Universidade Federal do Amazonas

Instituto de Computação
Programa de Pós-Graduação em Informática

Deteção de Spam Baseada na Evolução das Características com Presença de Concept Drift

Márcia Henke

Manaus – Amazonas – Brasil
Março, 2015.

Deteção de Spam Baseada na Evolução das Características com Presença de Concept Drift

Tese submetida ao Programa de Pós-Graduação do Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para obter o título de Doutor em Informática.

Márcia Henke

Orientador: Prof. Eduardo James Pereira Souto, DSc.
Coorientadora: Prof.ª Eulanda Miranda dos Santos, PhD.

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Henke, Márcia
H512d Deteção de Spam Baseada na Evolução das Características com Presença de Concept Drift / Márcia Henke. 2015
135 f.: il. color; 31 cm.

Orientador: Eduardo Jaimes Pereira Souto
Coorientadora: Eulanda Miranda dos Santos
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. Concept Drift. 2. Aprendizagem de Máquina. 3. Deteção Mudança. 4. Transferência de Aprendizado. I. Souto, Eduardo Jaimes Pereira II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

“Deteção de Spam Baseada na Evolução das Características com Presença de Concept Drift”

Márcia Henke

Tese defendida e aprovada pela banca examinadora constituída pelos professores:

PROF. EDUARDO JAMES PEREIRA SOUTO, DSC. - PRESIDENTE

PROF. EDUARDO LUZEIRO FEITOSA, DSC. - MEMBRO

PROF. ALTAIR OLIVO SANTIN, DSC. – MEMBRO

PROF. MOISÉS GOMES DE CARVALHO, DSC. – MEMBRO

PROF. ROGÉRIO PATRÍCIO CHAGAS DO NASCIMENTO, DSC. - MEMBRO

Março de 2015.

Aos meus filhos.

Agradecimentos

Em primeiro lugar a Deus, que me abençoou nesta caminhada e proporcionou, mesmo diante de todos os obstáculos, que eu não esmorecesse e fosse persistente.

Aos meus pais, Arcílio e Dione, que sempre com palavras de encorajamento me fortaleceram para continuar a caminhada que escolhi.

Aos meus filhos, Douglas e Bianca, que sempre foram compreensivos e meus principais incentivadores. Principalmente Douglas que tem uma compreensão melhor da situação e que com palavras fortalecedoras quando conversamos falava: "*você está quase lá mãe! Não pode desistir!"*. Você não imagina o quanto isso me fortalecia!

Aos meus orientadores Eduardo Souto e Eulanda Santos, que sempre se mostraram disponíveis e pacientes em ouvir minhas colocações e com muita sabedoria me conduziram nessa caminhada, pois sem a experiência que me foi passada não saberia contornar os obstáculos nesta trajetória.

A secretária administrativa do programa de pós-graduação, Elienai Nogueira e toda a equipe que lhe assiste, que com seu profissionalismo e competência nos orienta na parte burocrática e auxilia em todas as etapas e cumprimentos de protocolos estabelecidos pelo programa.

A toda equipe de doutores que compõe o corpo docente do Instituto de Computação da UFAM por compartilhar seus conhecimentos e suas experiências que de alguma maneira, ou por meio das disciplinas ministradas, ou mesmo, em uma conversa informal pelos corredores do Instituto, colaboraram com seu conhecimento e otimismo.

Aos meus colegas e professores do grupo de pesquisa que compartilham as dificuldades que passam e também sempre tem uma palavra de incentivo para nos fortalecer. Obrigada: Prof. Dr. Eduardo Feitosa, Prof. Dr. Davi Viana, Dr. Rodrigo Ribeiro de Oliveira, Gilbert Martins, Kaio Rafael, Eduardo Nunan, Clayton Maia, Alex Monteiro, Wesllen Sousa, Thiago Rocha, Francisco Ivan R. de Andrade.

Um agradecimento especial ao Prof. Dr. Ruiter Caldas pela sua simplicidade e carisma no tratar sempre que eu lhe recorria com alguma dúvida seja sobre a disciplina de FTC ou com relação à burocracia do Instituto de Computação.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e Fundação de Apoio Institucional Rio Solimões (UNISOL) pelo suporte financeiro para que eu pudesse me dedicar exclusivamente à pesquisa.

" (...) *nós escolhemos ir à lua (...) e fazer outras coisas, não porque elas são fáceis, mas porque são difíceis, porque esse objetivo servirá para organizar e medir o melhor de nossas energias e aptidões, porque esse é um desafio que estamos dispostos a aceitar, (...) e o único que pretendemos vencer..."*

(J. F. Kennedy, *Rise University Stadium* - em setembro de 1962)

Resumo

As mensagens eletrônicas (e-mails) ainda são consideradas as ferramentas de maior prestígio no meio empresarial e pessoal, pois apresentam baixo custo e facilidade de acesso. Por outro lado, os e-mails tornaram-se um grande problema devido à elevada quantidade de mensagens não desejadas, denominadas *spam*, que lotam as caixas de e-mails dos usuários. Dentre os diversos problemas causados pelas mensagens *spam*, destaca-se o fato de ser atualmente o principal vetor de propagação de atividades maliciosas como vírus, *worms,* cavalos de Tróia, *phishing, botnets*, dentre outros. Tais atividades permitem ao atacante acesso indevido a dados sigilosos, segredos de negócios ou mesmo invadir a privacidade das vítimas para obter alguma vantagem.

Diversas abordagens, comerciais e acadêmicas, têm sido propostas para impedir o envio de mensagens de e-mails indesejados como filtros implementados nos servidores de e-mail, mecanismos de classificação de mensagens de *spam* para que os usuários definam quando determinado assunto ou autor é fonte de propagação de *spam* e até mesmo filtros implementados em componentes eletrônicos de rede. Em geral, as abordagens de filtros de e-mail são baseadas na análise do conteúdo das mensagens para determinar se tal mensagem é ou não um *spam*.

Um dos maiores problemas com essa abordagem é a deteção de *spam* na presença de *concept drift.* A literatura conceitua *concept drift* como mudanças que ocorrem no conceito dos dados ao longo do tempo como a alteração das características que descrevem um ataque ou ocorrência de novas características. Muitos Sistemas de Deteção de Intrusão (IDS) usam técnicas de aprendizagem de máquina para monitorar a taxa de erro de classificação no intuito de detetar mudança. Entretanto, quando a deteção ocorre, algum dano já foi causado ao sistema, fato que requer atualização do processo de classificação e a intervenção do operador do sistema.

Com o objetivo de minimizar os problemas mencionados acima, esta tese propõe um método de deteção de mudança, denominado Método orientado à Análise da Evolução das Características de Ataques (MECA). O método proposto é composto por três etapas:

1) treino do modelo de classificação; 2) deteção de mudança; e 3) transferência do aprendizado. A primeira etapa emprega modelos de classificação comumente adotados em qualquer método que utiliza aprendizagem de máquina. A segunda etapa apresenta duas novas estratégias para contornar *concept drift*: HFS (*Historical-based Features Selection*) que analisa a evolução das características com base no histórico ao longo do tempo; e SFS (*Similarity-based Features Selection*) que observa a evolução das características a partir do nível de similaridade obtido entre os vetores de características dos domínios fonte e alvo. Por fim, a terceira etapa concentra seu objetivo nas seguintes questões: o que, como e quando transferir conhecimento adquirido. A resposta à primeira questão é fornecida pelas estratégias de deteção de mudança, que identificam as novas características e as armazenam para que sejam transferidas. Para responder a segunda questão, a abordagem de transferência de representação de características é adotada. Finalmente, a transferência do novo conhecimento é realizada tão logo mudanças que comprometam o desempenho da tarefa de classificação sejam identificadas.

O método MECA foi desenvolvido e validado usando duas bases de dados públicas, sendo que uma das bases foi construída ao longo desta tese. Os resultados dos experimentos indicaram que é possível inferir um limiar para detetar mudanças a fim de garantir o modelo de classificação sempre atualizado por meio da transferência de conhecimento. Além disso, um diferencial apresentado no método MECA é a possibilidade de executar a tarefa de classificação em paralelo com a deteção de mudança, sendo as duas tarefas independentes. Por fim, o MECA utiliza o algoritmo de aprendizagem de máquina SVM (*Support Vector Machines*), que é menos aderente às amostras de treinamento. Os resultados obtidos com o MECA mostraram que é possível detetar mudanças por meio da evolução das características antes de ocorrer uma degradação significativa no modelo de classificação utilizado.

Palavras chave: *concept drift*, aprendizagem de máquina, deteção mudança, transferência de aprendizado.

Abstract

Electronic messages (emails) are still considered the most significant tools in business and personal applications due to their low cost and easy access. However, e-mails have become a major problem owing to the high amount of junk mail, named spam, which fill the e-mail boxes of users. Among the many problems caused by spam messages, we may highlight the fact that it is currently the main vector for the spread of malicious activities such as viruses, worms, trojans, phishing, botnets, among others. Such activities allow the attacker to have illegal access to penetrating data, trade secrets or to invade the privacy of the sufferers to get some advantage.

Several approaches have been proposed to prevent sending unsolicited e-mail messages, such as filters implemented in e-mail servers, spam message classification mechanisms for users to define when particular issue or author is a source of spread of spam and even filters implemented in network electronics. In general, e-mail filter approaches are based on analysis of message content to determine whether or not a message is spam.

A major problem with this approach is spam detection in the presence of concept drift. The literature defines concept drift as changes occurring in the concept of data over time, as the change in the features that describe an attack or occurrence of new features. Numerous Intrusion Detection Systems (IDS) use machine learning techniques to monitor the classification error rate in order to detect change. However, when detection occurs, some damage has been caused to the system, a fact that requires updating the classification process and the system operator intervention.

To overcome the problems mentioned above, this work proposes a new changing detection method, named Method oriented to the Analysis of the Development of Attacks Characteristics (MECA). The proposed method consists of three steps: 1) classification model training; 2) concept drift detection; and 3) transfer learning. The first step generates classification models as it is commonly conducted in machine learning. The second step introduces two new strategies to avoid concept drift: HFS (Historical-based Features Selection) that analyzes the evolution of the features based on over time historical; and SFS (Similarity-based Features Selection) that analyzes the evolution of the features from

the level of similarity obtained between the features vectors of the source and target domains. Finally, the third step focuses on the following questions: what, how and when to transfer acquired knowledge. The answer to the first question is provided by the concept drift detection strategies that identify the new features and store them to be transferred. To answer the second question, the feature representation transfer approach is employed. Finally, the transfer of new knowledge is executed as soon as changes that compromise the classification task performance are identified.

The proposed method was developed and validated using two public databases, being one of the datasets built along this thesis. The results of the experiments shown that it is possible to infer a threshold to detect changes in order to ensure the classification model is updated through knowledge transfer. In addition, MECA architecture is able to perform the classification task, as well as the concept drift detection, as two parallel and independent tasks. Finally, MECA uses SVM machine learning algorithm (Support Vector Machines), which is less adherent to the training samples. The results obtained with MECA showed that it is possible to detect changes through feature evolution monitoring before a significant degradation in classification models is achieved.

Keywords: concept drift, machine learning, intrusion detection, transfer learning.

