de seu protótipo $R(\mathbf{t}) \in \mathcal{S}^*$ mais fortemente conectado. Este caminho pode ser incrementalmente identificado avaliando o valor de custo ótimo $V(\mathbf{t})$ da seguinte maneira:

$$V(\mathbf{t}) = \min_{\forall s \in \mathcal{V}_1} \{\max\{V(\mathbf{s}), d(\mathbf{s},\mathbf{t})\}\}. \tag{3.2}$$

## 3.2 OPF com Grafo $k$-nn

Papa e Falcão [25] introduziram uma variante do OPF supervisionado apresentado na seção anterior na qual tanto os arcos quantos os nós são ponderados. Nesta variante, denominada OPF$_{knn}$, a relação de adjacência é agora definida por um grafo de $k$-vizinhos mais próximos, isto é, $\mathcal{G} = (\mathcal{V}, \mathcal{A}_k)$, em que $\mathcal{A}_k$ representa os $k$-vizinhos mais próximos de cada amostra em $\mathcal{V}$. Tal abordagem pode ser vista como um problema dual do OPF$_{cpl}$, visto que agora o objetivo é maximizar o custo de cada amostra em $\mathcal{V}$, e os protótipos são agora posicionados nas regiões com altas concentrações de amostras. Novamente, esta abordagem é composta por uma fase de treinamento e uma fase de teste, como descrito a seguir.

### 3.2.1 Treinamento

A fase de treinamento do OPF$_{knn}$ visa construir uma floresta de caminhos ótimos usando um algoritmo similar àquele apresentado na seção anterior (*Algoritmo* 1), mas usando uma diferente função de custo de caminho e relação de adjacência. Como mencionado anteriormente, o OPF$_{knn}$ posiciona os protótipos nas regiões com alta densidade de amostras. A fim de alcançar esse propósito, uma função densidade de probabilidade, do inglês *probability density function* (pdf), é associada a cada nó do grafo de treinamento $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A}_k)$ da seguinte maneira:

$$\rho(\mathbf{s}) = \frac{1}{\sqrt{2\pi\sigma^2|A(\mathbf{s})|}} \sum_{\forall t \in A(\mathbf{s})} \exp\left(\frac{-d^2(\mathbf{s},\mathbf{t})}{2\sigma^2}\right), \tag{3.3}$$

em que $\sigma = \frac{d_f}{3}$ e $d_f$ é o maior comprimento de arco em $\mathcal{G}_1$. O OPF$_{knn}$ visa maximizar a função de custo de caminho para todas as amostras de treinamento de acordo com $f_{min}$, como segue:

$$\begin{aligned} f_{min}(\langle\mathbf{t}\rangle) &= \begin{cases} \rho(\mathbf{t}) & \text{se } \mathbf{t} \in \mathcal{S}, \\ \rho(\mathbf{t}) - \delta & \text{caso contrário} \end{cases} \\ f_{min}(\pi_s \cdot \langle\mathbf{s},\mathbf{t}\rangle) &= \{ \min\{f_{min}(\pi_s), \rho(\mathbf{t})\}, \end{aligned} \tag{3.4}$$

em que $\delta$ é um número suficientemente pequeno que evita a divisão da zona de influência em múltiplas zonas de influência. O *Algoritmo* 2 implementa os principais conceitos do OPF$_{knn}$.

Como mencionado anteriormente, o OPF$_{knn}$ trabalha de forma similar ao OPF$_{cpl}$, mas agora a ideia é maximizar o mapa de custo, ao invéz de minimizá-lo. Entretanto, o principal problema com $f_{min}$ simples refere-se ao fato de que ele não pode garantir, pelo menos, um protótipo por classe. Para lidar com essa deficiência, uma versão modificada de $f_{min}$ é utilizada, como mostrada a seguir:

$$\begin{aligned} f^*_{min}(\langle\mathbf{t}\rangle) &= \begin{cases} \rho(\mathbf{t}) & \text{se } \mathbf{t} \in \mathcal{S} \\ \rho(\mathbf{t}) - \delta & \text{caso contrário} \end{cases} \\ f^*_{min}(\pi_s \cdot \langle\mathbf{s},\mathbf{t}\rangle) &= \begin{cases} -\infty & \text{se } \lambda(\mathbf{t}) \neq \lambda(\mathbf{s}) \\ \min\{f_{min}(\pi_s), \rho(\mathbf{t})\} & \text{caso contrário.} \end{cases} \end{aligned} \tag{3.5}$$

**Algoritmo 2** – OPF$_{knn}$ Algoritmo Principal

| | |
|---|---|
| Entrada: | Um grafo de treinamento $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A}_k)$, $\lambda(\mathbf{s})$ para todo $\mathbf{s} \in \mathcal{V}_1$ e uma função de valor de caminho $f_{min}$. |
| Saída: | Mapa de rótulos $L$, mapa de valores de caminho $V$ e floresta de faminhos ótimos $P$. |
| Auxiliares: | Fila de prioridade $Q$ e a variável *tmp*. |

1. **Para cada** $\boldsymbol{s} \in \mathcal{V}_1$, **seja** $P(\boldsymbol{s}) \leftarrow nil$, $V(\boldsymbol{s}) \leftarrow \rho(s) - \delta$
$L(\boldsymbol{s}) \leftarrow \lambda(\boldsymbol{s})$ *e Insira* $\boldsymbol{s}$ *em* $Q$.
2. **Enquanto** $Q$ *é não vazia,* **faça**
3. *Remova de* $Q$ *uma amostra* $\boldsymbol{s}$ *tal que* $V(\boldsymbol{s})$ *é máximo.*
4. **Se** $P(\boldsymbol{s}) = nil$, **então**
5. $V(\boldsymbol{s}) \leftarrow \rho(\boldsymbol{s})$.
6. **Para cada** $\boldsymbol{t} \in \mathcal{A}_k(\boldsymbol{s})$ *e* $V(\boldsymbol{t}) < V(\boldsymbol{s})$, **faça**
7. $tmp \leftarrow \min\{V(\boldsymbol{s}), \rho(\boldsymbol{t})\}$.
8. **Se** $tmp > V(\boldsymbol{t})$ **então**
9. $L(\boldsymbol{t}) \leftarrow L(\boldsymbol{s})$, $P(\boldsymbol{t}) \leftarrow \boldsymbol{s}$, $V(\boldsymbol{t}) \leftarrow tmp$.
10. *atualize a posição de* $\boldsymbol{t}$ *em* $Q$.

A Equação 3.5 pondera todos os arcos $(\mathbf{s}, \mathbf{t}) \in \mathcal{A}_k$ tal que $\lambda(\mathbf{t}) \neq \lambda(\mathbf{s})$ com $-\infty$ prevenindo, dessa maneira, que tais arcos pertencam a algum caminho ótimo (isto é, evita que uma amostra $\mathbf{t}$ seja conquistada por uma amostra $\mathbf{s}$ de outra classe).

Finalmente, o OPF$_{knn}$ precisa ajustar o parâmetro $k$, que controla o tamanho da janela utilizada para calcular a pdf de cada nó na fase de treinamento. Papa e Falcão [26] propuseram uma busca exaustiva pelo $k \in [1, k_{max}]$ que maximiza a acurácia no conjunto de treinamento, sendo $k_{max}$ um parâmetro definido pelo usuário. De maneira simples, a ideia é usar o valor de $k^*$ que maximiza a acurácia do OPF$_{knn}$ no conjunto de treinamento. O *Algoritmo* 3 implementa este procedimento.

**Algoritmo 3** – Treinamento

| | |
|---|---|
| Entrada: | Grafo de treinamento $\mathcal{G}_1$, $\lambda(s)$ para todo $\mathbf{s} \in \mathcal{V}_1$, $k_{max}$ e funções de valores de caminho $f_{min}$ e $f^*_{min}$. |
| Saída: | Mapa de rótulos $L$, mapa de valores de caminho $V$ e Floresta de Caminhos Ótimos $P$. |
| Auxiliares: | variáveis $i$, $k$, $k^*$, $MaxAcc \leftarrow -\infty$, $Acc$ e vetores $FP$ e $FN$ de tamanho $c$. |

1. *Para* $k = 1$ *até* $k_{max}$ *faça*
2. *Crie o grafo* $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A}_k)$ *com nós ponderados pela Equação 3.3.*
3. *Calcule* $(L, V, P)$ *usando o Algoritmo 2 com* $f_{min}$.
4. *Para cada classe* $i = 1, 2, \ldots, c$, *faça*
5. $FP(i) \leftarrow 0$ *e* $FN(i) \leftarrow 0$.
6. *Para cada amostra* $\boldsymbol{t} \in Z_1$, *faça*
7. *Se* $L(\boldsymbol{t}) \neq \lambda(\boldsymbol{t})$, *então*
8. $FP(L(\boldsymbol{t})) \leftarrow FP(L(\boldsymbol{t})) + 1$.
9. $FN(\lambda(\boldsymbol{t})) \leftarrow FN(\lambda(\boldsymbol{t})) + 1$.
10. *Calcule acurácia.*
11. *Se* $Acc > MaxAcc$, *então*
12. $k^* \leftarrow k$ *e* $MaxAcc \leftarrow Acc$.
13. *Destrua o grafo* $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A}_k)$.
14. *Crie o grafo* $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A}_{k^*})$ *com nós ponderados pela Equação 3.3.*
15. *Calcule* $(L, V, P)$ *usando o Algoritmo 2 com* $f^*_{min}$.

### 3.2.2 Classificação

Uma amostra $\mathbf{t} \in \mathcal{V}_2$ pode ser associada a uma dada classe $i, i = 1, 2, \ldots, c$ simplesmente identificando qual raiz (protótipo) oferece o caminho ótimo como se essa amostra fosse parte da floresta. Considerando os $k$-vizinhos mais próximos de $\mathbf{t}$, é então empregada a Equação 3.3 para calcular $\rho(\mathbf{t})$ para maior avaliação da Equação 3.6:

$$V(\mathbf{t}) = \max_{\forall(\mathbf{s},\mathbf{t})\in A_k^*} \{\min\{V(\mathbf{s}), \rho(\mathbf{t})\}\}. \tag{3.6}$$

O rótulo de $\mathbf{t}$ será o mesmo de seu precursor $P^*(\mathbf{t})$.

# Capítulo 4

# Metodologia

Neste Capítulo, será apresentada a metodologia utilizada para o desenvolvimento dos experimentos relacionados à essa dissertação.