Abreviações

CART	Classification and Regression Trees
CBR	Case-Based Reasoning
ECUE	Email Classification Using Examples
ETSS	Emerging Technologies and System Security
FAE	Feature Adaptive Ensemble
FN	False Negatives
FP	False Positives
HFS	Historical-based Features Selection
IDS	Intrusion Detection Systems
MECA	**M**étodo de deteção de mudança baseado na **E**volução das **C**aracterísticas de **A**taques
RF	Random Forests
SFS	Similarity-based Features Selection
SVD	Singular Value Decomposition
SVM	Support Vector Machines
TN	True Negatives
TP	True Positives

Conteúdo

Resumo 7

Abstract 9

Lista de Figuras 14

Capítulo 1 Introdução 17

1.1. Motivação 19

1.2. Objetivos 20

1.3. Contribuições 21

1.4. Estrutura da Tese 22

Capítulo 2 Fundamentação Teórica 24

2.1. Aprendizagem de Máquina 24

2.1.1. Definição 24

2.1.2. Técnicas de Aprendizagem de Máquina 25

2.1.2.1. SVM (Support Vector Machines) 25

2.1.2.2. K-means 27

2.2. Deteção de Mudança 29

2.2.1. Terminologia 29

2.2.2. Definição 31

2.2.3. Tipos de *Concept Drift* 32

2.2.4. Métodos de Reação a *Concept Drift* 34

2.2.5. Método de Deteção de *Concept Drift* 36

2.3. Transferência de Aprendizado 37

2.3.1. Terminologia 38

2.3.2. Definição 38

2.3.3. Arranjo para Transferência de Aprendizado 39

2.3.4. Transferência de Aprendizado Transdutivo 42

Capítulo 3 Trabalhos Relacionados 44

3.1. Concept Drift 44

3.2. Transferência de Aprendizado 53

Capítulo 4 Método de Deteção de Spam Baseado na Evolução das Características de Ataque 60

4.1. MECA: Visão Arquitetural 60

4.2. Entrada de Dados 63

4.3. Etapa de Treino e Construção do Modelo de Classificação 64

4.4. Etapa de Deteção de Mudança 66

4.4.1. *Historical-based Features Selection* (HFS) 67

4.4.2. *Similarity based Features Selection* (SFS) 71

4.5. Etapa de Transferência de Aprendizado 73

4.6. Considerações **75**

Capítulo 5 Experimentos e Resultados ***77***

5.1. Protocolo Experimental **77**

5.1.1 Bases de Dados **77**

5.1.1.1. ECUE (Email Classification Using Examples) 77

5.1.1.2. ETSS (Emerging Technologies and Security System) 79

5.1.2. Medidas de Desempenho **80**

5.1.3. Classificador **81**

5.2. Experimentos **82**

5.2.1. Avaliação do Comportamento Estático e da Evolução das Características **83**

5.2.1.1. Experimentos 83

a) Série 1 - Information Gain somente no Treino 83

b) Série 2 - Information Gain no Teste 84

5.2.1.2. Análise dos Resultados 84

5.2.1.3. Discussão 87

5.2.2. HFS (Historical-based Feature Selection) **88**

5.2.2.1. Experimentos 90

a) Série 3 – HFS em janelas de tempo trimestrais 90

b) Série 4 – HFS em janelas de tempo mensais 90

5.2.2.2. Análise dos Resultados 91

5.2.2.3. Discussão 94

5.2.3. Similarity based Features Selection (SFS) **95**

5.2.3.1. Experimentos 95

a) Série 5 – SFS 96

5.2.3.2. Análise dos Resultados 97

5.2.3.3. Discussão 102

Capítulo 6 Conclusões e Trabalhos Futuros ***104***

6.1. Resumo dos Trabalhos Realizados **104**

6.2. Trabalhos Futuros **107**

Referências ***109***

Apêndice A Construção Base de dados 115

Apêndice B Similaridade empregando Distância Euclidiana 131

Lista de Figuras

Figura 2.1. Modelo genérico de aprendizagem de máquina. 25

Figura 2.2. Hiperplano de separação ótima para um problema com duas classes..... 26

Figura 2.3. Mapeamento do espaço de entrada via função *kernel*..... 27

Figura 2.4. Ilustração dos Quatro Tipos de Estrutura de *Concept Drift* (adaptada de Zliobaite [27]) 34

Figura 2.5. Uma Visão das Diferentes Configurações de Transferência de Aprendizado, adaptado de Pan [50]. 40

Figura 3.1 Janela de tempo deslizante sobre exemplos apresentando descrições correntes e antigas [34]. 45

Figura 3.2 Parcial caracterização [55] 46

Figura 4.1. Visão geral da Arquitetura MECA. MECA é dividido em três etapas. A etapa de treino e construção do classificador gera o modelo de predição, enquanto a etapa de deteção de mudança observa um limiar para determinar uma mudança e a etapa de Transferência de aprendizado atualiza o domínio fonte..... 63

Figura 4.2. O processo de pré-processamento de dados para o MECA. A partir de uma coleção de dados brutos são selecionados dados de interesse para iniciar o pré-processamento aplicando alguma técnica para reduzir a dimensionalidade do espaço de características e transformar tais dados em vetores de características. 64

Figura 4.3. Etapa de Treino e Construção Modelo de Classificação. O conjunto de treino analisa e refina a configuração dos parâmetros para o modelo de classificação, medindo seu desempenho sobre o conjunto de validação. 66

Figura 4.4. Estratégia HFS observa o histórico das características baseado no parâmetro peso para estimar o novo conhecimento; (a) análise entre os vetores (x0 , x1) de representação do problema, domínio fonte e alvo; (b) geração do novo vetor de características mais relevantes baseada na frequência e peso..... 69

Figura 4.5. Algoritmo da estratégia HFS 70

Figura 4.6. Etapa de Deteção de Mudança determina o momento de uma mudança virtual e/ou real. Enquanto a mudança não é detectada, a tarefa de classificação é enviada para o classificador originado no instante de tempo t0 e as características identificadas com maior relevância entre os dois domínios são armazenadas em um repositório de mudanças virtual que construirá um novo conhecimento. 73

Figura 4.7. Etapa de Transferência de Aprendizado se preocupa em o que, como e quando transferir o novo conhecimento para atualizar o domínio fonte. Instâncias antigas são descartadas para manter o balanceamento das amostras entre as classes da representação do problema. 75

Figura 5.1. Formato das amostras que compõem a base de dados ECUE. 78

Figura 5.2. Taxa de erro obtida usando um modelo de classificação estático sobre a base de dados ECUE - Série 1 85

Figura 5.3. Taxas de erros obtidas usando um modelo de classificação estático (Série 1) e um modelo atualizado mensalmente (Série 2). 86

Figura 5.4. Percentual de características ausentes comparando os domínios fonte e alvo para a base de dados ECUE. 87

Figura 5.5. Vetor de características que descreve a janela de tempo de fevereiro composta por 293 e-mail. 89

Figura 5.6. Vetor de características que descreve a janela de tempo de março composta por 447 e-mail. 89

Figura 5.7. Percentuais de taxa de erro obtidos nas Séries de Experimentos 1 e 3 sobre a base de dados ECUE. 92

Figura 5.8. Percentuais de taxa de erro obtidos nas Séries de Experimentos 1 e 4 sobre a base de dados ECUE. 92

Figura 5.9. Comparação de percentuais de taxas de erro atingidos com a Série 3 e 4 sobre a base de dados ECUE. 93

Figura 5.10. Dispersão dos dados das Séries de experimentos 1, 3 e 4 sobre a base de dados ECUE. 94

Figura 5.11. Dispersão das amostras de e-mail nas janelas de teste de setembro a dezembro de 2012, em relação as amostras de e-mail *spam* do domínio de treino por meio da medida de similaridade. 97

Figura 5.12. Desempenho do modelo de classificação aplicando a Série 1 (aprendizagem de máquina tradicional) e Série 5 (estratégia SFS) sobre a base de dados ETSS. 98

Figura 5.13. Comparação entre as medidas de similaridade resultantes para as Séries 1 e 5. 99

Figura 5.14. Desempenho do modelo de classificação sobre janelas de tempo diárias, com a Série 5 sobre base de dados ETSS – setembro 2012. 100

Figura 5.15. Desempenho do modelo de classificação sobre janelas de tempo diárias, com a Série 5 sobre base de dados ETSS – setembro 2012. 101

Figura 5.16. Desempenho do modelo de classificação sobre janelas de tempo diárias, com a Série 5 sobre base de dados ETSS – setembro 2012. 101

Figura A.1. Desempenho do classificador com todas as características extraídas via BoW. Classificador SVM treinado e testado com estratégia de validação cruzada usando 10 partições. 120

Figura A.2. Taxa de erro obtida usando um modelo de classificação estático sobre a base de dados ETSS - Série 1 126

Figura A.3. Taxas de erros obtidas usando um modelo de classificação estático (Série 1) e um modelo atualizado mensalmente (Série 2) sobre base de dados ETSS. 127

Figura A.4. Percentual de características ausentes comparando os domínios fonte e alvo para a base de dados ETSS. 129

Lista de Tabelas

Tabela 2.1. Funções de *Kernel* mais utilizadas com SVM 27

Tabela 2.2. Diferentes Abordagens para Transferência de Aprendizado [50]. 41

Tabela 2.3. Diferentes Abordagens usadas em Diferentes Configurações para Transferência de Aprendizado [50] 42

Tabela 3.1. Comparação dos Trabalhos Relacionados considerando os Métodos de Deteção e Reação à Deteção de Mudança (*Concept Drift*) 51

Tabela 3.2. Comparação dos Trabalhos Relacionados considerando os Abordagem de Transferência de Aprendizado e Estratégia empregada 58

Tabela 5.1. Distribuição de dados da base de treino ECUE. O rótulo S/data indica que a amostra não tem data. 79

Tabela 5.2. Distribuição de dados da base de teste ECUE. 79

Tabela 5.3. Distribuição de dados da base de treino ETSS. 80

Tabela 5.4. Distribuição de dados da base de teste ETSS. 80

Tabela 5.5. Matriz de confusão 81

Tabela 5.6. Quantidade de amostras usadas para atualizar modelo de classificação em janelas de tempo diária considerando classe *spam* e legítima – setembro de 2012. 102

Tabela 5.7. Quantidade de amostras de e-mails usadas para atualizar modelo de classificação em janelas de tempo Diárias para classe *spam* e legítima – outubro de 2012 102

Tabela 5.8. Quantidade de amostras de e-mails usadas para atualizar modelo de classificação em janelas de tempo Diárias para classe *spam* e legítima – novembro de 2012. 102

Tabela A.1. Distribuição Base de Dados ETSS Bruta 118

Tabela A.2. Distribuição da Base de Dados ETSS 119

Tabela A.3. Representação das classes de Spam e Nonspam por meio da quantidade de Características presente em cada classe e quantidade de características Spam encontradas no vetor de Características Nonspam 121

Tabela A.4. Distribuição das amostras de treino/domínico fonte – ETSS 122

Tabela A.5. Redução da Dimensionalidade do Espaço de Características 123

Tabela A.6. Ajuste Parâmetros para Domínio Fonte 124

Tabela A.7 Distribuição das amostras de teste - ETSS 124

Capítulo 1
Introdução

Com uma rápida consulta às estatísticas de tráfego na Internet, percebe-se que o tráfego conhecido, solicitado, ou em outras palavras, legítimo, foi dominado pelo tráfego de mensagens eletrônicas não solicitadas (*spam*), atividades fraudulentas como *phishing*[1] e *pharming*[2], ataques de negação de serviço, proliferação de vírus, *worms* e outros códigos maliciosos, ou seja, tráfego ilegítimo, improdutivo, não desejado e não solicitado.

A título de exemplo, em Maio de 2012, segundo o relatório anual de ameaças de segurança do grupo *SophosLab*, foi capturado um *botnet* que espalhava *malware* e mensagens *spam*. Aproximadamente 30 milhões de computadores faziam parte da rede de disseminação desses ataques maliciosos, chegando a um montante de 30 bilhões de e-mails infectados.

No Brasil, o CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil) também contabilizou, no ano de 2014, um total de 993.088 reclamações relacionadas ao envio de *spam*. Desse montante, 29% corresponde a e-mails *spam* que foram utilizados como vetores para direcionamento de páginas de propaganda, máquinas identificadas como emissoras de e-mails *spam* e proxies. O restante (71%) corresponde a reclamações de fontes diversas, devidamente investigadas e comprovadas pelo CERT[1].

Muitos pesquisadores atribuem o aumento dessas atividades maliciosas ao fato da Internet ser um ambiente sem controle e sem critérios de segurança claramente estipulados, além da ineficiência das atuais soluções como *firewalls*, *anti-spam*, *anti-malware*, *anti-phishing* e IDSs (*Intrusion Detection Systems*) [2][3][4][5].

Tipicamente, a efetividade fornecida pelas atuais soluções é comprometida pelas altas taxas de falsos alarmes, pela falta de cooperação com outras soluções ou por

---

1 *Phishing* é um tipo de fraude eletrônica caracterizada pela tentativa de obter informações pessoais privilegiadas por meio de sites falsos ou mensagens eletrônicas forjadas.

2 *Pharming* refere-se ao ataque de envenenamento de *cache* DNS cujo objetivo é preparar terreno para atividades de *phishing*.

vulnerabilidades na infraestrutura de rede. Além disso, muitas das soluções dependem de ativação ou interferência humana, na forma de configuração, adequação e ajuste, para funcionar.

Na busca por soluções mais efetivas e corretas, especialmente aquelas relacionadas à segurança na Internet, a comunidade de pesquisadores de segurança tem investido tempo e recursos para tornar a aprendizagem de máquina uma ferramenta poderosa e eficaz nessa área. A ideia é bastante simples: se as máquinas podem aprender quando um sistema está funcionando normalmente e quando está sob ataque, então é possível construir mecanismos capazes de, automaticamente, responder a ataques e anomalias normalmente encontradas em uma rede.

Entretanto, a maioria dos sistemas propostos atua de forma estática, ou seja, os sistemas são criados com base nas características dos ataques obtidas em um determinado momento. Por exemplo, os antivírus atuais baseiam seu processo de deteção na análise de uma sequência de caracteres que representa um vírus (assinatura). Tal abordagem tem sido completamente evadida pela maioria dos vírus metamórficos que automaticamente alteram seus códigos à medida que se propagam. Os sistemas *anti-phishing* e *anti-spam* baseiam suas soluções por meio da observação do comportamento e das características dos ataques em um instante *t* do tempo. Porém, o comportamento e as características dos ataques tratados por essas ferramentas mudam com o passar do tempo. Por este motivo, a eficiência de tais soluções é completamente dependente do processo de atualização das bases de dados e dos serviços fornecidos por estas.

Porém, devido às limitações das soluções estáticas, na literatura em aprendizagem de máquina já é possível encontrar propostas de soluções para problemas que evoluem com o tempo, como é caso da maioria dos ataques [6][7][8][9]. Essas soluções podem ser divididas em dois grupos: implícitas e explícitas. Nos métodos implícitos, a base de dados é atualizada e o algoritmo de aprendizagem de máquina é re-treinado em instantes de tempo *t* pré-definidos. Logo, são métodos que apresentam elevado custo computacional, especialmente porque atualizam o sistema mesmo sem ocorrência de mudanças no problema. Por outro lado, os métodos explícitos buscam primeiramente detetar a ocorrência de mudanças para então, atualizar o sistema. Entretanto, o mecanismo de deteção normalmente é baseado no monitoramento da taxa de acerto do sistema. Portanto, a taxa de acerto do sistema precisa reduzir bruscamente para que a mudança seja detectada. Essa situação certamente implica em prejuízos causados pelo sistema. No caso

de deteção de *spam*, por exemplo, muitos falsos positivos e falsos negativos serão emitidos pelo sistema antes que a mudança seja detectada.

Para tentar minimizar os efeitos de algumas dessas deficiências, esta tese propõe que o processo de deteção de ataques ocorra por meio da análise da evolução das características dos ataques para que as informações que descrevem os ataques sejam atualizadas em função dessa evolução. Para isso, esta tese propõe dois métodos de deteção de ataques orientados à análise da Evolução das Características dos Ataques (MECA), cujo processo de monitoramento da evolução das características dos ataques é baseado em técnicas de deteção de mudança (do inglês, *Concept Drift*) [10] e a atualização do modelo de classificação é baseada no uso de transferência de aprendizagem (do inglês, *Transfer Learning*) [11].

O conceito de *concept drift* é utilizado para determinar o momento em que ocorre uma evolução significativa das características, fato que indica explicitamente uma mudança no ambiente de deteção de ataque. Técnicas de transferência de aprendizagem são empregadas para manter atualizada a base de conhecimento sobre os ataques.

1.1. Motivação

A questão chave na tentativa de controlar a aceitação de e-mails *spam* está na natureza desse tipo de ataque. O processo de deteção de *spam* é complexo por se tratar de um ataque que apresenta um comportamento dinâmico. Vários fatores podem ser considerados como impactantes e inibem a deteção de *spam*, como: similaridade cada vez maior entre *spam* e e-mails legítimos; ocorrências sazonais aproveitadas pelos *spammers* como promoções de datas específicas (natal, "*black Friday"*, dia dos namorados, "*thanksgiving"*) e notícias bombásticas (por exemplo, a morte de "Steve Jobs") são usadas para propagar outros ataques.

Além de todas as situações anteriormente discutidas, os atacantes empregam diferentes técnicas para ludibriar os filtros ou sistemas de deteção de *spam*. Artifícios técnicos como o emprego de palavras-chave com grafias diferentes, tais como "*free*" e "viagra" grafadas como *"fr33", "f.r.e.e"*, *"v.i.a.g.r.a", "v1agra" e "vi@gr@"* são comumente utilizados. As estratégias empregadas pelos atacantes tornam cada vez mais dinâmicas as características que descrevem este tipo de ataque, o que reforça a dificuldade e a complexidade de se manter sistemas desse tipo sempre atualizados.

Inúmeras aplicações têm sido propostas na literatura para deteção de ataques *spam*. Isso reforça a dificuldade do problema e a busca por soluções mais próximas do ideal e mais adequadas. Por outro lado, estudos realizados em [12]–[15] demonstram que a maioria das aplicações que utilizam aprendizagem de máquina para filtrar e-mail e classificá-los em mensagens *spam* ou legítimas são deficientes quando o conjunto de dados que representa o problema torna-se desatualizado e não mais representativo. Além disso, dentre as aplicações que buscam detetar a necessidade de mudança no modelo de classificação, a maioria utiliza o monitoramento da taxa de precisão do modelo para tomar a decisão de atualizá-lo. Conforme discutido no início desta introdução, essa forma de deteção é deficiente por depender de uma queda brusca na taxa de precisão do modelo para, então, indicar a necessidade de atualização.

Esta tese busca contribuir com o estado-da-arte ao propor um método diferenciado, denominado MECA, que determine o momento ideal para a atualização da base de conhecimento para gerar um novo modelo de classificação, sem que essa ação cause um prejuízo ou dano ao sistema. A arquitetura do método MECA apresenta três etapas bem definidas e independentes, possibilitando o emprego de diferentes estratégias sem afetar o funcionamento da arquitetura do método, e podendo abordar diferentes tipos de ataques. São desenvolvidas duas estratégias de deteção de mudança empregadas na segunda etapa da arquitetura proposta no método MECA. Tais estratégias empregam dois conceitos da área de aprendizagem de máquina: *concept drift* e transferência de aprendizado.

1.2. Objetivos

O objetivo geral desta tese é desenvolver e demonstrar que um método de deteção de ataque *spam*, baseado na evolução das características de ataques - MECA, pode determinar automaticamente o momento de mudança no comportamento deste ataque e transferir o novo conhecimento para atualizar a base de dados empregando um limiar de deteção de mudança para efetuar a transferência imediatamente. Para este fim, será utilizada aprendizagem de máquina, empregando os conceitos de transferência do aprendizado e *concept drift*.

Os objetivos específicos são descritos a seguir:

1. Definir uma estratégia para detetar mudança na presença de *concept drift* por meio

do monitoramento das características que descrevem as classes de e-mails legítimos e *spam*, estipulando um limiar para deteção de mudança;

2. Definir uma estratégia para transferência de aprendizado que efetue transferência de maneira automática e que mantenha o conhecimento do domínio de treino sempre atualizado de maneira que o modelo de classificação não degrade seu desempenho ao longo do tempo;
3. Protótipo.
4. Validação.
5. Construção de base de dados pública.

1.3. Contribuições

A partir dos objetivos definidos, esta tese apresenta as seguintes contribuições:

1. O método desenvolvido. MECA apresenta uma característica importante, como segue: as etapas de deteção de mudança e transferência de aprendizado apresentam flexibilidade para empregar diferentes estratégias nestas etapas, ou seja, MECA possui uma arquitetura não é dependente das estratégias desenvolvidas. O método proposto é original no sentido de proporcionar alterações no emprego de estratégias, sem afetar a automatização do processo de deteção de atividades maliciosas;
2. Duas estratégias de deteção de mudança:
    2.1. HFS – *History-based Feature Selection*, onde o processo de deteção de mudança ocorre por meio da análise dos conjuntos de características ao longo do tempo, sendo que um parâmetro peso é utilizado para determinar as características mais relevantes. O vetor de características do domínio de treino é comparado aos vetores de características correspondentes das janelas de tempo seguintes. As características que apresentam os maiores valores de peso e as características que não se encontravam no domínio de treino, mas são relevantes no domínio alvo, são selecionadas para compor a nova representação do conhecimento;
    2.2. SFS – *Similarity-based Features Selection,* onde o processo de deteção de mudança ocorre por meio do uso do cálculo do modelo vetorial para analisar

a similaridade entre os vetores de características dos domínios de treino e de teste. Quanto mais rara uma característica, menor será sua taxa de similaridade e maior será sua relevância para o processo de deteção de mudança.

3. Uma nova base de dados de ataques *spam*, *spam dataset* ETSS (*Emerging Technologies and System Security*), construída com mensagens coletadas no período de Junho de 2012 a Outubro de 2013. A base será disponibilizada para ser utilizada em outras pesquisas na área de aprendizagem de máquina e segurança em redes de computadores.
4. A estratégia de transferência de aprendizado é intrínseca às estratégias de deteção de mudança. Os requisitos de transferência (o que, quando e como transferir) são atendidos a partir da observação da evolução das características ao longo do tempo, empregando a abordagem de transferência de características.

A seguir, são descritas as produções científicas referentes a resultados obtidos nesta tese, aceitas:

- Henke, M., et al. "*Aprendizagem de Máquina para Segurança em Redes de Computadores: Métodos e Aplicações*". Minicurso. XI Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais (SBSeg 2011), pp. 53-103, Brasília, Distrito Federal, novembro, 2011.
- Henke, M., Souto, E., Santos, E., M. "*Deteção de Intruso usando Conjunto de k-NN gerado por Subespaços Aleatórios*". XI Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais (SBSeg 2011), Brasília, Distrito Federal, novembro, 2011.
- Henke, M., Souto, E., Santos, E., M. *"Analysis of the Evolution of Features in Classification Problems with Concept Drift: Application to Spam Detection"*. IFIP/IEEE IM 2015 Symposium on Integrated NetWork Management (IM 2015).

1.4. Estrutura da Tese

O restante desta tese está organizado como segue:

O Capítulo 2 discute os dois principais conceitos usados nesta tese: deteção de mudança (*concept drift*) e transferência de aprendizado. As definições dos dois conceitos

são apresentadas, assim como os tipos de abordagens utilizadas para identificar mudanças e transferir aprendizado. Além disso, alguns trabalhos relacionados empregando os dois conceitos são discutidos. Finalizando, é apresentada uma síntese de ambos os conceitos no sentido de relacionar as abordagens que melhor atendem aos objetivos desta pesquisa.

O Capítulo 3 apresenta os trabalhos relacionados divididos em duas seções baseadas nos conceitos empregados nesta tese: *concept drift* e transferência de aprendizado. A análise efetuada sobre os trabalhos relacionados é voltada à forma como os trabalhos mencionados determinam os seguintes pontos: o momento da deteção de mudança em ambientes dinâmicos, como a mudança detectada é contornada, abordagens para construir e transferir o aprendizado e como atualizar o novo conhecimento a partir do aprendizado adquirido.

O Capítulo 4 trata do método de deteção de ataque *spam*, baseado na evolução das características de ataques (MECA). É apresentada uma visão geral da arquitetura e uma breve comparação com as aplicações atuais. Em seguida, as três etapas que compõem a arquitetura são descritas com mais detalhes. As estratégias empregadas em cada etapa são apresentadas, assim como a forma de utilização dos conceitos discutidos no Capítulo 2.

O Capítulo 5 apresenta os experimentos e resultados obtidos com as estratégias desenvolvidas para deteção de mudança e transferência de aprendizado. Inicialmente, o protocolo experimental é descrito, incluindo as bases de dados, as medidas para avaliar o desempenho do modelo de classificação e o classificador SVM (*Support Vector Machine*). Os experimentos são conduzidos incialmente pela avaliação da efetividade do modelo de classificação empregando aprendizagem de máquina tradicional. Na sequência, as estratégias propostas são avaliadas por meio da comparação de desempenho do modelo de classificação sobre janelas de tempo trimestrais, mensais e diárias. Ao final dos resultados de cada estratégia, uma breve discussão é apresentada.

Por fim, o Capítulo 6 apresenta as conclusões e a descrição de alguns trabalhos a serem realizados no futuro.

Capítulo 2

Fundamentação Teórica

Este capítulo introduz a fundamentação teórica ao entendimento desta tese. O capítulo inicia com uma breve explanação sobre aprendizagem de máquina, técnicas de aprendizagem de máquina, deteção de mudança e transferência de aprendizado e demais conceitos relacionados ao escopo da tese.

2.1. Aprendizagem de Máquina

Aprendizagem de máquina é parte da área de Inteligência Artificial. Para ser inteligente um sistema que está em um ambiente dinâmico deveria ter a habilidade de aprender. Se o sistema pode aprender e adaptar-se as mudanças, o projetista do sistema não precisa prever e oferecer soluções para todas as situações possíveis [Alpaydin, 2010].

Para entender como ocorre o processo de aprendizagem e identificar a evolução nos padrões que determinam se um e-mail é legítimo ou *spam*, serão apresentados, a seguir, a definição de aprendizagem de máquina e uma breve descrição sobre as duas técnicas de aprendizagem de máquina empregadas nesta tese.

2.1.1. Definição

Existem inúmeras atividades associadas à noção de aprendizagem, dificultando a definição exata do termo e tornando-o dependente de contexto. Porém, no contexto de aplicações computacionais, uma definição muito precisa de aprendizagem de máquina pode ser encontrada em [Alpaydin, 2010]:

"... são programas de computador utilizados para otimizar um critério de desempenho, usando dados de exemplo ou experiência do passado".