O trabalho estuda a ameaça de spam em quatro diferentes ambientes, a saber: E-mail, SMS, comentários em blogs e comentários em vídeos da plataforma Youtube. Para analisar o problema de spam nesses ambientes, foram utilizadas quatro bases públicas e reais que permitiram a realização dos experimentos e o cumprimento do objetivo deste trabalho. Com relação a spam em e-mail, foram utilizadas as bem conhecidas bases do conjunto Enron [27]. O conjunto foi formado a partir da caixa de entrada de empregados da empresa Enron, criando as bases Enron1, Enron2, Enron3, Enron4, Enron5, as quais foram complementadas por outras bases, como *SpamAssassin* [1], *Bekkerman*[28], *Honeypot project* [2], a coleção de spam de *Bruce Guenter* [3] e spams coletados por um dos autores do referido artigo. A Tabela 4.1 resume o conjunto Enron, apresentando a quantidade de amostras de spams e hams em cada uma das bases.

Tabela 4.1: Bases do conjunto Enron.

| *Base* | *Ham* | *Spam* | *Total* |
|---|---|---|---|
| Enron1 | 3.672 | 1.500 | 5.172 |
| Enron2 | 1.496 | 4.361 | 5.857 |
| Enron3 | 4.012 | 1.500 | 5.512 |
| Enron4 | 1.500 | 4.500 | 6.000 |
| Enron5 | 1.500 | 3.675 | 5.175 |

Acerca de SMS, foi utilizada a base *SMS Spam Collection*, pública e real, disponibilizada por Almeida et al. [29]. A base é composta por 4.827 hams e 747 mensagens de spam, compondo um total de 5.574 amostras. Com relação à comentários em posts de blogs, é utilizada a base *BlogSpam Collection* [21], que é composta por 1.024 comentários rotulados como spam ou ham extraídos de 50 blogs diferentes. As mensagens, divididas em duas classes, são da seguinte proporção: 68% de spam, ou seja, 697 amostras, e 32% de ham, ou seja, 327 amostras. Por fim, sobre os comentários em vídeos do Youtube, foi utilizado o conjunto TubeSpam, disponibilizado por Alberto et al. [24]. O conjunto é composto por comentários extraídos dos cinco vídeos mais vistos de todos os tempos do Youtube e pode ser sumarizado pela Tabela 4.2.

Considerando que algumas bases, como *Blogspam*, *SMS Spam Collection* e o conjunto Enron são desbalanceadas, fica inuitivo a tentativa de criar subconjuntos de treino e teste que sejam balanceados, objetivando a melhora de desempenho dos classificadores. Entretanto,alguns testes realizados mostraram que o balanceamento desses conjuntos não impactou no resultado

[1] Veja: http://spamassassin.apache.org/ <acesso em 2/03/2016>

[2] Consulte: http://www.projecthoneypot.org/ <acesso em 2/03/2016>

[3] Veja: http://untroubled.org/spam/ <acesso em 2/03/2016>

Tabela 4.2: Bases do conjunto TubeSpam.

| *Base* | *Ham* | *Spam* | *Total* |
|---|---|---|---|
| Psy | 175 | 175 | 350 |
| KatyPerry | 175 | 175 | 350 |
| LMFAO | 236 | 202 | 438 |
| Eminem | 245 | 203 | 438 |
| Shakira | 174 | 196 | 370 |

obtidos pelos classificadores baseados em Floresta de caminhos ótimos, motivo pelo qual o balanceamento foi descartato.

No presente trabalho, serão comparados dois métodos de extração de características das amostras, a saber: token e 3gram. O processo de *tokenização* consiste em quebrar o fluxo do texto utilizando delimitadores pré-definidos para, então, extrair todos os tokens (termos) de todas as mensagens. Os tokens formam o que convencionalmente chama-se de dicionário, aqui denotado por $\mathcal{D}$. Os delimitadores utilizados foram espaços em branco, *tabs*, *returns*, pontos, vírgulas, dois pontos e traços. Nenhuma outra técnica de pré-processamento foi utilizada, tendo em vista que tais técnicas tendem a prejudicar a acurácia na filtragem de spam [30, 31].

Tais tokens extraídos compõem o espaço de características e, a partir desse espaço, é criada uma matriz esparsa que informa, para cada mensagem, a presença ou ausência de um token da mensagem no espaço de características, isto é, se um token de uma mensagem está presente no espaço de características é atribuído o valor 1 a ele e, caso esteja ausente, o valor 0 é atribuído. Assim, as características pelas quais serão atribuídos os rótulos spam ou ham à cada mensagem é determinada pela presença ou ausência do token da mensagem no espaço de características.

Com relação à 3gram, foi utilizada uma metodologia baseada naquela empregada por Kanaris et al. [32]. *N-grams* são *strings* de tamanho $N$ extraídas de uma mensagem de texto e, aqui, são utilizadas *strings* de tamanho 3, que produzem resultados melhores se comparados a tokens, segundo Kanaris et al. [32]. Assim, não foram utilizados $N$ maiores que 3 devido ao aumento significativo da dimensionalidade do problema. Toda mensagem será quebrada em *strings* de tamanho de 3. Considere, como exemplo, a seguinte mensagem: "Essa frase.". Utilizando 3gram, as *strings* ficarão dessa maneira, levando em consideração que espaços em branco são substituídos pelo caracter "_": |Ess|, |ssa|, |sa_|, |a_f|, |_fr|, |fra|, |ras|, |ase|. O mesmo processo de criação de um dicionário usado na *tokenização* foi aqui empregado.

Dada a maneira como os métodos trabalham, a Tabela 4.3 apresenta o número de características extraída de cada base.

Pode-se perceber que a utilização de 3gram aumenta o tamanho do espaço de características, o que é esperado para bases que são heterogêneas, isto é, são compostas por diferentes tipos de mensagens. Entretanto, o oposto ocorre com as bases do conjunto Enron, que têm o tamanho do espaço de características dimunuído quando é utilizado 3gram. Isso se explica pelo fato delas serem mais homogêneas, ou seja, como as bases foram formadas a partir da caixa de e-mail dos funcionários da empresa Enron, somente as mensagens de spam que passavam pelo filtro acabaram compondo a base, visto que as demais eram bloqueadas.

Todas as bases foram divididas em um conjunto de treinamento e um conjunto de teste para ambos os métodos utilizados, isto é, token e 3gram, mas a proporção não foi a mesma. As bases do conjunto *TubeSpam* e *BlogSpam* foram divididas na proporção 50% - 50%, e, partir dessa divisão, foram executadas 10 rodadas de experimentos do tipo validação cruzada. As bases do conjunto Enron e a base *SMS Spam Collection* foram divididas na proporção 70% - 30%, porém utilizando conjuntos fixos de amostras. Essa diferença se dá por dois motivos: (1) a dimensionalidade do problema analisado, já que e-mail e SMS possuem uma quantidade de amostras e número de características muito superior ao que apresenta as outras bases e

Tabela 4.3: Número de características de cada base por método.

| *Base* | $Quantidade - Token$ | $Quantidade - 3gram$ |
|---|---|---|
| Psy | 1.883 | 7.297 |
| KatyPerry | 2.204 | 8.955 |
| LMFAO | 1.465 | 5.130 |
| Eminem | 2.332 | 7.103 |
| Shakira | 1.922 | 6.030 |
| BlogSpam | 11.844 | 16.743 |
| SMS Spam Collection | 12.622 | 20.021 |
| Enron1 | 50.561 | 23.811 |
| Enron2 | 40.251 | 17.885 |
| Enron3 | 53.874 | 22.651 |
| Enron4 | 69.533 | 26.574 |
| Enron5 | 42.287 | 18.964 |

(2) a maioria dos textos encontrados na literatura sugere uma divisão parecida com a que foi praticada aqui, já que os melhores resultados foram atingidos dessa maneira.

Além dos classificadores baseados em floresta de caminhos ótimos, foram avaliados os seguintes classificadores: SVM, ANN com perceptrons multicamada (ANN-MLP) e o classificador knn. Para implementar o SVM, foi utilizada a LibSVM [33] com o kernel de base radial com parâmetros otimizados por uma validação cruzada com 5-folds, e com $C \in \{2^{-5}, 2^{-3} \ldots, 2^{13}, 2^{15}\}$ e $\gamma \in \{2^3, 2^1, \ldots, 2^{-13}, 2^{-15}\}$. Com relação à ANN-MLP, foi utilizada a biblioteca FANN library [34] com uma arquitetura neural $|\mathcal{D}|$:8:8:2, em que $|\mathcal{D}|$ representa o tamanho do dicionário extraído de cada base, duas unidades escondidas com oito neurônios cada uma e dois neurônios de saída, cada um correspondendo à uma classe, spam ou ham. Com relação ao OPF, foi utilizada a LibOPF [35], que permite ao valor de $k_{max}$, na versão do OPF com grafo KNN, ser definido pelo usuário e que, no caso, foi definido como sendo 10. A partir de então, o valor de $k$ que maximizar a acurácia no conjunto de treinamento será utilizado. A versão com grafo completo não necessita de parâmetros para serem ajustados. Por fim, com respeito à $k$nn, foi utilizada uma própria implementação com $k \in \{1, 3, 5, \ldots, 100\}$, ou seja, foi obtido o valor de $k$ que maximiza a acurácia no conjunto de treinamento no intervalo apresentado.

Por fim, para avaliar os classificadores, foram utilizados as seguintes medidas de desempenho:

- *Spam Caught* (SC%): quantidade de spam corretamente classificada como spam;
- *Blocked Ham* (BH%): quantidade de ham incorretamente classificada como spam;
- *Mattheus Correlation Coefficient* (MCC%): coeficiente de correlação entre a classificação binária predita e observada, assumindo valores entre -1 e 1, sendo -1 um total desacordo entre as predições; 0 representando um valor não melhor que qualquer predição aleatória; e 1 indicando a predição perfeita;
- Acurácia (ACC%): quantidade de acertos dividido pelo número total de amostras.

Acerca do MCC, fazem-se necessários alguns comentários a mais devido à sua complexidade. Apresentado em 1975 por Matthews, B.W. [36], essa medida faz uso da Matriz de Confusão que, baseada nas taxas de *False Positive* (FP), *False Negative* (FN), *True Positive* (TP) e *True Negative* (TN), consegue calcular o MCC, apresentado na Equação 4.1:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN))}}. \tag{4.1}$$

Além dessas medidas de desempenho, o tempo de treinamento será usado para comparar a eficiência dos classificadores. Essa medida é importante, pois mostra quanto tempo um classificador leva para treinar novamente após uma nova mensagem de spam ou ham, até então desconhecida pelo classificador, ter suas características extraídas e adicionadas ao dicionário, a fim de que esta mesma mensagem não chegue até o usuário novamente, caso seja spam, e não fique presa na caixa de spam ou até mesmo não chegue ao destino, caso seja ham.