Independente da definição exata de aprendizagem, é amplamente aceito na literatura que sistemas de aprendizagem confiáveis são de importância estratégica em diversas áreas de aplicação, pois há muitas tarefas que não podem ser solucionadas por

meio do uso de técnicas de programação clássicas. Podemos citar como exemplo, a classificação de e-mails em duas classes: legítimos e *spam*. Nesse tipo de problema a entrada é conhecida: um documento e-mail que em seu caso mais simples é um arquivo texto; e a saída também é conhecida: *spam* ou não *spam*. Porém, a forma como a entrada deverá ser convertida na saída é desconhecida.

Os algoritmos de aprendizagem de máquina, portanto, podem ser a chave para solucionar problemas dessa natureza, pois são algoritmos que podem "aprender" a definir padrões das classes envolvidas no problema, a partir de exemplos reais obtidos do ambiente. A Figura 2.1 mostra um modelo genérico de aprendizagem de máquina. O ambiente fornece informação para um elemento de aprendizagem que usa essa informação para fazer melhoramentos em uma base de conhecimento e então, um elemento de desempenho usa essa base para executar sua tarefa.

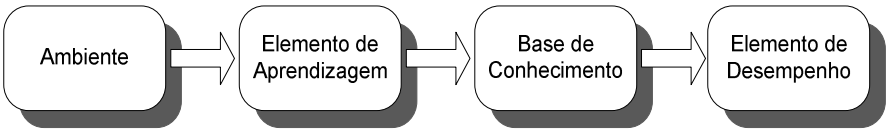

Figura 2.1. Modelo genérico de aprendizagem de máquina.

2.1.2. Técnicas de Aprendizagem de Máquina

Esta seção apresenta duas de uma variedade de técnicas de aprendizagem de máquina encontradas na literatura. As duas técnicas apresentadas são empregadas nesta tese, SVM (*Support Vector Machines*) e *k-Means*.

2.1.2.1. SVM (Support Vector Machines)

É uma técnica de classificação amplamente aplicada em problemas de segurança de redes tais como deteção de *phishing* [16] e deteção de intrusos [17]. Basicamente, o funcionamento de SVM pode ser descrito da seguinte forma: dadas duas classes e um conjunto de instâncias de treinamento cujas amostras pertencem a essas classes, SVM constrói um hiperplano que divide o espaço de características em duas regiões, maximizando a margem de separação entre as mesmas. Esse hiperplano é conhecido

como hiperplano de separação ótima. As amostras desconhecidas (exemplos de teste) são então mapeadas para esse mesmo espaço, e atribuídas a uma das classes [18].

A Figura 2.2 mostra o hiperplano de separação ótima (reta separadora) para um problema bidimensional típico e linearmente separável. As retas pontilhadas H_1 e H_2, paralelas ao hiperplano, constituem o par de hiperplanos que geram a margem máxima pela minimização do vetor peso *w*. Além disso, $|b|/||w||$ é a distância perpendicular do hiperplano à origem e $||w||$ é a norma Euclidiana de *w*. Os pontos que estão em um dos hiperplanos H_1 e H_2 são chamados vetores de suporte. Esses pontos, indicados na Figura 2.2 por círculos extras, alteram a solução encontrada caso sejam removidos. Além disso, em fase de uso de SVM, apenas os vetores de suporte são necessários para que dados desconhecidos sejam classificados.

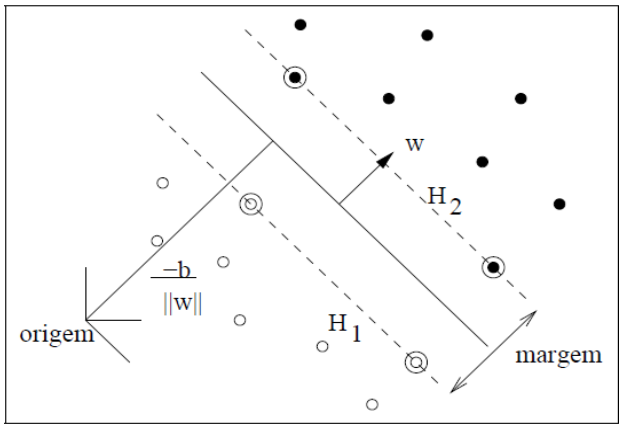

Figura 2.2. Hiperplano de separação ótima para um problema com duas classes.

O algoritmo original de SVM não encontra a solução desejada quando aplicado a dados não linearmente separáveis, característica presente na maioria dos problemas reais [19]. Entretanto, tais problemas podem ser solucionados por SVM por meio da utilização de funções de separação de dados mais complexas do que funções lineares. Sendo assim, o uso de diferentes funções *kernel* possibilita a construção de SVM com diferentes tipos de superfícies de decisão não-linear no espaço de entrada.

Com o uso de funções *kernel,* as instâncias são inicialmente mapeadas para um espaço de características de maior dimensão que o espaço de características original. Permitindo dessa forma, a classificação em espaços não linearmente separáveis. A Figura 2.3 mostra o processo de transformação de um domínio não linearmente separável, em um problema linearmente separável por meio do aumento da dimensão, consequência do mapeamento feito por uma função *kernel* $F(x)$.

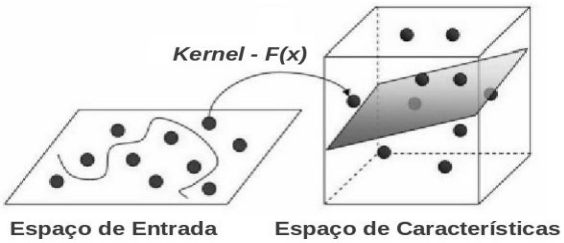

Figura 2.3. Mapeamento do espaço de entrada via função *kernel*.

Dentre as funções *kernel* mais usadas destacam-se: Polinômios, Funções de Base Radial (RBF) ou Gaussiana e Sigmóide, definindo diferentes máquinas de aprendizagem conforme Tabela 2.1. Nessa tabela, *x* representa os vetores de suporte, enquanto *y* representa os dados de teste.

Tabela 2.1. Funções de *Kernel* mais utilizadas com SVM

Tipo de *Kernel*	**Função *K* (x,x_i)**	**Tipo de Classificador**
Polinomial	$[(x*y)+1]^d$	Máquina de Aprendizagem Polinomial
Gaussiano ou (RBF)	$\exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)$	Rede RBF
Sigmoidal	$\tanh[\beta_0(x*y)]+\beta_1$	Perceptron de duas camadas

A principal vantagem de SVM é a baixa probabilidade de erros de generalização [20]. Quanto à utilização de SVM para deteção de intrusão, por exemplo, há duas vantagens principais. A primeira está relacionada à rapidez do algoritmo em fase de uso, uma vez que o desempenho em tempo real é de primordial importância para esse tipo de aplicação. A segunda razão é a escalabilidade, pois SVM é relativamente insensível ao número de pontos de dados. Dessa forma, a taxa de precisão na classificação não depende diretamente da dimensão do espaço de características [21]. Entretanto, SVM tem como desvantagem a demanda por elevada complexidade computacional na fase de treinamento, fato que pode inviabilizar o uso de SVM em problemas que necessitem de treinamento online.

2.1.2.2. K-means

Algoritmos de agrupamento são geralmente usados de forma não supervisionada. Ou seja, é apresentado um conjunto de instâncias de dados que devem ser agrupados de

acordo com alguma similaridade. O algoritmo tem como informação de entrada somente o conjunto de características que descrevem cada instância dada.

K-means [22] é um popular algoritmo de agrupamento comumente usado para particionar automaticamente um conjunto de dados em *k* grupos. Dado o conjunto de amostras $Z = \{z_1,...,z_N\}, z_j \in \Re^n$. O usuário deverá definir a quantidade de grupos *k* que *k-means* criará, sendo que para cada grupo haverá um centroide $\mu_1, \mu_2, \dots, \mu_k$. São selecionados inicialmente *k* centroides, os quais são iterativamente refinados pelo algoritmo da seguinte forma:

1. Inicialize os centroides $\mu_1, \mu_2, \dots, \mu_k$.
2. Agrupe cada uma das *N* amostras ao centroide mais próximo.
3. Calcule novamente o valor de cada centroide μ_i.
4. Repita os passos 2 e 3 até que não ocorra mais mudanças em μ_i.

As *k* primeiras amostras da base são normalmente escolhidas como os *k* centroides iniciais, enquanto que medidas de distância, como a distância Euclidiana, são usadas para o cálculo da similaridade entre as demais amostras e os centroides de cada grupo. O cálculo das distâncias é, portanto, a etapa que mais exige processamento. Se existem *N* pontos e *k* centroides calcula-se *N* x *k* distâncias neste passo. O centroide mais próximo de cada amostra vai 'incorporá-la', ou seja, a amostra vai pertencer ao grupo representado pelo mesmo. Após, os valores das coordenadas dos centroides são refinados, pois para cada grupo que possui mais de uma amostra, o novo valor do centroide é calculado por meio da média dos atributos de todas as amostras do grupo. Esse passo é repetido até a convergência, isto é, o algoritmo calcula a distância entre as amostras da base e os centroides iterativamente, até que mais nenhuma amostra mude de classe.

Chandola et al. [23] destacam que a utilização de técnicas baseadas em agrupamento proporciona algumas vantagens como a operacionalização de maneira não supervisionada; sendo frequentemente adaptada a tipos complexos de dados e; rapidez na fase de teste, desde que o número de grupos com que cada instância precisa ser comparada seja uma constante pequena. Porém, ainda segundo os autores, o uso destas técnicas acarreta desvantagens, como por exemplo, em relação ao desempenho, onde há uma enorme dependência da eficácia dos algoritmos em capturar a estrutura central das instâncias. Além disso, é difícil de ser calculada por não existir conhecimento sobre a saída desejada. Outra desvantagem se deve a complexidade computacional para agrupamento dos dados, sendo frequentemente um gargalo, especialmente para

aplicações online como os problemas de segurança em redes de computadores.

2.2. Deteção de Mudança

Ambientes dinâmicos são ambientes que estão em constante mudança, que evoluem com o tempo. O problema de deteção de intrusão, por exemplo, pode ser considerado dinâmico porque novos tipos de técnicas de invasão ou ataques, colaborativos ou não, podem alterar o ambiente regularmente.

Os filtros de mensagens eletrônicas, por exemplo, que têm como objetivo classificar as mensagens como *spam* ou não *spam*, dependem de aspectos que podem evoluir com o tempo, tais como: hábitos pessoais dos usuários e as estratégias utilizadas pelos atacantes para ludibriar tais filtros. Um sistema *antispam* pode ter seu desempenho de classificação prejudicado em função das mudanças provocadas por esses dois fatores.

Nesse contexto, uma maneira de observar a evolução do ataque, em janelas de tempo, é aplicar o conceito de deteção de mudanças. Esta seção apresenta inicialmente algumas terminologias necessárias para entender melhor a definição e as abordagens de deteção de mudanças. Também são apresentadas algumas definições de aprendizagem de máquina e seleção de características. Este último, é um ponto importante utilizado nesta tese para determinar uma mudança de padrão em problemas dinâmicos. Além disso, alguns trabalhos relacionados empregando técnicas de deteção de mudanças são descritos.

2.2.1. Terminologia

Para entender o processo de deteção de mudança é necessário conhecer alguns dos termos e conceitos comumente empregados nessa linha de pesquisa. A seguir, encontram-se definições de classes, características ou atributos e base de dados, extraídas de [24] e [25]:

- **Classes**: uma classe possui objetos similares, enquanto objetos de classes diferentes são dissimilares. Por exemplo, sites *web* falsificados representando um ataque do tipo *phishing web* podem ser categorizados como objetos da classe *phishing*, enquanto páginas *web* não falsificadas são objetos da classe *Non phishing*. Pode-se definir que um problema de classificação de dados possui ***C***

classes, rotuladas de r_1 a r_C, organizadas em um conjunto de rótulos $\Omega = \{r_1, \dots, r_C\}$ e que cada objeto pertence a apenas uma classe.

- **Características ou atributos**: Os objetos que compõem as classes são descritos por meio de características ou atributos. As características podem ser nominais, tais como palavras-chave em *phishing web* (*account, update, confirm*, etc.) ou podem ser numéricas, como tamanho ou quantidade de pontos na URL. Os valores de características de um objeto *x* são organizados em um vetor *n*-dimensional $x = [x_1, \dots, x_n] \in \mathbb{R}^n$. O espaço $\mathbb{R}^n$ é denominado espaço de características, sendo que cada eixo corresponde a uma característica do objeto.