# Capítulo 5

# Resultados

Este Capítulo apresenta os resultados obtidos a partir da metodologia utilizada no Capítulo 4.

## 5.1 Apresentação dos Resultados

Os resultados aqui apresentados estão organizados por conjunto de dados e, dessa maneira, serão mostrados o que foi obtido em cada método, ou seja, token e 3gram, verificando, assim, aquele que é mais eficaz na detecção de conteúdo malicioso, e isto será seguido por alguns comentários desses resultados. Ao final da apresentação de todos os resultados, será feita uma análise comparativa geral de tudo o que foi apresentado. Os resultados em negrito representam as técnicas mais eficazes e eficientes, isto é, melhores resultados nas medidas de desempenho e tempo de treinamento mais rápido, respectivamente.

### 5.1.1 Blog Spam

O primeiro conjunto de dados verificado é composto pela base *BlogSpam Collection* e os resultados alcançados pelas medidas de desempenho são mostrados nas Tabelas 5.1 e 5.3, bem como os respectivos tempos de treinamento para cada classificador são apresentados nas Tabelas 5.2 e 5.4.

Tabela 5.1: Resultados - *BlogSpam* (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 51.88% | **3.37%** | 66.39% | 0.4769 |
| $Knn$ | **98.96%** | 92.89% | 69.18% | 0.1664 |
| OPF | 98.06% | 91.75% | 68.95% | 0.1485 |
| OPF$-knn$ | 98.90% | 92.47% | 69.28% | 0.1680 |
| SVM | 93.06% | 31.33% | **85.16**% | **0.6518** |

Tabela 5.2: Tempo de Treinamento - *BlogSpam* (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 10020.50 |
| $Knn$ | 186.08 |
| OPF | **3.26** |
| OPF$-knn$ | 25.98 |
| SVM | 7264.48 |

Tabela 5.3: Resultados - *BlogSpam* (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 88.58% | 80.66% | 66.13% | 0.4043 |
| $Knn$ | **99.68%** | 87.77% | 71.33% | 0.2789 |
| OPF | 99.51% | 88.07% | 71.11% | 0.2675 |
| OPF$-knn$ | 99.65% | 89.46% | 70.76% | 0.2538 |
| SVM | 92.75% | **29.22%** | **85.63%** | **0.6641** |

Tabela 5.4: Tempo de Treinamento - *BlogSpam* (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 13126.20 |
| $Knn$ | 267.17 |
| OPF | **4.51** |
| OPF$-knn$ | 41.27 |
| SVM | 10754.2 |

De maneira geral, todos os classificadores se apresentaram abaixo do esperado, sendo o SVM o melhor deles, atingindo uma acurácia de 85.63%. O principal fator responsável por esse mal resultado é a quantidade de mensagens legítimas classificadas como spam, representado pela taxa BH. Num ambiente real, ficaria impossível implementar alguma solução que utilizasse quaisquer um desses classificadores, o que mostra a real necessidade de melhoramentos em todas as ferramentas de classificação. Apesar de a maioria deles acertarem mais de 90% dos spams, à exceção do ANN-MLP usando token, o MCC ficou duramente penalizado por causa do BH, o que pode ser inferido pela disparidade entre o que deveria ser classificado corretamente e o que realmente foi. Entretanto, cabe destacar o valor de BH obtido pelo classificador ANN-MLP usando token, um valor aceitável para uma aplicação real. O OPF, tanto usando grafo completo quanto grafo $knn$, obteve resultados similares aos demais, porém ele se destaca no tempo de treinamento, ficando muito abaixo dos outros e conseguindo atingir resultados semelhantes. Observando todos os quesitos analisados, o SVM se saiu melhor, mas com o custo do tempo de treinamento muito superior ao do OPF e também ao dos demais.

### 5.1.2 Social Spam

Agora será analisado o conjunto de dados *TubeSpam*, composto por cinco bases, a saber: Eminem, KatyPerry, LMFAO, Psy e Shakira, tendo seus resultados apresentados nessa ordem. As Tabelas 5.5 e 5.7 apresentam os resultados obtidos usando token e 3gram, respectivamente. Já as Tabelas 5.6 e 5.8 mostram o tempo de treinamento dos classificadores usando token e 3gram, respectivamente.

Tabela 5.5: Resultados - Eminem (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | **97.56%** | 36.96% | 81.91% | 0.6623 |
| $Knn$ | 67.89% | 1.96% | 81.56% | 0.6779 |
| OPF | 67.32% | **1.57%** | 81.42% | 0.6780 |
| OPF$-knn$ | 65.54% | 1.76% | 80.31% | 0.6602 |
| SVM | 89.76% | 4.02% | **92.58%** | **0.8540** |

Tabela 5.6: Tempo de Treinamento - Eminem (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 0.58 |
| $Knn$ | 2.59 |
| OPF | **0.11** |
| OPF$-knn$ | 0.93 |
| SVM | 293.96 |

Tabela 5.7: Resultados - Eminem (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 78.13% | 50.59% | 65.11% | 0.4257 |
| $Knn$ | 74.31% | 0.49% | 85.73% | 0.7481 |
| OPF | 74.31% | 0.49% | 85.73% | 0.7481 |
| OPF$-knn$ | 68.55% | **0.43**% | 83.73% | 0.7092 |
| SVM | **94.96**% | 1.27% | **96.67**% | **0.9343** |

Tabela 5.8: Tempo de Treinamento - Eminem (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 1025.15 |
| $Knn$ | 8.39 |
| OPF | **0.37** |
| OPF$-knn$ | 3.48 |
| SVM | 909.35 |

Os resultados apresentados mostram superioridade do classificador SVM. Entretanto, seu tempo de treinamento continua sendo um entrave para uma aplicação real. O pior classificador foi o ANN-MLP que, apesar de ter apresentado a melhor taxa de SC usando token, classificou muitos hams erroneamente, penalizando sua acurácia e seu MCC. O KNN, OPF-$knn$ e o OPF apresentaram resultados bem semelhantes, sendo bons candidatos à aplicações reais, principalmente pelo fato do OPF-$knn$ ter atingido a menor taxa de BH. A taxa de BH ficou bem baixa, principalmente quanto utilizado 3gram que, apesar de ser dimensionalmente maior que token, consegue representar melhor as características comuns à mensagens legítimas. O tempo de treinamento do OPF novamente foi destaque, ficando bem abaixo dos demais.

As Tabelas 5.9 e 5.11 mostram os resultados obtidos pela base KatyPerry e as Tabelas 5.10 e 5.12 apresentam o tempo de treinamento de cada classificador para, respectivamente, token e 3gram.

Tabela 5.9: Resultados - KatyPerry (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | **93.75**% | 34.77% | 79.49% | 0.6300 |
| $Knn$ | 56.48% | 6.59% | 74.94% | 0.5420 |
| OPF | 61.93% | 10.11% | 75.91% | 0.5513 |
| OPF$-knn$ | 56.48% | 6.48% | 75.00% | 0.5489 |
| SVM | 84.66% | **4.09**% | **90.28**% | **0.8119** |

O classificador SVM se sobressaiu aos demais, atingindo os melhores resultados. Apesar de, quando usando token, ANN ter conseguido melhor taxa de SC, os demais resultados o deixam

Tabela 5.10: Tempo de Treinamento - KatyPerry (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 0.35 |
| $Knn$ | 1.16 |
| OPF | **0.06** |
| OPF$-knn$ | 0.54 |
| SVM | 188.71 |

Tabela 5.11: Resultados - KatyPerry (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 68.75% | 30.57% | 69.09% | 0.4893 |
| $Knn$ | 41.82% | **1.36**% | 70.23% | 0.4916 |
| OPF | 40.91% | **1.36**% | 69.77% | 0.4843 |
| OPF$-knn$ | 34.20% | **1.36**% | 66.42% | 0.4260 |
| SVM | **87.61**% | 1.59% | **93.01**% | **0.8658** |

Tabela 5.12: Tempo de Treinamento - KatyPerry (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 1121.66 |
| $Knn$ | 4.97 |
| OPF | **0.27** |
| OPF$-knn$ | 2.33 |
| SVM | 766.39 |

como o pior classificador. Para token, o OPF foi ligeiramente superior a OPF-$knn$ e KNN. Já usando 3gram, as melhores taxas de BH ficaram com KNN e ambas versões do OPF, apesar de que o SVM atingiu um nível bastante similar. Em relação ao tempo, o OPF foi bem mais eficiente que os demais.

Os resultados obtidos pela base LMFAO são apresentados nas Tabelas 5.13 e 5.15 assim como os tempos de treinamento dos classificadores são apresentados nas Tabelas 5.14 e 5.16.

Tabela 5.13: Resultados - LMFAO (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | **89.07**% | 21.98% | 83.97% | 0.6889 |
| $Knn$ | 72.88% | **2.28**% | 84.34% | 0.7178 |
| OPF | 77.29% | 18.12% | 79.41% | 0.6134 |
| OPF$-knn$ | 69.15% | 5.15% | 81.00% | 0.6604 |
| SVM | 84.83% | 4.26% | **89.86**% | **0.8056** |

O classificador SVM foi o melhor para a base LMFAO, principalmente quando considerado o método 3gram, que também, de modo geral, melhorou todos os resultados. ANN continua, como nas demais bases já mostradas, errando muito ao classificar mensagens legítimas como spam, o que não ocorre com o KNN, OPF-$knn$ e OPF, que são os que mais acertam nesse quesito, apesar de, usando token, o OPF ter obtido resultado ruim. O tempo de treinamento continua a favor do OPF, bem abaixo dos demais.

A base Psy tem seus resultados apresentados nas Tabelas 5.17 e 5.19 e o tempo de treinamento de cada classificador, para cada método, apresentado nas Tabelas 5.18 e 5.20.