- **Aprendizagem de Máquina (AM)**: aprendizagem de máquina supervisionada é formalmente definida em [26] com o objetivo de predizer uma variável alvo $y \in \Omega$ em uma tarefa de classificação, dada uma amostra de entrada na forma de vetor características $x = [x_1, \dots, x_n] \in \mathbb{R}^n$.

- **Base de Dados**: os algoritmos de aprendizagem de máquina precisam de informações provenientes dos dados disponíveis do problema. Essa informação normalmente está organizada na forma de um conjunto de dados $Z = \{z_1, \dots, z_N\}, \in \mathbb{R}^n$. Em problemas de aprendizagem supervisionada, o rótulo da classe de z_j é definido por $l(z_j) \in \Omega, j = 1,\dots,N$. A base de dados deve ser constituída por instâncias do problema, as quais devem ser representadas como vetores de características. Por exemplo, dadas diversas instâncias de URLs, não é uma tarefa trivial representar as URLs por meio de vetores de características como: tamanho de URL, quantidade de pontos na URL, palavras-chave, caracteres hexadecimais, etc. No entanto, definir um padrão de características relevantes que represente esse conjunto é decisivo no resultado do processo de classificação automática.

- **Instâncias**: dada uma base de dados $Z = \{z_1, z_2, \dots, z_N\}$, contendo vetores de características n-dimensionais que descrevem N instâncias, ou amostras de dados, uma instância, é, portanto, um vetor $z_j = \{x_1, x_2, \dots, x_n\}$ n-dimensional, que representa uma ocorrência do problema investigado.

- **Fonte (S)** [27]: seja $z_j \in \mathbb{R}^n$ uma instância em um espaço de características de dimensionalidade n. Conforme mencionado anteriormente, em problemas de

classificação, z_j possui uma classe r_i, onde $r_1, r_2, \ldots, r_C$ é o conjunto de rótulos de classes de um problema. O classificador ideal para atribuir a classe r_i à amostra z_j é completamente determinado pelas probabilidades a priori das classes $P(r_i)$ e pelas funções de densidade de probabilidade da classe condicional $\rho(z_j|r_i)$. Em qualquer momento, tem-se um ou mais dados ativos. Assim, um conjunto de probabilidades a priori das classes e as funções de densidade de probabilidade da classe condicional são definidas como conceito ou dados fonte conforme equação 1:

$$\mathbf{S} = \left\{\left(P(r_1), \rho(\mathrm{z_j}|r_1)\right), \left(P(r_2), \rho(\mathrm{z_j}|r_2)\right), \ldots, \left(P(r_C), \rho(z_j|\mathrm{k})\right)\right\} \quad (1)$$

- **Relevância de atributos ou características** [28]: sendo D_i, com, $1 \leq i \leq N$, um domínio de instâncias $Z = \{z_1, \ldots, z_N\}$. Considerando ρ uma distribuição de probabilidade sobre D e Ω um espaço de rótulos (classes), busca-se em AM modelar e identificar uma função objetivo $r: Z \rightarrow \Omega$ de acordo com as características relevantes de D. Um conjunto de dados E composto por instâncias $|E|$ pode ser visto como o resultado de uma amostragem de D sob a probabilidade total de vezes de $|E|$ e rotulando seus elementos usando r.

- **Seleção de características** [28]**:** sendo x um conjunto original de características, com cardinalidade $|x| = n$. O problema de seleção de características contínua se refere ao fato de assinalar o peso ω_i para cada característica $x_i \in x$ de tal maneira que a correspondente ordem da sua relevância seja preservada [28].

Além da definição dos principais termos utilizados na área de aprendizagem de máquina, a definição de mudança e a categorização dos principais tipos de mudanças encontrados na literatura são fundamentais para a compreensão desta tese. Essas definições são apresentadas na próxima seção.

2.2.2. Definição

Mudança de conceito, termo conhecido na literatura de AM como *concept drift*, pode ser definida como uma alteração nas classes definidas ao longo do tempo [29], ou de um modo geral, mudanças podem ser causadas por alterações nas probabilidades a *priori* das classes ou nas distribuições de probabilidade das classes [30]. Portanto, pode-se distinguir as seguintes fontes para *concept drift*:

- Mudança de probabilidade prévia para a classe $p(r)$ (probabilidade a *priori*);
- Mudança de distribuição de probabilidade de classe-condicional $p(x|r)$;
- Mudança que afeta a predição, a probabilidade a posteriori $p(r|x)$;

Formalmente, dado um problema de fluxo contínuo, o conhecimento do problema é representado por amostras de dados $z_1, z_2, \ldots, z_N$, em que cada exemplo é um vetor *n*-dimensional em um espaço pré-definido $\mathbb{R}^n$ de *n* características. Em cada instante de tempo t, os exemplos compõem um conjunto Z de **N** exemplos recentes, enquanto $\bar{Z}$ simboliza o conjunto contendo os $\bar{N}$ exemplos que representavam o problema no instante $t-1$. Em um problema dinâmico, os exemplos em Z não são gerados pela mesma distribuição utilizada em $\bar{Z}$ [31].

Ambientes podem apresentar diferentes tipos de mudanças. Logo, um sistema de classificação pode ser construído de acordo com o tipo de mudança que pode ocorrer no ambiente de aplicação do sistema. Por exemplo, estratégias baseadas em janelas de tempo que deslizam sobre os dados de treinamento podem ser usadas para detetar mudanças lentas. No caso de mudanças repentinas, classificadores estáticos equipados com um mecanismo de deteção de mudança podem ser empregados. A seguir, os principais tipos de mudanças são descritos.

2.2.3. Tipos de *Concept Drift*

Kuncheva [32] [33] destaca quatro tipos gerais de mudanças: (1) mudança abrupta (do inglês *sudden drift*), (2) mudança gradual (do inglês *gradual drift*), (3) mudança incremental (do inglês *incremental drift*) e (4) contexto recorrente (do inglês *reoccuring contexts*).

Os tipos de mudanças são apresentados com auxílio do padrão de configuração dos dados fonte sobre o tempo. Para melhor exemplificar os tipos de mudanças, sem perda de generalidade, são consideradas apenas duas fontes de dados em função do tempo: S_I *e* S_{II}, onde S_I representa a fonte de dados em um tempo t_0, enquanto S_{II} representa a fonte de dados no tempo t_1.

O tipo mais simples é a mudança abrupta, que ocorre quando a fonte S_I altera-se abruptamente, evoluindo em um tempo t_1 para uma fonte S_{II}. Em outras palavras, o conceito associado à fonte S_I é substituído pelo conceito associado à fonte S_{II}. Esse tipo

de *drift* ocasiona uma redução significativa na taxa de classificação correta dos sistemas, dado que os classificadores treinados com dados de uma fonte, não são robustos a dados de outras fontes.

Mudança gradual é outro tipo encontrado com frequência na literatura, mas existem dois termos misturados sobre esse tipo. O primeiro tipo de mudança gradual se refere ao período quando ambos, S_I e S_{II}, são ativos [34],[35]. Com o passar do tempo, a probabilidade de ocorrência de amostras da fonte S_I diminui, enquanto a probabilidade de ocorrência de amostras da fonte S_{II} aumenta. Portanto, no início desse processo de mudança gradual, uma amostra da fonte S_{II} facilmente pode ser confundida com ruído aleatório. No segundo tipo de mudança, também referenciada como gradual, são incluídas mais de duas fontes, mas a diferença entre as fontes é muito pequena e a mudança só é observada após um período de tempo mais longo. Por esse motivo, na literatura esse tipo de mudança também é chamado incremental.

Por fim, o último tipo de mudança é conhecido como contexto recorrente, e ocorre quando o conceito (fonte) ativo reaparece depois de algum tempo. Esse tipo de mudança difere de uma sazonalidade comum, pois não é previsivelmente periódica. Portanto, não é possível definir quando a fonte reaparece.

A Figura 2.4 apresenta uma visão da estrutura dos principais tipos de *drift,* assumindo somente uma característica, e somente dados de uma classe, sendo que uma fonte é caracterizada por uma cor. Pode-se observar a partir da disposição das cores apresentadas na Figura 2.4 a representação característica de cada tipo de mudança. Na mudança abrupta, o conceito representado pela cor azul muda repentinamente para o conceito representado pela cor verde. Enquanto na mudança gradual, a cor verde vai gradualmente se manifestando no conjunto de dados fonte. Na mudança incremental, existe uma mistura de cores até que a cor verde se sobreponha. Por fim, o contexto recorrente é claramente observado pela presença inicial da cor azul, a qual é repentinamente substituída pela cor verde, sendo que a cor azul reaparece novamente.

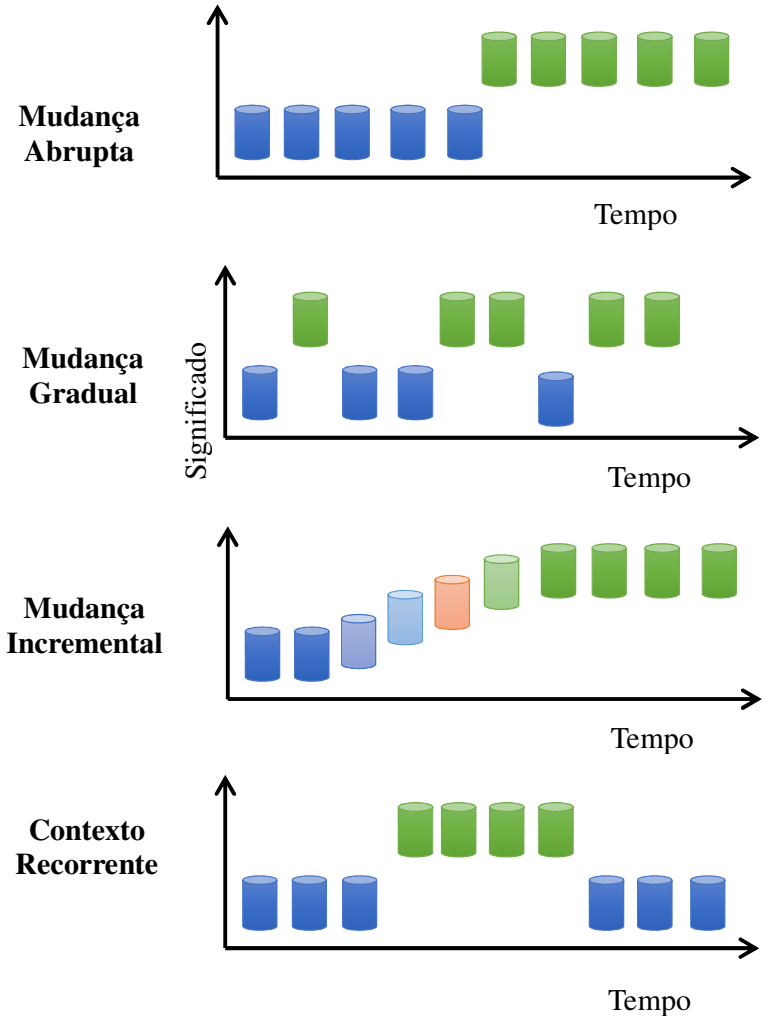

Figura 2.4. Ilustração dos Quatro Tipos de Estrutura de *Concept Drift* (adaptada de Zliobaite [27])

Existe atualmente na literatura uma tentativa de categorizar esses tipos de mudanças em categorias mutualmente exclusivas baseadas em recorrência, velocidade, gravidade e previsibilidade [36]. Porém, nesta tese se manterá a estrutura dos quatro tipos de *concept drift* apresentados nesta seção.

2.2.4. Métodos de Reação a *Concept Drift*

Existem três principais abordagens para reagir a mudanças apontadas na literatura atual: (1) seleção de instância (do inglês *instance selection*), (2) peso de instância (do inglês *instance weighting*) e (3) conjuntos de classificadores (do inglês *ensemble learning*) ou aprendizado com múltipla descrição de conceito [11]. Essas três abordagens utilizam estratégias diferentes para contornar uma mudança, conforme descrito nesta seção.

Seleção de instância é a abordagem mais comum para reagir a mudanças. Consiste na definição de uma janela de dados, que representa o conceito aprendido, a qual se move sobre as amostras mais recentes, ou seja, as amostras mais recentes classificadas erroneamente são adicionadas ao domínio de treino substituindo as amostras mais antigas. O conceito aprendido é utilizado para estimar um futuro próximo [34].

Outra abordagem para contornar mudanças baseia-se na atribuição de peso às instâncias. Nessa estratégia, a capacidade de alguns algoritmos de aprendizagem, como SVM, para processar o peso das instâncias [37] é utilizada para manter o conceito atualizado. Instâncias podem ter seus pesos atribuídos de acordo com sua idade e sua representatividade para o conceito corrente.