Tabela 5.14: Tempo de Treinamento - LMFAO (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 0.52 |
| $Knn$ | 1.62 |
| OPF | **0.07** |
| OPF$-knn$ | 0.56 |
| SVM | 170.32 |

Tabela 5.15: Resultados - LMFAO (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 77.54% | 34.75% | 71.87% | 0.5086 |
| $Knn$ | 83.73% | 1.09% | 90.73% | 0.8278 |
| OPF | 83.90% | 1.09% | 90.82% | 0.8293 |
| OPF$-knn$ | 82.29% | **0.99**% | 90.00% | 0.8156 |
| SVM | **92.54**% | 2.08% | **95.02**% | **0.9030** |

Tabela 5.16: Tempo de Treinamento - LMFAO (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 282.93 |
| $Knn$ | 5.85 |
| OPF | **0.25** |
| OPF$-knn$ | 2.09 |
| SVM | 582.06 |

Tabela 5.17: Resultados - Psy (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | **92.16**% | 35.40% | 78.46% | 0.6149 |
| $Knn$ | 72.84% | 5.86% | 83.43% | 0.6896 |
| OPF | 59.20% | **2.64**% | 78.17% | 0.6135 |
| OPF$-knn$ | 65.57% | 14.14% | 75.66% | 0.5624 |
| SVM | 85.23% | 3.22% | **90.97**% | **0.8265** |

Tabela 5.18: Tempo de Treinamento - Psy (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 0.38 |
| $Knn$ | 1.09 |
| OPF | **0.06** |
| OPF$-knn$ | 0.46 |
| SVM | 161.48 |

Tabela 5.19: Resultados - Psy (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 70.45% | 31.61% | 69.43% | 0.5088 |
| $Knn$ | 57.27% | 2.07% | 77.49% | 0.6050 |
| OPF | 57.27% | 2.07% | 77.49% | 0.6050 |
| OPF$-knn$ | 52.27% | **1.26**% | 75.26% | 0.5724 |
| SVM | **90.34**% | 2.07% | **94.11**% | **0.8853** |

Tabela 5.20: Tempo de Treinamento - Psy (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 510,72 |
| $Knn$ | 4.37 |
| OPF | **0.23** |
| OPF$-knn$ | 1.96 |
| SVM | 631.87 |

Quando é considerado o método 3gram, pode-se dizer que o SVM foi melhor em três das quatros medidas, e no método token, em duas. O OPF obteve bons resultados na classificação correta de mensagens legítimas, dada sua taxa de BH. Entretanto, não foi melhor que o OPF-$knn$, que obteve a menor taxa usando 3gram. O ANN apresenta boa taxa de SC, porém seu BH é alto, o que reflete no ACC e MCC, fazendo com que, de modo geral, seja um mal classificador. O tempo de treinamento é favorável ao OPF mais uma vez, e o resultado de suas medidas são medianos.

Por fim, as Tabelas 5.21 e 5.23 mostram os resultados para as medidas de desempenho da base Shakira e as Tabelas 5.22 e 5.24 mostram o tempo de treinamento tanto para token e para 3gram, respectivamente.

Tabela 5.21: Resultados - Shakira (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 87.24% | 23.57% | 81.51% | 0.6513 |
| $Knn$ | 48.05% | **0.41**% | 75.35% | 0.5665 |
| OPF | 47.13% | 0.71% | 74.76% | 0.5542 |
| OPF$-knn$ | 40.80% | 0.61% | 71.84% | 0.5037 |
| SVM | **89.43**% | 5.71% | **92.00**% | **0.8422** |

Tabela 5.22: Tempo de Treinamento - Shakira (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 51.30 |
| $Knn$ | 1.27 |
| OPF | **0.06** |
| OPF$-knn$ | 0.54 |
| SVM | 170.29 |

Tabela 5.23: Resultados - Shakira (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 62.76% | 27.96% | 67.68% | 0.5509 |
| $Knn$ | 55.98% | **0.10**% | 79.24% | 0.6329 |
| OPF | 55.52% | **0.10**% | 79.03% | 0.6294 |
| OPF$-knn$ | 48.74% | **0.10**% | 75.84% | 0.5770 |
| SVM | **89.77**% | 2.04% | **94.11**% | **0.8841** |

Tanto para token quanto para 3gram, SVM atingiu os melhores resultados considerando SC, ACC e MCC e o mesmo pode ser observado para o KNN considerando BH, porém, ressaltando

Tabela 5.24: Tempo de Treinamento - Shakira (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 661.06 |
| $Knn$ | 4.24 |
| OPF | **0.21** |
| OPF$-knn$ | 1.83 |
| SVM | 535.71 |

que o OPF e OPF-$knn$, utilizando 3gram obtiveram os mesmos resultados que o KNN. De modo geral, quando os classsificadores utilizam 3gram, os resultados são melhores, apesar do tempo de treinamento aumentar. Este, por sinal, continua sendo o diferencial do OPF, ficando muito abaixo dos demais.

### 5.1.3 SMS Spam

Os resultados obtidos pelo conjunto de dados *SMS Spam Collection* são mostrados nas Tabelas 5.25 e 5.27, enquanto os tempos de treinamento de cada classificador são apresentados nas Tabelas 5.26 e 5.28, respectivamente para token e 3gram.

Tabela 5.25: Resultados - *SMS Spam Collection* (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | **99.56**% | 24.15% | 79.03% | 0.5418 |
| $Knn$ | 58.67% | **0.00**% | 94.44% | 0.7425 |
| OPF | 57.78% | **0.00**% | 94.33% | 0.7363 |
| OPF$-knn$ | 50.67% | **0.00**% | 93.37% | 0.6860 |
| SVM | 89.33% | 0.14% | **98.45**% | **0.9320** |

Tabela 5.26: Tempo de Treinamento - *SMS Spam Collection* (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 117410.43 |
| $Knn$ | 82989.23 |
| OPF | **413.50** |
| OPF$-knn$ | 1584.60 |
| SVM | 414688.00 |

Tabela 5.27: Resultados - *SMS Spam Collection* (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 0 | 0 | 0 | 0 |
| $Knn$ | 63.11% | **0.00**% | 95.05% | 0.7726 |
| OPF | 63.11% | **0.00**% | 95.05% | 0.7726 |
| OPF$-knn$ | 52.44% | **0.00**% | 93.61% | 0.6988 |
| SVM | **89.78**% | **0.00**% | **98.63**% | **0.9401** |

Usando o método 3gram, os resultados de todos os classificadores, à exceção do classificador ANN, que não conseguiu classificar os dados, melhoraram. Os classificadores conseguiram

Tabela 5.28: Tempo de Treinamento - *SMS Spam Collection* (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 124501.15 |
| $Knn$ | 145906.42 |
| OPF | **328.24** |
| OPF$-knn$ | 2622.67 |
| SVM | 447126.75 |

detectar um número alto de spams e não bloquearam nenhuma mensagem legítima com esse método. Mais uma vez, SVM se sobressaiu aos demais, entretanto, tanto o OPF quanto o OPF$-knn$ obtiveram bons resultados, e têm o tempo de treinamento como seu principal diferencial, ficando muito abaixo dos demais.

### 5.1.4 E-mail Spam

Por fim, serão apresentados os resultados obtidos pelos classificadores analisando todas as bases do conjunto de dados Enron. Os resultados obtidos pela base Enron1 são apresentados na Tabela 5.29 ,em relação ao método utilizando token, e na Tabela 5.31 são mostrados os resultados usando ao método usando 3gram. Os respectivos tempos de treinamento de cada classificador, de acordo com cada método, são apresentados nas Tabelas 5.30 e 5.32.

Tabela 5.29: Resultados - Enron1 (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | **95.78**% | **5.99**% | **94.52**% | **0.8732** |
| $Knn$ | 93.11% | 22.41% | 82.09% | 0.6481 |
| OPF | 90.89% | 20.42% | 82.86% | 0.6505 |
| OPF$-knn$ | 93.56% | 25.77% | 79.83% | 0.6177 |
| SVM | **95.78**% | **5.99**% | **94.52**% | **0.8732** |

Tabela 5.30: Tempo de Treinamento - Enron1 (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 1625.18 |
| $Knn$ | 504671.21 |
| OPF | **1412.41** |
| OPF$-knn$ | 10683.25 |
| SVM | 1498138.25 |

Tabela 5.31: Resultados - Enron1 (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 70.89% | **0.9**% | 90.91% | 0.7770 |
| $Knn$ | 86.89% | 15.88% | 84.92% | 0.6694 |
| OPF | 86.44% | 16.69% | 84.21% | 0.6558 |
| OPF$-knn$ | 88.22% | 21.23% | 81.51% | 0.6188 |
| SVM | **96.89**% | 1.82% | **97.81**% | **0.9470** |

Tabela 5.32: Tempo de Treinamento - Enron1 (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | **524.31** |
| $Knn$ | 2819.02 |
| OPF | 557.17 |
| OPF$-knn$ | 2819.02 |
| SVM | 1156201.25 |

De modo geral, quando utilizado o método 3gram, os resultados foram melhores, com o classificador SVM se sobressaindo. Em relação ao método usando token, ANN foi o melhor em todos os medidores. É notável o fato de, utilizando 3gram, o tempo de treinamento para ANN ter sido inferior ao do OPF. Todos os classificadores souberam bem classificar spams, porém erraram BH, à exceção de ANN e SVM.

Os classificadores obtiveram os seguintes resultados para a base Enron2, apresentados nas Tabelas 5.33 e 5.35, tendo seus tempos de treinamento apresentados nas Tabelas 5.34 e 5.36, respectivamente para token e 3gram.

Tabela 5.33: Resultados - Enron2 (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | **99.55**% | 6.26% | 95.22% | 0.8870 |
| $Knn$ | 79.06% | 9.70% | 87.43% | 0.6781 |
| OPF | 76.84% | 10.69% | 86.12% | 0.6453 |
| OPF$-knn$ | 73.94% | 10.31% | 85.66% | 0.6282 |
| SVM | **99.55**% | **3.24**% | **97.64**% | **0.9103** |

Tabela 5.34: Tempo de Treinamento - Enron2 (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | **319.63** |
| $Knn$ | 584239.81 |
| OPF | 1457.03 |
| OPF$-knn$ | 11240.67 |
| SVM | 1535494.87 |

Tabela 5.35: Resultados - Enron2 (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 98.22% | 6.72% | 94.54% | 0.8699 |
| $Knn$ | 72.61% | 1.83% | 91.64% | 0.7729 |
| OPF | 74.39% | 1.37% | 92.44% | 0.7957 |
| OPF$-knn$ | 73.05% | 1.60% | 91.93% | 0.7811 |
| SVM | **98.44**% | **0.84**% | **98.97**% | **0.9731** |

Os resultados mostram que ANN e SVM foram os melhores classificadores para a base Enron2. A maior taxa de SC foi obtida por ambos os classificadores usando token, enquanto a menor taxa de BH foi obtida pelo SVM usando 3gram. Com este método, inclusive, OPF e OPF-$knn$ foram melhores, tendo suas taxas BH baixas e SC relativamente altas, o que conferiu a

Tabela 5.36: Tempo de Treinamento - Enron2 (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 571.39 |
| $Knn$ | 152053.69 |
| OPF | **337.57** |
| OPF$-knn$ | 2577.73 |
| SVM | 919512.75 |

ambos uma acurácia superior a 90%. Com relação ao tempo de treinamento, ANN surpreendeu e, usando token, foi o mais rápido e, mesmo com 3gram, obteve valor próximo ao OPF.