Por fim, na abordagem conjunto de classificadores, um grupo com diferentes classificadores é inicialmente gerado. Em seguida, as saídas dos membros do conjunto são combinadas ou alguns classificadores membros são selecionados para que um resultado final seja obtido. Geralmente, a função de combinação das saídas dos classificadores é voto majoritário ou voto ponderado [38]. Além de métodos incrementais de conjuntos de classificadores, é usada alguma condição dinâmica para remover, reativar ou gerar novos membros para o conjunto, os quais são usualmente construídos com os dados atuais do problema.

A precisão do classificador é utilizada na maioria das aplicações para detetar a mudança antes que uma das abordagens descritas nesta seção seja usada para contornar a mudança detectada. Por exemplo, em algumas aplicações da abordagem de seleção de instâncias, cada instância classificada erroneamente pelo classificador é armazenada para compor um novo conjunto de treino, formando assim, um novo conhecimento para o classificador [39]. Com relação à abordagem de pesos de instâncias, algumas aplicações utilizam a taxa de precisão do classificador para reconsiderar a idade das amostras e então, retirar do conjunto de treino as amostras mais antigas ou atribuir novos pesos às instâncias. Por fim, em abordagens do tipo conjunto de classificadores é verificado qual dos membros do conjunto apresenta a taxa de precisão com maior queda significativa ao longo do tempo, esse classificador é então removido e um novo membro com o conhecimento atualizado é incluído no conjunto.

Diferentemente da maioria das aplicações apresentadas na literatura, esta tese propõe uma estratégia de deteção de mudança baseada na evolução das características. Dado esse objetivo, é necessário entender como se constrói um conjunto de características

que representem o problema com ênfase nas características relevantes. O próximo capítulo apresentará com mais profundidade o conceito de relevância de características. Antes, porém, a próxima seção aborda métodos para deteção de mudança.

2.2.5. Método de Deteção de *Concept Drift*

Identificar mudanças em ambientes dinâmicos é um desafio, principalmente em problemas de deteção de atividades maliciosas tais como *spam*, *phishing*, *malware*, vírus, entre outros. Nesse tipo de problema ocorrem mutações de ataques muito rapidamente, as quais levam à alteração no padrão de ataque e ocasionam perdas na taxa de deteção, tornando o sistema fragilizado e desprotegido. Na tentativa de minimizar esses efeitos negativos, existem alguns métodos de deteção implícita ou explícita que Kucheva [32] apresenta e que podem ser baseados em pontos como:

- Distribuição de probabilidade;
- Relevância de características;
- Complexidade de modelos;
- Precisão do classificador.

O critério de distribuição de probabilidade se refere à distribuição da classe condicional para as classes que se distanciam dos seus valores iniciais. Baseado em quão bem a distribuição assumida adapta-se aos dados mais recentes, a mudança pode ser detectada e os dados antigos devem ser substituídos. Esses métodos de deteção de mudança comparam a probabilidade dos novos dados à distribuição inicial por meio de um limiar (*threshold*). Esse tipo de critério costuma ser usado em aplicações que processam grande quantidade de dados ou lotes de dados [40], [41].

O critério mais largamente usado é a precisão do classificador, tanto para deteção de mudança implícita quanto explícita. Nesse caso, a taxa de deteção do classificador é monitorada a fim de determinar quando uma mudança ocorreu [42][10][43]. Por exemplo, se a taxa de deteção tornar-se menor que um limiar pré-fixado, então a mudança é detectada e neste momento se usará alguma técnica para reagir a esta mudança.

O critério de relevância de características baseia-se na diferença de relevância de padrões de características usadas para discriminar as instâncias. Características ou mesmo combinações de características que foram relevantes no passado podem não ser suficientemente discriminativas no momento atual [44] [45] [46]. Ao manter-se o controle

sobre a melhor combinação de características é possível treinar um classificador atualizado para a mais recente distribuição de dados. Conforme mencionado na introdução, nesta tese se manterá o foco no critério de relevância de características para deteção de *concept drift*.

Técnicas de deteção de mudança podem ser agrupadas de maneiras distintas. A seguir são destacados alguns desses agrupamentos.

- **Deteção de mudança passiva ou implícita** – nesta abordagem o sistema não se preocupa em perceber as mudanças, os mecanismos de deteção estão implícitos ao método de reação. À medida que as informações surgem, o sistema se alimenta (atualiza) a cada nova informação [47].
- **Deteção de mudança Ativa ou Explícita** – nesta abordagem o sistema se preocupa em perceber a mudança, ou seja, os mecanismos de deteção estão explícitos. O sistema após perceber a alteração adapta-se à nova situação[48].

2.3. Transferência de Aprendizado

Pesquisas em transferência de aprendizado têm sido realizadas desde 1995 com diferentes nomes, tais como: aprendendo a aprender (do inglês *Learning to Learn*), transferência de conhecimento, transferência indutiva, aprendizado multitarefa, consolidação de conhecimento, meta aprendizado e aprendizado incremental [49]. Apesar dessa diversidade de nomes, nesta tese será utilizado apenas o termo transferência de aprendizado.

A maioria dos algoritmos tradicionais de aprendizagem de máquina, faz predição de dados futuros usando modelos estatísticos, assumindo que a distribuição dos dados futuros permanece igual à distribuição dos dados usados para treinar o algoritmo. São, portanto, modelos estáticos. Quando a distribuição dos dados muda, o modelo estático precisa ser reconstruído a partir de uma nova coleção de treino. Em contraste, transferência de aprendizado permite que domínios, tarefas e distribuição de dados usados em treino e teste sejam diferentes [50]. Dessa forma, busca-se reduzir a necessidade de geração de uma nova base de conhecimento sempre que alguma alteração na distribuição das informações é detectada.

Esta seção apresenta uma terminologia que facilitará o entendimento do conceito de transferência de aprendizado. Em seguida, são apresentadas definições, abordagens

para transferência de aprendizado e algumas aplicações envolvendo esse conceito. Por fim, o capítulo encerra com uma breve discussão.

2.3.1. Terminologia

Para que uma definição mais precisa de transferência de aprendizado seja apresentada, os seguintes termos precisam ser descritos [50]:

- **Domínio**: um domínio ***D*** é composto por dois componentes: um espaço de características X e uma distribuição da probabilidade marginal P(X). Por exemplo, se a tarefa de aprendizado é a classificação de um ataque, e cada termo representa uma característica binária, então X é o espaço de todos os termos do vetor e x_i é um exemplo particular de aprendizado. Em geral, conforme descrito na seção 2.1.1., se dois domínios são diferentes, então eles possivelmente têm espaço de características diferentes ou distribuição de probabilidade marginal diferente.
- **Tarefa**: dado um domínio $D = \{X, P(X)\}$, uma tarefa é composta por dois componentes: um conjunto de classes Ω e uma função objetivo desconhecida $f(.)$, que é denotada por $T = \{\Omega, f(.)\}$. A tarefa pode ser aprendida com base nos dados de treino. Após o aprendizado, a função $f(.)$ pode ser usada para predizer o rótulo de uma nova instância x. Por exemplo, em uma tarefa de classificação de ataque, Ω é o conjunto de todos os rótulos, isto é "*spam*" ou "não *spam*".
- **Domínio Fonte**: define-se domínio fonte de dados como $D_f = \{(x_{f_1}, y_{f_1}), \ldots, (x_{fN}, y_{fN})\}$, onde $x_{fi} \in X_f$ é a instância de dados e $y_{fi} \in \Omega_f$ corresponde ao rótulo da classe.
- **Domínio Alvo**: define-se domínio alvo de dados como $D_a = \{(x_{a_1}, y_{a_1}), \ldots, (x_{aN}, y_{aN})\}$, onde a entrada x_{ai} esta em X_a e $y_{ai} \in \Omega_a$, é a saída correspondente.

2.3.2. Definição

A partir das terminologias apresentadas acima, pode-se definir transferência de

aprendizado da seguinte forma: dado um domínio fonte D_f e uma tarefa de aprendizado fonte T_f, um domínio alvo D_a e uma tarefa alvo T_a, transferência de aprendizado tem como objetivo melhorar o aprendizado da função de predição $f_a(.)$ em D_a, usando o conhecimento proveniente de D_f e de T_f, sendo $D_f \neq D_a$, ou $T_f \neq T_a$ [50].

Como um domínio é um par $D = \{x, P(X)\}$, então a condição $D_f \neq D_a$ implica que $x_a \neq x_a$ ou $P_f(X) \neq P_a(X)$. Por exemplo, em uma classificação de ataques do tipo *spam,* isso significa que as características das amostras do domínio fonte são diferentes das características das amostras do domínio alvo ou a distribuição marginal dos dois domínios é diferente.

Da mesma forma, a tarefa é definida como um par $T = \{\Omega, P(y|X)\}$. Então, a $T_f \neq T_a$ implica que $\Omega_f \neq \Omega_a$ ou $P(Y_f|X_f) \neq P(Y_a|X_a)$. Quando os domínios fonte e alvo são os mesmos, $D_S = D_T$, e suas tarefas de aprendizado também são as mesmas, $T_f = T_a$, o problema do aprendizado torna-se um problema de aprendizado de máquina tradicional.

2.3.3. Arranjo para Transferência de Aprendizado

Existem três questões que orientam as estratégias de transferência de aprendizado: (1) o que, (2) como e (3) quando transferir [51].

A questão "o que transferir" procura identificar a parte do conhecimento que pode ser transferida dos domínios ou tarefas. O conhecimento pode ser dividido em conhecimento individual e conhecimento coletivo. Na primeira categoria estão as partes do conhecimento que são específicas de domínios ou tarefas individuais, enquanto na segunda categoria estão partes que podem ser comuns entre diferentes domínios. Essa segunda categoria é mais interessante para melhorar o desempenho do sistema no domínio ou tarefa alvo. Após descobrir o conhecimento que deve ser transferido, o algoritmo de aprendizagem precisa transferir o conhecimento, fase que levanta a questão "como transferir".

A questão "quando transferir" refere-se à identificação da situação ideal para a transferência ser realizada, assim como também é interessante saber em qual situação o conhecimento não deve ser transferido. Em algumas situações, quando o domínio fonte e o domínio alvo não se relacionam entre si, a transferência não é bem sucedida. No pior caso, essa transferência pode prejudicar o desempenho do aprendizado no domínio alvo,

situação que é frequentemente referenciada como “transferência negativa” [50]. A maioria dos trabalhos correntes foca em transferência de aprendizado sobre “o que transferir” e “como transferir”, por implicitamente assumir que o domínio fonte e o domínio alvo são relacionados entre si [50]. Nesta tese, essa hipótese também é assumida.

Encontram-se na literatura três categorias de transferência de aprendizado: (1) transferência de aprendizado indutivo, (2) transferência de aprendizado transdutivo e (3) transferência de aprendizado não supervisionada [51]. Essa classificação é baseada nas diferentes situações que podem ocorrer entre os domínios fonte e alvo e as tarefas fonte e alvo. A Figura 2.5 ilustra essa categorização, conforme descrito abaixo:

- No cenário de transferência de aprendizado indutivo, a tarefa alvo é diferente da tarefa fonte. Nesse caso, não importa se os domínios fonte e alvo são iguais ou não.
- No cenário de transferência de aprendizado transdutivo, as tarefas fonte e alvo são as mesmas, enquanto os domínios fonte e alvos são diferentes.
- No cenário de transferência de aprendizado não supervisionado, como em aprendizado indutivo, a tarefa alvo é diferente da tarefa fonte, mas as tarefas são relacionadas entre si.

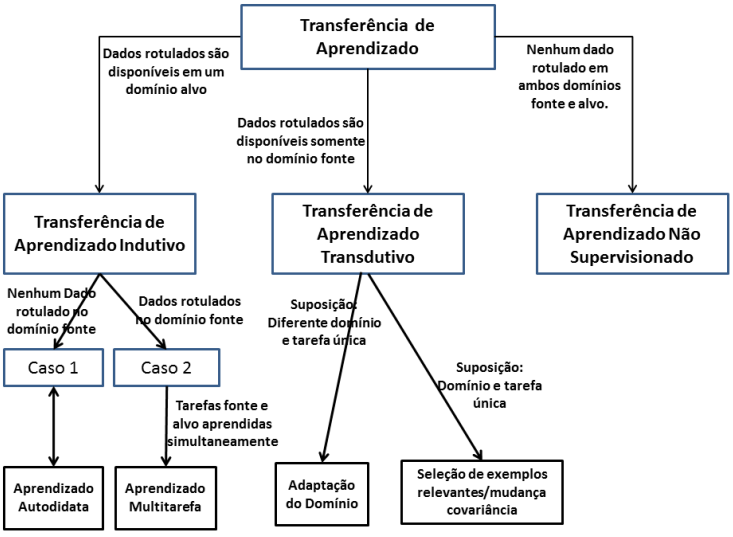

Figura 2.5. Uma Visão das Diferentes Configurações de Transferência de Aprendizado, adaptado de Pan [50].