Os resultados obtidos pelos classificadores na base Enron3 são apresentados nas Tabelas 5.37 e 5.39, tanto para token quanto para 3gram, enquanto os respectivos tempos de treinamento são apresentados nas Tabelas 5.38 e 5.40.

Tabela 5.37: Resultados - Enron3 (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | **99.33**% | 26.99% | 80.17% | 0.6453 |
| $Knn$ | 58.44% | 2.57% | 68.82% | 0.6504 |
| OPF | 54.67% | 4.73% | 84.22% | 0.5744 |
| OPF$-knn$ | 60.44% | 1.91% | 87.85% | 0.6805 |
| SVM | 96.73% | **1.21**% | **97.33**% | **0.9331** |

Tabela 5.38: Tempo de Treinamento - Enron3 (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 461491.75 |
| $Knn$ | 644436.75 |
| OPF | **1639.23** |
| OPF$-knn$ | 12960.05 |
| SVM | 1892918.25 |

Tabela 5.39: Resultados - Enron3 (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | **98.67**% | 11.13% | 91.54% | 0.8170 |
| $Knn$ | 64.44% | **0.16**% | 90.21% | 0.7502 |
| OPF | 64.22% | 1.08% | 89.48% | 0.7274 |
| OPF$-knn$ | 60.44% | 0.42% | 88.94% | 0.7154 |
| SVM | 97.11% | 0.58% | **98.79**% | **0.9694** |

O classificador ANN foi o que melhor conseguiu detectar spam, entretanto bloqueou muitas mensagens legítimas, o que o torna uma opção não útil em uma aplicação real. O SVM foi, de modo geral, o classificador que se saiu melhor, a um custo muito alto dado seu elevado tempo de treinamento. O classificador KNN obteve baixas taxas de BH, uma delas a menor de todas, mas sua taxa de SC foi pequena. O mesmo pode ser dito do OPF, o que fez com que sua acurácia não fosse tão elevada. Pela primeira vez, o OPF-$knn$ superou o OPF, obtendo, usando token e em todos o sentidos, melhores resultados. O tempo de treinamento é o grande trunfo

Tabela 5.40: Tempo de Treinamento - Enron3 (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 402.97 |
| $Knn$ | 154960.11 |
| OPF | **375.80** |
| OPF$-knn$ | 2909.23 |
| SVM | 1323189.25 |

do OPF, sendo muito abaixo dos demais, apesar de que o ANN, usando 3gram, obteve tempo de treinamento próximo ao obtido pelo OPF.

As Tabelas 5.41 e 5.43 apresentam os resultados que os classificadores obtiveram usando os métodos token e 3gram, respectivamente, e as Tabelas 5.42 e 5.44 seus tempos de treinamento relacionados à base Enron4.

Tabela 5.41: Resultados - Enron4 (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 74.89% | **0.89**% | 80.94% | 0.6461 |
| $Knn$ | 98.59% | 21.33% | 93.61% | 0.8254 |
| OPF | 97.26% | 18.00% | 93.44% | 0.8212 |
| OPF$-knn$ | 98.59% | 22.22% | 93.44% | 0.8207 |
| SVM | **99.93**% | 6.00% | **98.44**% | **0.9584** |

Tabela 5.42: Tempo de Treinamento - Enron4 (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 474738.12 |
| $Knn$ | 380508.44 |
| OPF | **2548.98** |
| OPF$-knn$ | 19642.67 |
| SVM | 1427490.00 |

Tabela 5.43: Resultados - Enron4 (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 92.67% | **0.89**% | 94.28% | 0.8649 |
| $Knn$ | 98.74% | 18.67% | 94.39% | 0.8472 |
| OPF | 98.59% | 18.67% | 94.28% | 0.8440 |
| OPF$-knn$ | 98.59% | 20.00% | 93.94% | 0.8347 |
| SVM | **99.70**% | 4.89% | **98.56**% | **0.9613** |

Os resultados obtidos pelos classificadores na base Enron4 apresentam um mesmo padrão tanto para token quanto para 3gram, isto é, elevadas taxas de SC e BH, sendo este último exceção para ANN e SVM. Para esses dois classificadores, ainda, 3gram melhorou os resultados das medidas de desempenho, enquanto para KNN, OPF e OPF-$knn$ os melhores resultados foram obtidos com token. O tempo de treinamento do OPF foi melhor em ambos os métodos, porém, como ANN usando 3gram obteve ótima performance e bom tempo de treinamento, o resultado não melhora a posição do OPF como um classificador indicado para um aplicação real.

Tabela 5.44: Tempo de Treinamento - Enron4 (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 3698.59 |
| $Knn$ | 219788.05 |
| OPF | **596.21** |
| OPF$-knn$ | 3886.32 |
| SVM | 1612431.50 |

A Tabela 5.45 apresenta os resultados que foram obtidos a partir da base Enron5 usando token, e a Tabela 5.46 apresenta o tempo de treinamento para cada classificador. Já as Tabelas 5.47 e 5.48 mostram, respectivamente, os resultados e tempo de treinamento usando 3gram.

Tabela 5.45: Resultados - Enron5 (token).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 92.38% | **0.22**% | 94.53% | 0.8806 |
| $Knn$ | 92.38% | 15.11% | 90.21% | 0.7650 |
| OPF | 92.47% | 17.33% | 89.63% | 0.7490 |
| OPF$-knn$ | 96.01% | 20.44% | 91.24% | 0.7827 |
| SVM | **99.64**% | 0.24% | **99.03**% | **0.9765** |

Tabela 5.46: Tempo de Treinamento - Enron5 (token).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | **207.16** |
| $Knn$ | 148199.77 |
| OPF | 1264.11 |
| OPF$-knn$ | 9238.57 |
| SVM | 659075.37 |

Tabela 5.47: Resultados - Enron5 (3gram).

| Classificadores | $SC(\%)$ | $BH(\%)$ | $ACC(\%)$ | MCC |
|---|---|---|---|---|
| ANN-MLP | 81.32% | **0.66**% | 86.54% | 0.7412 |
| $Knn$ | 93.11% | 8.89% | 92.53% | 0.8240 |
| OPF | 95.47% | 16.22% | 92.10% | 0.8052 |
| OPF$-knn$ | 95.10% | 18.22% | 91.24% | 0.7840 |
| SVM | **99.73**% | 2.44% | **99.10**% | **0.9781** |

Tabela 5.48: Tempo de Treinamento - Enron5 (3gram).

| Classificadores | Tempo (segundos) |
|---|---|
| ANN-MLP | 4393.15 |
| $Knn$ | 96414.17 |
| OPF | **289.11** |
| OPF$-knn$ | 2098.02 |
| SVM | 631439.06 |

Os classsificadores obtiveram desempenho semelhante tanto usando token quanto 3gram, visto que, à exceção de BH, cujo melhor resultado foi obtido pelo classificador ANN, as demais medidas de desempenho obtiveram seu melhor resultado com o classsificador SVM. O método usando 3gram foi ligeiramente superior à token e, mais uma vez, o classificador ANN foi mais rápido que o OPF, apenas, no entanto, usando token. OPF-*knn* classificou melhor as mensagens de spam do que a versão do OPF usando grafo completo, entretanto, as demais medidas foram melhor com esta versão.

## 5.2 Comparação entre token e 3gram

Nesta seção, é feita uma análise comparativa entre os métodos de extração de características utilizados em cada base para obtenção dos resultados, visando determinar aquela que melhor se adequa ao problema de detecção de conteúdo malicioso. Para tanto, cada subseção contém a análise de um conjunto de dados, em que, a partir de uma figura, é comparada cada medida de avaliação obtida por cada um dos métodos. A ideia principal é verificar a variação que houve, por exemplo, da taxa ACC quando foi utilizado token e quando foi utilizado 3gram. Todas as medidas são analisadas, permitindo a inferência, assim, do melhor método para o problema analisado neste trabalho.

### 5.2.1 Blog Spam

A Figura 5.1 apresenta a comparação entre os dois métodos para a base *BlogSpam Collection*. É possível dizer que o classificador ANN é o mais sensível à mudança de método, o que fica claro quando são observadas as taxas SC e BH. Para BH, por exemplo, a variação é mais de 70%, altamente significativa. Em contrapartida, SVM é o classificador que menos apresenta sensibilidade à mudança de método, tendo seus resultados semelhantes tanto usando token quanto 3gram. KNN, OPF e OPF-*knn* apresenta pouca variação, sendo a mais significativa em relação ao MCC.

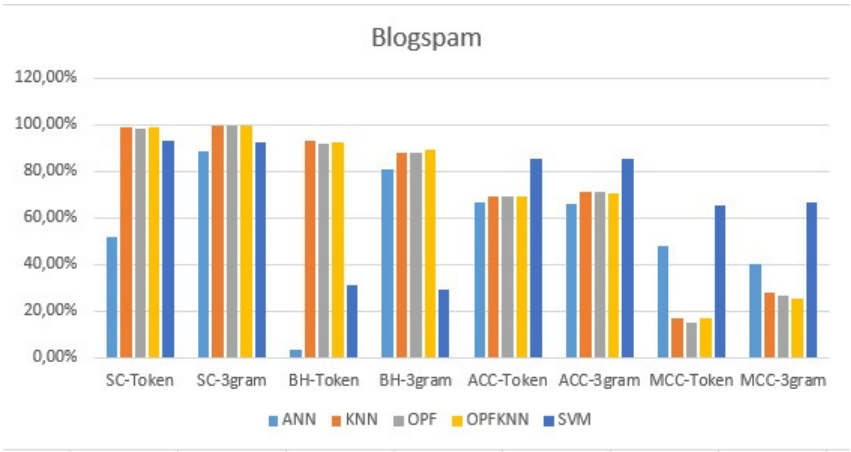


Figura 5.1: Comparação entre token e 3gram para a base BlogSpam.

### 5.2.2 Social Spam

As cinco bases que compõem o conjunto de dados *TubeSpam* não apresentam o mesmo comportamento, evidenciando a heterogeneidade apresentada nos comentários de cada vídeo a partir do qual é criada a base. A Figura 5.2 mostra a comparação entre token e 3gram para a base

Eminem. O classificador ANN, novamente, apresentou maior sensibilidade à mudança de método, entretanto, dessa vez, o OPF-*knn* foi o que se mostrou menos sensível. OPF, KNN e SVM apresentam sensibilidade semelhante para essa base, sendo que todos obtiveram uma melhora nos resultados quando foi utilizado 3gram.

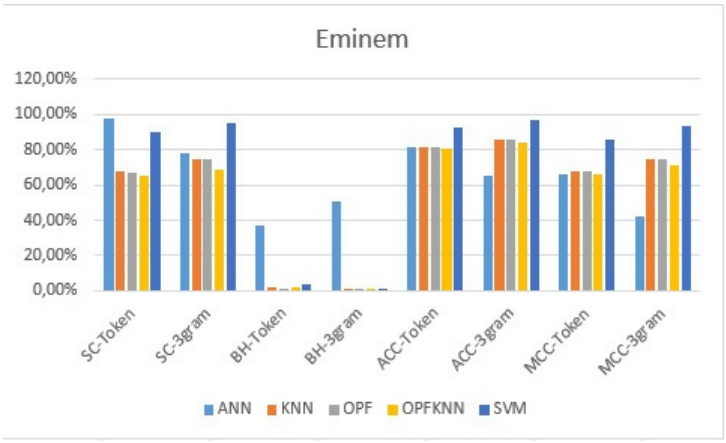


Figura 5.2: Comparação entre token e 3gram para a base Eminem.

A comparação para a base KatyPerry é apresentada na Figura 5.3. De modo geral, ANN e SVM representam os dois extremos da mudança de método, sendo ANN o mais sensível. Entretanto, é possível notar que o OPF e OPF-*knn* foram bastantes sensíveis à essa mudança também. KNN fica entre ANN e as duas versões do OPF. É importante notar que, à exceção do SVM, as taxas de SC caíram quando foi utilizado 3gram, mas, por outro lado, BH obteve melhores resultados quando este método foi usado.

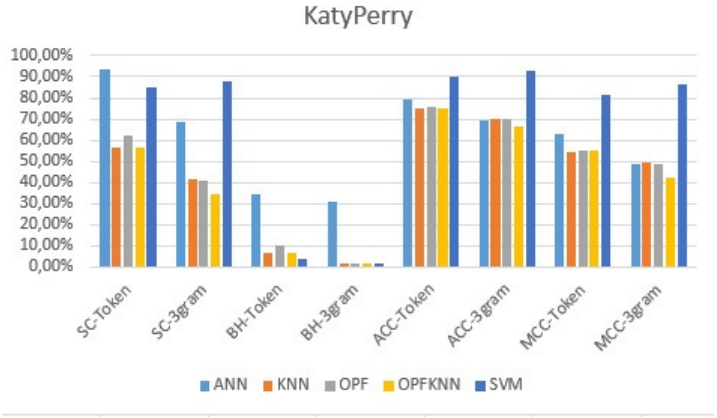


Figura 5.3: Comparação entre token e 3gram para a base KatyPerry.

A base LMFAO tem sua comparação apresentada na Figura 5.4. A sensibilidade de todos os classificadores nesta base foi maior devido à mudança de método. ANN continua sendo o mais sensível, com destaque para o BH, que piorou significativamente quando utilizado 3gram. Por outro lado, o OPF melhorou muito sua taxa de BH ao usar esse método. Os demais classificadores obtiveram uma melhora devido à essa mudança, porém, aqui, para todos eles, ela foi mais evidente.

A Figura 5.5 apresenta a comparação realizada para a base Psy. À exeção do OPF, os demais classificadores obtiveram significa mudança em relação ao SC, porém foi o OPF-*knn* que mais foi sensível em relação ao BH. ANN e KNN foram os mais sensíveis de modo geral, enquanto OPF e SVM vão na contramão dessa afirmação. Por fim, OPF-*knn* fica entreposto a essa separação.

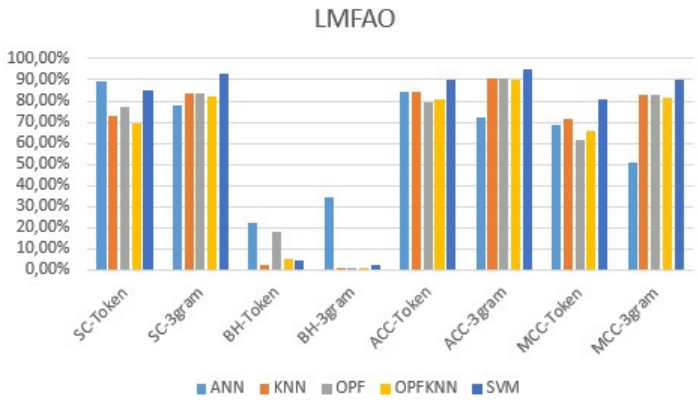


Figura 5.4: Comparação entre token e 3gram para a base LMFAO.

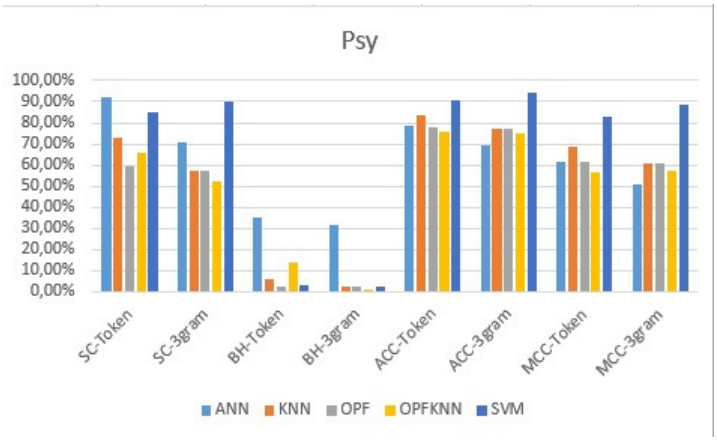


Figura 5.5: Comparação entre token e 3gram para a base Psy.

Finalizando, a Figura 5.6 mostra a comparação para a base Shakira. A base, de um modo geral, foi menos sensível à mudança de método, sendo que 3gram melhorou os resultados. Entretanto, pode-se verificar que SC, excluindo-se o obtido por SVM, foi o mais afetado pela mudança. SVM se manteve constante, com uma variação não superior a 3%.

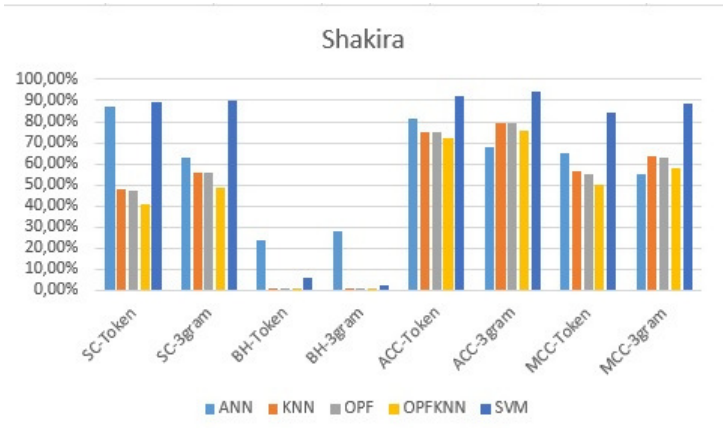


Figura 5.6: Comparação entre token e 3gram para a base Shakira.

### 5.2.3 SMS Spam

O resultado obtido pela comparação entre os métodos utilizados neste trabalho para o conjunto *SMS Spam Collection* é apresentado na Figura 5.7. Essa análise tem o inconveniente de que o ANN não conseguiu classificar os dados quando utilizado 3gram. Contudo, em relação aos demais classificadores, podemos notar um mesmo padrão: todas as medidas melhoraram quando foi utilizado 3gram. SVM foi o menos sensível, seguido pelo OPF-*knn*, enquanto OPF e KNN foram mais sensíveis, principalmente em relação à taxa de SC, o que refletiu diretamente em ACC e, principalmente, no MCC.

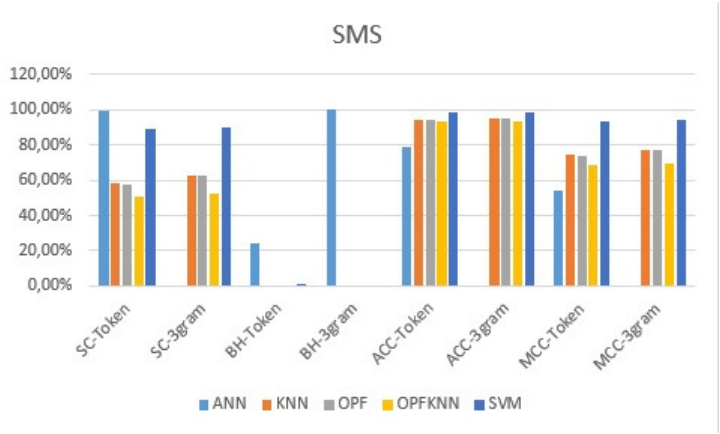


Figura 5.7: Comparação entre token e 3gram para a base SMS Spam Collection.

### 5.2.4 E-mail Spam

Nesta subseção serão analisadas as bases que compõem o conjunto de dados Enron. A Figura 5.8 apresenta a comparação para a base Enron1. Em todos os sentidos, SVM melhorou quando passou de token para 3gram. Os demais classificadores obtiveram uma queda na taxa SC, sendo o ANN o mais sensível. BH melhorou, principalmente para KNN, sendo que SVM e ANN obtiveram melhoras também e, por causa disso, o MCC de ambos aumentou.

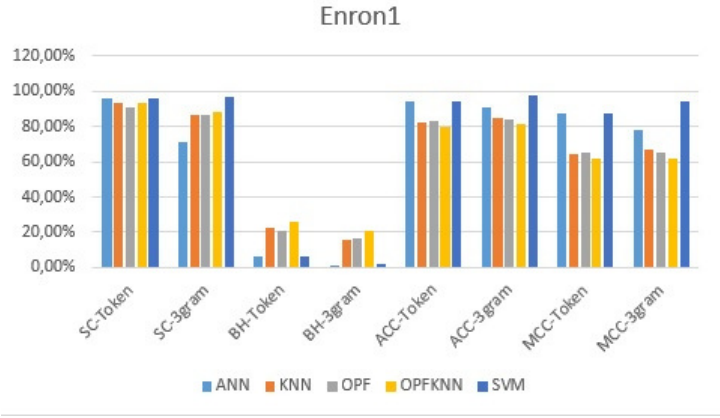


Figura 5.8: Comparação entre token e 3gram para a base Enron1.