As estratégias para transferência de aprendizado podem ser resumidas em quatro casos, com ênfase na questão “o que transferir”: (1) abordagem de transferência de aprendizado baseada em instância, (2) abordagem de transferência de representação de características, (3) abordagem de transferência de parâmetro e (4) abordagem de

transferência de conhecimento relacional. A Tabela 2.2 mostra essas quatro abordagens com uma breve descrição.

Tabela 2.2. Diferentes Abordagens para Transferência de Aprendizado [50].

Abordagens para transferência de aprendizado	Finalidade
Transferência de Instância	Rever a relevância de alguns dados rotulados no domínio fonte para uso no domínio alvo.
Transferência de Representação de Características	Encontrar uma boa representação de características que reduza a diferença entre os domínios fonte e alvo e o erro do modelo de classificação.
Transferência de Parâmetro	Descobrir parâmetros compartilhados ou prévios entre os modelos do domínio fonte e alvo, o qual pode auxiliar a transferência de aprendizado.
Transferência de Conhecimento Relacional	Constrói mapeamento de conhecimento relacional entre os domínios fonte e alvo. Ambos os domínios são domínios relacionais.

A primeira abordagem apresentada na Tabela 2.2 assume que certas partes dos dados do domínio fonte podem ser usadas para o aprendizado do domínio alvo. Para tanto, utiliza-se o monitoramento do peso de instâncias (do inglês *reweighting)*. Rever o peso de instâncias e de amostras relevantes são duas das principais técnicas nesse contexto.

Na abordagem de representação de características, o conhecimento usado para transferência entre os domínios é codificado na representação das características aprendidas. Com a nova representação de características, espera-se melhorar o desempenho na tarefa alvo significativamente.

Na abordagem de transferência de parâmetro, a transferência do conhecimento é baseada no compartilhamento de parâmetros ou prioridades. O conhecimento pode ser transferido por meio das tarefas. Por fim, na abordagem de transferência de conhecimento relacional, o conhecimento a ser transferido é o relacionamento entre os dados.

A Tabela 2.3 apresenta a relação entre as abordagens propostas para transferência de aprendizado e as configurações de transferência de aprendizado. A abordagem de transferência de instâncias atende às configurações de transferência de aprendizado indutivo e transdutivo, uma vez que os domínios podem ser iguais ou diferentes. A abordagem de maior flexibilidade é a abordagem de transferência de representação de características, que atende às três configurações. Já as duas últimas abordagens de

transferência somente atendem à configuração de transferência de aprendizado indutivo.

Tabela 2.3. Diferentes Abordagens usadas em Diferentes Configurações para Transferência de Aprendizado [50]

	Transferência de Aprendizado Indutivo	Transferência de Aprendizado Transdutivo	Transferência de Aprendizado não Supervisionado
Transferência Instâncias	X	X	
Transferência de Representação de Características	X	X	X
Transferência de Parâmetros	X		
Transferência Conhecimento Relacional	X		

A partir das relações entre abordagens de transferência de aprendizado e os conjuntos de configurações, com base na questão "o que transferir", esta tese foca na transferência de representação de características para a configuração de transferência de aprendizado transdutivo. Esta escolha está diretamente relacionada ao objetivo geral desta tese que é: desenvolver uma arquitetura para detetar a evolução de características de ataques de maneira dinâmica, presando pela estabilidade da taxa de precisão e uma baixa taxa de falso alarme, além de diminuir ou anular a interferência humana.

Para tanto, a próxima seção apresenta transferência de aprendizado transdutivo de uma forma um pouco mais aprofundada, com ênfase na representação de transferência de características.

2.3.4. Transferência de Aprendizado Transdutivo

A transferência de aprendizado transdutivo foi proposta incialmente por Arnold et al. [52]. Conforme definido na seção anterior, nesse tipo de transferência de aprendizado é requerido que as tarefas fonte e alvo sejam as mesmas, enquanto que os domínios fonte e alvo são diferentes.

Portanto, pode-se definir transferência de aprendizado transdutivo como segue: dado um domínio fonte D_f e uma tarefa de aprendizado correspondente T_f, um domínio alvo D_a e uma tarefa de aprendizado correspondente T_a, a transferência de aprendizado

transdutivo objetiva melhorar o aprendizado da função alvo prevista f_a (.) no domínio alvo D_a usando o conhecimento do domínio fonte D_f e tarefa fonte T_f, onde $D_f \neq D_a$ e $T_f = T_a$.

A literatura apresenta algumas abordagens para transferência de aprendizado transdutivo e a maioria das aplicações utiliza a abordagem de transferência de instância [53][54]. Essas aplicações têm como motivação a importância das instâncias. No entanto, esta tese se concentrará na abordagem de transferência de aprendizado baseada em representação de características. Esta abordagem tem como motivação a seleção do conjunto de características mais relevantes para uma representação melhor do problema.

2.3.4.1. Transferência de Representação de Características

A abordagem de transferência de representação de características tem como objetivo principal encontrar a melhor representação de características para minimizar a diferença entre o domínio fonte e o domínio alvo e consequentemente, manter o modelo de classificação sempre atualizado. Entretanto, encontrar bons subconjuntos de representação de características implica no tipo de domínio de dados que se possui. Por exemplo, se os dados do domínio fonte são rotulados, pode-se usar métodos de aprendizagem supervisionada para construir a representação de características. Por outro lado, se dados do domínio fonte não são rotulados, usa-se métodos de aprendizado não supervisionado. Esta tese utiliza conjuntos de dados rotulados, proporcionando que se use métodos de aprendizado supervisionado.

Capítulo 3 Trabalhos Relacionados

Este Capítulo apresenta uma análise sobre os principais trabalhos que empregam os conceitos apresentados no Capítulo 2. Os trabalhos são organizados didaticamente pelo tipo de conceito empregado em suas aplicações, e ao final de cada seção, é apresentado um quadro com um resumo das aplicações que se assemelham à proposta desta tese.

3.1. Concept Drift

Com foco em problemas dinâmicos, as aplicações apresentadas nesta seção utilizam alguma técnica de deteção de mudança para tratar o problema de mudança constante. Diferentemente dos algoritmos de aprendizagem de máquina tradicionais, abordagens de deteção de mudança se preocupam em manter um controle dinâmico, procurando perceber mudanças entre a base de conhecimento e amostras desconhecidas.

Pesquisas com foco neste contexto têm sido realizadas há aproximadamente duas décadas. Duas soluções clássicas na área de pesquisa sobre *concept drift* são: Schlimmer e Granger [55] - denominado Stangger, e Widmer e Kubat [34] - denominado FLORA.

O *framework* FLORA trabalha profundamente a questão de janelas de tempo. Usando janelas deslizantes no aprendizado de um classificador binário, isto é, amostras mais recentes são contempladas para compor o novo aprendizado, enquanto amostras mais antigas são excluídas, conforme ilustrado na Figura **3.1**. O *framework* utiliza dinamicamente três regras: uma regra cobre somente exemplos positivos, por exemplo, em casos de filtros de *spam* as amostras positivas representam os e-mails *spam*; outra somente exemplos negativos (e-mails legítimos ou não *spam*); e a última é uma combinação considerando amostras positivas e negativas. FLORA foi melhorado com alguns aprimoramentos, dando origem ao FLORA2, FLORA3 e FLORA4. O FLORA2 é capaz de adaptar o tamanho da janela. FLORA3 melhora a capacidade de adaptação e a capacidade de reativar regras obsoletas. Finalmente, FLORA4 inclui melhores estratégias para o tratamento de ruído. A abordagem usada pelo *framework* FLORA para contornar o problema de *concept drift* está no emprego de janelas deslizantes que consideram sempre, as amostras mais recentes para o aprendizado do modelo de classificação.

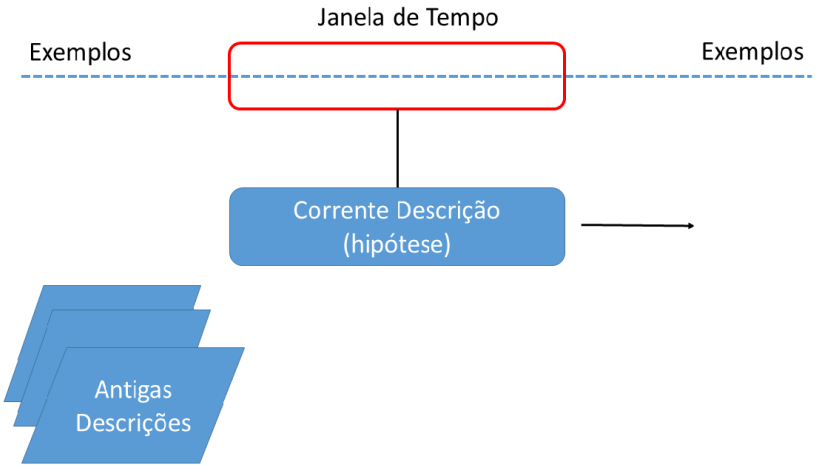

Figura 3.1 Janela de tempo deslizante sobre exemplos apresentando descrições correntes e antigas [34].

O sistema STAGGER [55] busca descrever os dados pelo conjunto mais simples de características Cada uma dessas caracterizações que descrevem as amostras tem um par inicial de pesos. O sistema envolve um processo de projeção que combina essa representação do conceito distribuído em relação às amostras novas. Para cada caracterização, um dos pesos é usado, caso a caracterização seja similar à amostra, enquanto o outro peso é usado caso não seja similar à amostra. Os pesos são ajustados pelo processo de avaliação, o qual mantém o controle do número de vezes que cada caracterização foi similar ou não às amostras. A avaliação é o processo de determinar a eficácia das descrições dos conceito internos.

No STAGGER, cada caracterização na descrição do conceito é continuamente avaliada, adaptando os seus pesos. Essa avaliação reflete descobertas na aprendizagem, baseadas no número de vezes que cada caracterização foi bem sucedida e falhou com uma predição. A partir das combinações de pesos, é identificado o *concpet drift* e contornada a mudança para o novo aprendizado. A Figura 3.2 apresenta o processo de refinamento que busca modificar o aprendizado e melhorar sua medida de desempenho pela evolução. Os elementos da descrição do conceito distribuído são modificados por especialização, generalização e inversão. Cada possível caracterização booleana de valores de atributo pode ser vista como um nó no espaço de todas as funções. A Figura 3.2 representa uma pequena porção deste espaço sobre um domínio simples (cada elipse representa uma função booleana). Qualquer das duas possíveis funções booleanas são parcialmente ordenadas ao longo de uma dimensão de generalidade. Nós mais elevados na Figura 3.2 são mais gerais, enquanto os nós mais específicos estão abaixo.

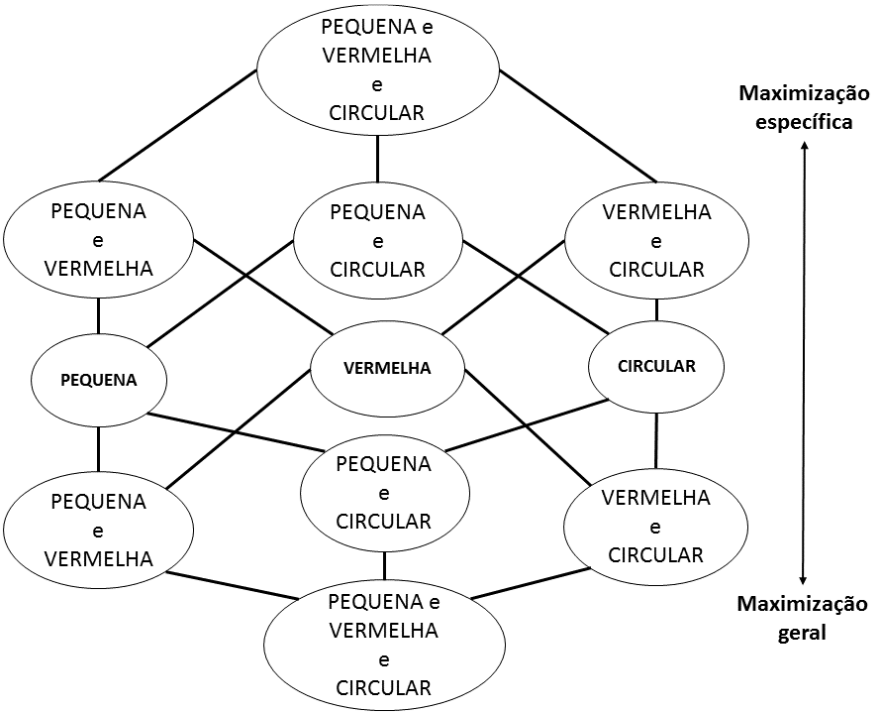

Figura 3.2 Parcial caracterização [55]

A seguir, outras aplicações mais recentes são apresentadas com o objetivo de identificar as estratégias ou métodos empregados para detetar e contornar a mudança dinâmica.