Na Figura 5.9 está representada a sensibilidade de cada classificador com relação à mudança de método para a base Enron2. ANN, diferentemente do ocorrido anteriormente, foi o menos sensível à essa mudança. KNN, OPF, OPF-*knn* e SVM experimentaram uma significativa melhora da taxa de BH, o que, consequentemente, elevou ACC e MCC. Com relação à taxa SC, KNN foi o mais sensível, enquanto OPF-*knn* e SVM estão na contramão dessa afirmação.

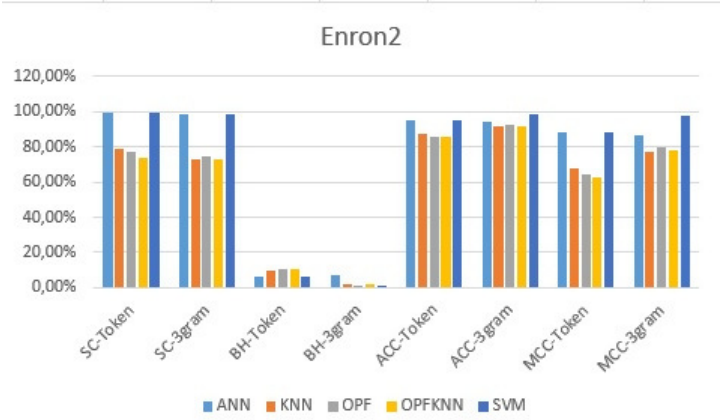


Figura 5.9: Comparação entre token e 3gram para a base Enron2.

A base Enron3 tem sua comparação apresentada na Figura 5.10. O classificador SVM foi o menos sensível à variação do método. O OPF teve uma mudança significativa, principalmente quando considerada a taxa SC, com aumento de quase 10%. Quando utilizado 3gram, a taxa de BH para todos os classificadores foi menor, o que elevou as taxas de ACC e MCC.

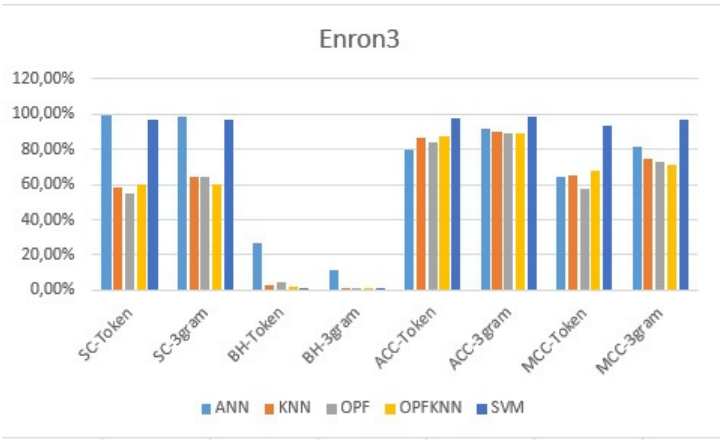


Figura 5.10: Comparação entre token e 3gram para a base Enron3.

A Figura 5.11 apresenta a comparação dos dois métodos obtida pela base Enron4. Com exceção do classificador ANN, os demais não apresentam grande sensibilidade à mudança de método, apesar de que com 3gram os resultados obtiveram uma pequena melhora. A grande diferença é obtida por ANN com a taxa SC, que foi elevada em quase 20%, o que refletiu diretamente nas taxas ACC e MCC, fazendo-as aumentar.

Por fim, Enron5 mostra seus resultados da comparação na Figura 5.12. Por ela, pode-se observar a menor variação registrada devido à mudança de método, obtida pelo SVM. Em menor escala, isso pode ser observado também tanto pelo OPF quanto pelo OPF-*knn*. ANN e KNN foram mais sensíveis, respectivamente à SC e BH, o que influenciou diretamente ACC e MCC.

## 5.3 Análise dos Resultados

Depois dos resultados apresentados, é possível dizer que o OPF, considerando ambas as versões aqui usadas, não superou os classificadores do estado-da-arte. Entretanto, conforme já dito, as pesquisas na área de detecção de conteúdo malicioso estão à procura de métodos tanto mais

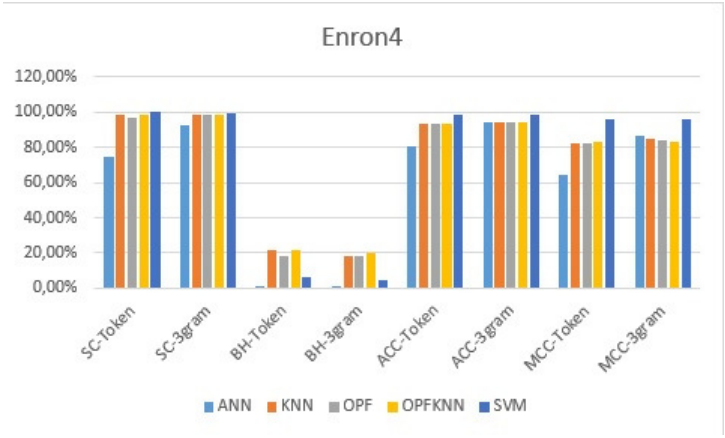


Figura 5.11: Comparação entre token e 3gram para a base Enron4.

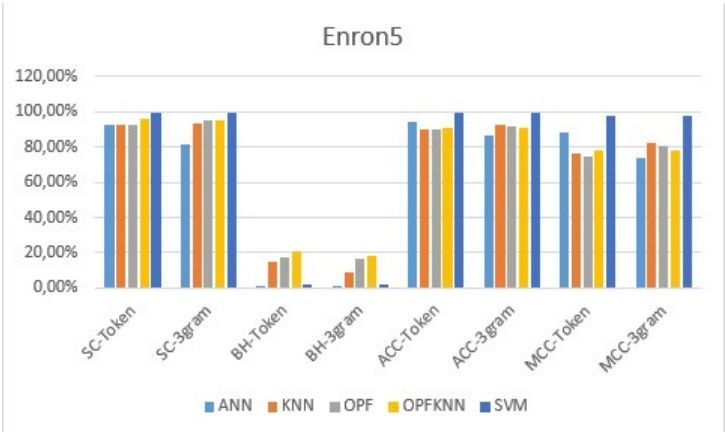


Figura 5.12: Comparação entre token e 3gram para a base Enron5.

eficazes quanto mais eficientes, ou seja, aqueles que conciliam boa classificação correta feita num curto período de tempo. Pelos resultados obtidos, pode-se, com certeza, afirmar que o OPF resolve a necessidade de eficiência, entretanto é necessário uma melhora em sua eficácia.

A base *BlogSpam* foi, de maneira geral, a mais desafiadora no projeto. Ao mesmo tempo em que ambas versões do OPF classificaram corretamente a maioria das mensagem de spam, erraram quase todas as legítimas, o que é pior do que deixar passar um spam. Isso afeta diretamente a acurácia, o que o torna inepto para uma aplicação real. O classificador SVM foi o que obteve o melhor resultado para esta base, mas mesmo assim, bloqueou mais de 29% das mensagens legítimas. O fato dessa base ter sido construída a partir de 50 diferentes blogs colabora para a heterogeneidade dos comentários, visto que cada blog trata de um assunto diferente. Isso leva os *spammers* à variarem o conteúdo dos spams postados em cada blog, dificultando a ação de classificadores, já que não é mantido um padrão.

Quando é considerado o conjunto *TubeSpam*, pode-se afirmar que é possível ver uma diferença quando utiliza-se token e quando utiliza-se 3gram, principalmente em relação ao BH. Ambas versões do OPF se saíram muito bem nesse quesito, bloqueando pouquíssimas mensagens legítimas. Entretanto, sua taxa de SC foi bem aquém do desejado, deixando passar muitos spams. Isso é prejudicial também. Porém, o tempo de treinamento para essas bases foi menos de um segundo para o OPF e não mais que 4 segundos para OPF-*knn*, um valor extremamente rápido, e muito acima do SVM que, apesar de ter obtido melhor desempenho de classificação, chega a levar até mais de 15 minutos para treinar. É importante notar que OPF se saiu bem na taxa de BH devido à semelhança entre os comentários legítimos encontrados nesses vídeos, já que os fãs que os assistem tendem a seguir um padrão de comentário, muitas vezes estabelecido

pelo próprio autor das canções. Já as mensagens de spams variam bastante, pois é um terreno muito fértil que pode ser explorado o de comentários em vídeos.

Já com relação ao conjunto *SMS Spam Collection*, é possível inferir que OPF possui boa avaliação, já que não bloqueou nenhuma mensagem legítima e bloqueou, quando é considerado 3gram, mais de 60% dos spams. É notório que essa taxa de detcção de spam precisa ser melhorada. Entretanto, quando comparado ao SVM, o OPF ganha muito em tempo de treinamento. Enquanto o SVM demora, no mínimo, 4,8 dias para treinar, o OPF demora apenas 328 segundos, uma diferença substancial. Apesar de aumentar o tamanho do espaço de características, o método baseado em 3gram se sai melhor para classificar esses dados, já que consegue generalizar mais o padrão das mensagens.

Por fim, as bases do conjunto Enron apresentam um desafio a ambas versões do OPF. Emails, de um modo geral, são muitos mais heterogêneos que qualquer outro tipo de mensagem, já que não têm limites de tamanho e aceitam muitos tipos de anexos. Quando são analisadas as bases Enron1 e Enron5, é possível verificar uma dificuldade do OPF e OPF-*knn* em classificar corretamente as mensagens legítimas, apesar de acertarem a maioria das mensagens de spam. O melhor desempenho foi na base Enron2 que, apesar de sua taxa de SC não ser muito alta, a taxa de BH usando 3gram fica pouco acima de 1%. Esse método, inclusive, favorece o não bloqueio de mensagens legítimas, como pode ser verificado em cada base já descrita. Isso acontece por causa da semelhança encontrada em hams e pelo fato de 3gram conseguir representar melhor tais mensagens.

# Capítulo 6

# Conclusão

Pela análise dos resultados obtidos, conclui-se que o OPF, seja em sua versão com grafo completo ou seja em sua versão com grafo *knn*, não superou o classificador do estado-da-arte, ou seja, o SVM. Apesar de o tempo de treinamento deste ser muito superior ao daquele, suas medidas de desempenho, isto é, as relacionadas à eficácia do classificador, são mais satisfatórias e se aproximam do que uma aplicação real exige. Em algumas bases, o OPF conseguiu classificar corretamente a maioria das mensagens de spam, como nas bases BlogSpam, Enron1, Enron4 e Enron5. Entretanto, sua taxa de BH foi duramente penalizada, classificando erroneamente muitas mensagens legítimas como spam. Ao mesmo tempo, pode-se dizer que, quando a taxa de SC, isto é, spam corretamente classificado como spam, fica numa faixa de valores entre 50% e 80%, a quantidade de mensagens legítimas incorretamente classificadas como spam cai, chegando mesmo a 0%, ou seja, todas as mensagens legítimas são corretamente classificadas. Isso ocorre principalmente nas bases do conjunto *TubeSpam*, Enron3 e *SMS Spam Collection*. Já na base Enron2, a taxa de SC é relativamente alta, porém a taxa de BH também é, fazendo o desempenho geral do classificador cair.

Acerca dos métodos utilizados, pode-se observar que o OPF teve seus resultados melhorados em 10 das 12 bases quando passou a usar 3gram ao invés de Token. Apesar de ampliar o espaço de características, essa técnica consegue descrever melhor uma mensagem, seja ela spam ou ham, já que um conjunto maior de características é utilizado, aumentando o detalhamento de cada mensagem, o que facilita para o classificador diferenciar uma mensagem legítima de uma ilegítima. Entretanto, nota-se um pequeno aumento no tempo de treinamento de cada base dos conjuntos *BlogSpam* e *TubeSpam* quando utilizado 3gram. Novamente, isso é explicado pelo fato do aumento do número de características. A base *SMS Spam Collection*, apesar de ter o número de características aumentado, conseguiu ser treinada pelo classificador OPF usando 3gram mais rapidamente que quando utilizado o método Token. A parte mais interessante fica por conta do conjunto Enron que, indo na contramão de todas as demais bases, teve seu número de características reduzido quando foi utilizado 3gram. Isto se explica pelo fato de que mensagens em e-mail, apesar de terem assuntos mais variados, tendem a utilizar termos semelhantes, o que, quando subdivididos, geram substrings semelhantes, que são excluídas do espaço de características por não serem únicas.

Por fim, pode-se dizer que os resultados não excluem os classificadores baseados em Floresta de Caminhos Ótimos de serem utilizados em uma aplicação real. Seus tempos rápidos de treinamento, requisito para aplicações modernas de segurança da informação, os mantêm como candidatos. Porém, melhorias são necessárias quando é considerado sua eficácia, aumentando a taxa de SC e diminuindo a taxa de BH, tornando-o um classificador ideal para uso no contexto atual.

## 6.1 Trabalhos Futuros

Devidos aos resultados obtidos pelo OPF, está em desenvolvimento uma versão modificada que trata de um modo diferente as amostras no espaço de características. A intenção é fazer com que o OPF opere de modo semelhante ao SVM, ou seja, levando as características para um espaço em que seja possível uma separação linear das amostras, diminuindo drasticamente a taxa de erros. Essa versão, cujo nome é *opf_map*, criará um novo conjunto de características, que será calculado a partir da distância entre amostras. Espera-se que uma amostra de uma mensagem legítima seja bem diferente de uma amostra de uma mensagem de spam, o que, em teoria, quando analisadas pelo classificador, diminuiria a taxa de erros. Após a implantação, testes serão realizados para verificar os resultados e aplicar essa melhoria nos conjuntos de dados aqui estudados.

## 6.2 Trabalhos Publicados

Os seguintes trabalhos foram publicados no decorrer do desenvolvimento deste projeto de mestrado:

FERNANDES, D.; COSTA, K. A. P. ; ALMEIDA, T. A. ; PAPA, J. P. . SMS Spam Identification Through Optimum-path Forest-based Classifiers. In: 14th International Conference on Machine Learning and Applications, 2015, Miami. Machine Learning in Information and System Security Issues, 2015.

SILVA, L. A. ; COSTA, K. A. P. ; RIBEIRO, P. B. ; FERNANDES, D. ; PAPA, J. P. . On the Feasibility of Optimum-Path Forest in the Context of Internet-of-Things-based Applications. Recent Progress in Space Technology, v. 5, p. 1, 2015.

# Referências Bibliográficas


[1] R. Langner. Stuxnet: Dissecting a Cyberwarfare Weapon . *Security and Privacy*, 9:49–51, 2011.

[2] U.Franke and J. Brynielsson. Cyber Situational Awareness - A Systematic review of the literature. *Computer and Security*, 46:18–31, 2014.

[3] Spam and Phishing in the First Quarter of 2015. Acessado em Março de 2016.

[4] K.P.Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[5] J.P. Papa, A.X. Falcão, and C.T.N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.

[6] J.P. Papa, A.X. Falcão, V.H.C. Albuquerque, and J.M.R.S. Tavares. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, 45(1):512–520, 2012.

[7] J. Yang, L. Peng, and T. Liu. Anti-spam Model Based on AIS in Cloud Computing Environments. *Computer modelling and new technologies*, 18:97–102, 2014.

[8] N. Jatana and K. Sharma. Bayesian Spam Classification: Time Efficient Radix Encoded Fragmented Database Approach. In *International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 939–942, 2014.

[9] J.M. Rao and D.H. Reiley. The Economics of Spam. *The Journal of Economic Perspectives*, 26(3):87–110, 2012.

[10] A.G. Kakade, P.K. Kharat, A.K. Gupta, and T. Batra. Spam filtering techniques and MapReduce with SVM: A study. In *Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*, pages 59–64, 2014.

[11] T.A. Almeida and A. Yamakami. Advances in Spam Filtering Techniques. In *Computational Intelligence for Privacy and Security*, pages 199–214. Springer, 2012.

[12] P.D. Grünwald. *The Minimum Description Length Principle*. The MIT Press, 2007.

[13] T.S. Guzella and W.M. Caminhas. A review of machine learning approaches to Spam filtering . *Expert Systems with Applications*, 36:10206–10222, 2009.

[14] Portio Research. Analysis and Growth Forecasts for Mobile Messaging Markets Worldwide: 6th Edition. Technical report, Portio Research, 2011.

[15] T. Landesman. Cloudmark's 2013 Annual Global Messaging Threat Report. Technical report, CloudMark, 2014.

[16] M.H. Shirali-Shahreza and M. Shirali-Shahreza. An Anti-SMS-Spam Using CAPTCHA. In *International Colloquium on Computing, Communication, Control, and Management*, 2008.

[17] S. Gunal, A.P. Uysal, S. Ergin, and E.S Gunal. A Novel Framework for SMS Spam Filtering. In *International Symposium on Innovations in Intelligent Systems and Applications*, pages 1–4, 2012.

[18] M. Taufiq Nuruzzaman, C. Lee, and D. Choi. Independent and Personal SMS Spam Filtering. In *IEEE International Conference on Computer and Information Technology*, pages 429–435, 2011.

[19] T. Teraguchi, K. Nakamura, S. Tanaka, and K. Kitano. Detection Method of Blog Spam Based on Categorization and Time Series Information. In *26th International Conference on Advanced Information Networking and Applications Workshops*, pages 801–808, 2012.

[20] T.A. Almeida and T. C. Alberto. Learning to Block Undesired Comments in the Blogosphere. In *12th International Conference on Machine Learning and Applications*, volume 2, pages 261–266, 2013.

[21] G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. In *1st AIRWeb*, pages 1–6, 2005.

[22] C. Radulescu, M. Dinsoreanu, and R. Potolea. Identification of Spam Comments using Natural Language Processing Techniques. In *IEEE International Conference on Intelligent Computer Communication and Processing*, pages 29–35. IEEE, 2014.

[23] A. Madden, I. Ruthven, and D. McMenemy. A classification scheme for content analyses of YouTube video comments. *Journal of Documentation*, 69(5):693–714, 2013.

[24] T. C. Alberto, J. Von Lochter, and T. A. Almeida. TubeSpam: Comment Spam Filtering on YouTube. In *14th IEEE International Conference on Machine Learning and Applications*, pages 1–6, 2015.

[25] J.P. Papa and A.X. Falcão. A New Variant of the Optimum-Path Forest Classifier. In *Proceedings of the 4th International Symposium on Advances in Visual Computing*, pages 935–944, Berlin, Heidelberg, 2008. Springer-Verlag.

[26] J. P. Papa and A. X. Falcão. A Learning Algorithm for the Optimum-Path Forest Classifier. In A. Torsello, F. Escolano, and L. Brun, editors, *Graph-Based Representations in Pattern Recognition*, volume 5534 of *Lecture Notes in Computer Science*, pages 195–204. Springer Berlin Heidelberg, 2009.

[27] V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam filtering with Naive Bayes - Which Naive Bayes? In *3rd International Conference on Email and Anti-Spam*, pages 1–5, 2006.

[28] R. Beckermann, A. McCallum, and G. Huang. Automatic categorization of email into folders: benchmark experiments on Enron and SRI corpora. Technical report, University of Massachusetts Amherst, 2004.

[29] T.A. Almeida, J.M. Gómez Hidalgo, and A. Yamakami. Contributions to the study of SMS Spam Filtering: New Collection and Results. In *ACM Symposium on Document Engineering (ACM DOCENG'11)*, pages 259–262, 2011.

[30] G. Gormack. Email Spam Filtering: A Systematic Review. In *Foundations and Trends in Information Retrieval*, volume 1, pages 335–455. Now Publishers, 2008.

[31] L. Zhang, J. Zhu, and T. Yao. An Evaluation of Statistical Spam Filtering Techniques. *ACM TALIP*, 3, 2004.

[32] I. Kanaris, K. Kanaris, I. Houvardas, and E. Stamatatos. WORDS VS. CHARACTER N-GRAMS FOR ANTI-SPAM FILTERING. *International Journal on Artificial Intelligence Tools*, 20(10):1–20, 2006.

[33] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[34] S. Nissen. *Implementation of a Fast Artificial Neural Network Library (FANN)*, 2003. Department of Computer Science University of Copenhagen (DIKU). Software available at `http://leenissen.dk/fann/`.

[35] J.P. Papa, C.T.N. S., and A.X. Falcão. *LibOPF: A library for the design of optimum-path forest classifiers*, 2014. Software version 2.1 available at `http://www.ic.unicamp.br/~afalcao/LibOPF`.

[36] B.W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451, 1975.