Lee et al. [56] propõem um modelo de deteção de *spam* baseado em Florestas Aleatórias (do inglês *Randon Forests* - RF), o qual permite o uso de seleção de características e otimização de parâmetros. A RF constrói um conjunto de árvores do tipo CART (*Classification And Regression Tree*) usando mecanismo de seleção aleatória de amostras e de características. Por usar esse mecanismo, cada nó da árvore seleciona um pequeno subconjunto de características, fator que permite a rápida criação de um classificador mesmo que os dados possuam alta dimensionalidade. A otimização de parâmetros permite ajustar os valores da quantidade de variáveis no subconjunto aleatório de cada nó e a quantidade de árvores na floresta. Enquanto a seleção de características tem como objetivo descobrir as características relevantes dentre todas as características existentes para diminuir o tempo de processamento e melhorar a taxa de deteção. Para tanto, os autores baseiam sua metodologia de aplicação em uma taxa de deteção que não pode ser menor que 95%.

Hayat et al. [14] propõem um sistema de filtro de *spam* baseado em modelo de

linguagem. Como o conteúdo do e-mail muda ao longo do tempo, então o modelo treinado torna-se obsoleto e incapaz de detetar novos *spams*. A abordagem proposta usa técnica de modelo de linguagem para representar a distribuição dos blocos de conteúdo. Um modelo de linguagem assinala uma probabilidade para cada palavra no documento. Cada palavra em cada bloco terá um peso. Esse peso indica a importância da palavra no bloco. É importante observar que as probabilidades são baseadas no conteúdo do bloco, portanto, como uma mudança qualquer afeta a distribuição dos dados, essa situação pode ser considerada uma mudança no conteúdo por meio do modelo de linguagem empregado. Porém a maneira que os autores aferem a deteção de mudança é a partir da comparação da taxa de precisão do modelo de classificação com um *theshold,* e geram um novo modelo de classificação.

Delany et al. [12] apresentam ECUE[3] (*Email Classification Using Examples – ECUE*) baseado em CBR (*Case-Based Reasoning*) como solução para o problema de filtro de *spam*. CBR é considerada uma técnica de aprendizado de máquina lenta (do inglês *lazy learning*). Os exemplos de dados de treino representam os casos da base de casos de um aprendiz de CBR. Em ECUE, cada e-mail é um caso representado como um vetor de características. A abordagem para deteção de mudanças que os autores utilizam é a seleção de instâncias, ou seja, a base de casos precisa ser atualizada a fim de incluir novos tipos de *spam* e de e-mail legítimos ao longo do tempo. Dado um volume de e-mails que um único usuário recebe por semana, existe uma necessidade de gerenciar ativamente os dados de treino. Logo, os e-mails classificados incorretamente como *spam* ou legítimo são inseridos ao treino do modelo como forma de atualizar a base de conhecimento.

Wenerstrom e Giraud-carrier [57] apresentam um conjunto de classificadores para deteção de mudanças que utiliza conjuntos de características adaptativas (*Feature Adaptive Ensemble* - FAE). FAE usa classificadores Naïve Bayes de forma incremental, isto é, os classificadores se adaptam a cada nova inclusão de uma instância de treino. Além disso, cada classificador em FAE é especializado em um subconjunto de características e novas características podem ser adicionadas a qualquer tempo, facilitando a tarefa de deteção de mudança contextual. FAE é capaz de adaptar-se bem a mudanças, pois utiliza diversos classificadores de diferentes idades e toma decisões sobre a remoção de um classificador do conjunto com base na precisão individual de cada

[3] http://www.comp.dit.ie/sjdelany/Dataset.htm

classificador. Conforme o modelo de classificação reduz o desempenho da sua tarefa de classificação baseada na taxa de precisão, um novo classificador é gerado para compor o conjunto de classificadores.

Tsymbal et al. [58] usam uma técnica de integração de conjunto de classificadores para melhorar o manuseio de deteção de mudança em um nível de instância. Em uma integração dinâmica, cada classificador base recebe um peso proporcional à sua precisão local na vizinhança da instância teste corrente, em vez de usar a precisão global de classificação como em voto de peso normal. A maioria das abordagens de conjuntos usa alguns critérios para dinamicamente apagar, reativar ou criar novos membros do conjunto. Esses critérios são normalmente baseados na consistência do modelo em relação aos dados correntes. Além disso, aprendizado baseado em instância é usado a fim de predizer o desempenho local do modelo base em um conjunto de dados. Para tanto, Tsymbal et al. [58] usam três técnicas dinâmicas baseadas na mesma estimativa de desempenho local: seleção dinâmica, voto dinâmico e voto dinâmico com seleção. Contudo, os autores consideram a precisão local dos modelos de classificação que compõem o conjunto de classificadores para determinar o momento de gerar um novo modelo de classificação.

Gu-Hsin et al. [59] apresentam um *framework* colaborativo para geração de regras de spam para troca e gerenciamento. O *framework* é construído com base em algoritmo genético e aprendizado por reforço empregando as regras. Se uma regra é raramente usada, sua força diminui. Classificações errôneas também levam à redução do uso da regra. Se as regras têm alta empregabilidade, então estas regras são muitas vezes usadas e apresentam um melhor desempenho. Se as regras têm baixa empregabilidade, estas regras ficam fora do escopo ou imprecisas. Os servidores de e-mail apenas mantém ou permitem regras alta empregabilidade para melhorar a sua deteção e desempenho. As mudanças são contornadas pelo emprego das regras, ou seja, as regras mudam conforme muda a empregabilidade devido ao comportamento dos ataques spam. Porém, essa mudança é definida pelo desempenho do classificador, ou seja, as regras recebem incremento ou decremento em seus pesos de acordo com a melhora ou redução do desempenho do modelo de classificação.

Pérez-Díaz et al.[60] apresentam alternativas para a aplicação da teoria dos conjuntos aproximados (do inglês *rough set theory*) para filtro de spam empregando algoritmos de classificação SVM e Navy Bayes. Para contornar o problema de concept drift é utilizada a geração dos conjuntos de regras como forma mais eficaz de acordo com

o desempenho do modelo de classificação. Para tanto, os conjuntos de regras são gerados a partir dos dados para definir a redução ou o aumento na desempenho do modelo de classificação. Além das regras serem geradas a partir dos dados o conjunto de regras, estas devem ser recalculadas para incorporar o conhecimento adicionado nas últimas mensagens. Desta forma, os autores se utilizam do desempenho do classificador para seleciona e atualizar as regras e o conhecimento construído.

Idre Zliobataite [61] apresenta uma aplicação com seleção de amostras relevantes para o treino baseada na similaridade da observação das amostras alvo. Similaridades considerando espaço e tempo são combinadas. O algoritmo determina o tamanho de um conjunto de treino ótimo baseado no desempenho do modelo de classificação. O autor usa o classificador K-NN (*Nearest Neighbors*) para tarefa de classificação. O *concept drift* é identificado a cada amostra com a projeção da similaridade de espaço e tempo para ser incluída na base de treino como novo conhecimento. As amostras incluídas são todas que apresentam similaridade com as amostras de treino. O autor trabalha com dois algoritmos onde um emprega janela de tempo fixa e o outro algoritmo janela de tempo variável. Novas amostras, com similaridade mais próxima do conjunto de dados de treino, são adicionadas de acordo com o desempenho do modelo de classificação.

Angel [62] apresenta um *framework* de deteção de mudanças empregando um meta-modelo (*Meta-Model Drift Detector*). Este meta-modelo é baseado em aprendizado recorrente. O sistema proposto é capaz de lidar com o contexto subjacente que resulta das mudanças detectadas em todo o processo de aprendizagem. O meta-modelo é treinado em paralelo com o processo de aprendizagem. Quando é detectado um desvio, o meta-modelo confere se o contexto da informação é igual a qualquer uma das informações anteriormente gerenciadas pelo processo de aprendizagem, fornecendo o modelo mais adequado armazenado para lidar com o conceito recorrente. Para tanto, o meta-modelo observa o espaço de características que gerou o modelo base em relação ao espaço de características da nova amostra que surge. A mudança é detectada quando a distribuição de probabilidade entre os dois espaços é diferente.

Lu et al. [63] apresentam um novo método para detetar mudança em um sistema de raciocínio baseado em casos (do inglês *case-based reasoning*). Em vez de medir a distribuição do caso atual, é introduzido um novo modelo de confiabilidade que detecta diferenças através de mudanças nesta confiabilidade. Para tanto, o método não exige conhecimento prévio da distribuição dos casos e fornece garantias estatísticas sobre a

confiabilidade das alterações detectadas, assim como descrições significativas e quantitativas destas mudanças. Em vez de observar o espaço de características, são observadas as medidas de confiabilidade para detetar mudança. A atualização do modelo de classificação é efetivada com a deteção de casos que se distancia dos modelos de confiabilidade.

Diante do contexto exibido pelas aplicações apresentadas anteriormente, a próxima seção apresenta uma síntese e um quadro que resume as abordagens de deteção de mudança e as estratégias empregadas para detecção a mudança.

3.1.2. Síntese

Manusear dados cuja natureza muda ao longo do tempo é uma das principais dificuldades na aprendizagem de máquina. Para recuperar ou aprender tais dados, é preciso estratégias para as seguintes tarefas: (1) detetar quando a mudança ocorre; (2) decidir quais exemplos manter e quais descartar e ainda; (3) revisar o modelo corrente quando mudanças significativas são detectadas.

A maioria das aplicações empregam a abordagem de seleção de instâncias [12] com a chegada de novos dados para contornar a mudança [13], [15], [58] – [63]. Porém, essa estratégia pode ser muito cara, pois o montante de dados que chega pode ser muito crítico, e para algumas aplicações, tais como categorização de *spam*, o *feedback* do usuário é necessário para rotular dados, ação que requer tempo e outros recursos. Uma maneira de superar essas questões é detetar mudanças e adaptar o modelo somente se necessário, o que não tem ocorrido na maioria das aplicações disponíveis na literatura, conforme apresentado na seção anterior. Dentre as soluções que buscam primeiro detetar a mudança para então adaptar o classificador, a maioria detecta mudanças a partir da queda da taxa de precisão [35], [57] – [63], ou seja, constatada a queda na taxa de precisão, então alguma abordagem para manusear a mudança é empregada (seleção de instância, peso de instância ou conjunto de classificadores). Portanto, nesse caso é indispensável detetar as mudanças antes de tomar uma decisão sobre a atualização do modelo.

No entanto, esse tipo de estratégia apesenta um sério problema, pois depende de uma queda significativa na taxa de precisão para identificar a ocorrência de uma mudança. Dessa forma, em problemas deteção de *spam*, *phishing* ou *malware,* por exemplo, quando detectada a queda na taxa de precisão do classificador, o sistema já sofreu algum dano ou prejuízo. Apesar dessa desvantagem, diversos métodos de deteção de mudança têm sido

apresentados na literatura [10], [42], [64] para monitorar erros de classificadores online. A maioria dos métodos baseia-se na taxa de erro, *recall*, e taxa de precisão do classificador para medir deteção de mudança.

A Tabela 3.1 apresenta uma comparação entre as diversas aplicações apresentadas nesta seção com o objetivo de salientar os métodos de deteção e reação empregados na presença de *concept drift*. Todas as aplicações utilizam como método de deteção a precisão do modelo de classificação e como método de reação a abordagem de seleção de instâncias considerando algum critério para selecionar tais instâncias. Algumas utilizam o critério de idade, ou seja, as instâncias mais novas são selecionadas para compor o novo conhecimento e as mais antigas retiradas. Outras utilizam critérios de similaridades entre as instâncias. As instâncias mais similares às amostras do domínio de treino são adicionadas para reforçar o aprendizado. Considerando o critério de peso atribuído às características, são selecionadas as instâncias que contemplam tais características.

Com uma proposta diferenciada, esta tese busca detetar a mudança com base na relevância de características. Será proposto um método de seleção de características original. Ampliando uma deteção ativa e contornando a mudança com transferência de aprendizado por meio da transferência de instâncias baseadas na evolução das características. A deteção da mudança objetiva empregar um limiar que se anteceda à degradação do modelo de classificação para minimizar os danos e prejuízos que são causados ao sistema quando se considera como medida de deteção de mudança o desempenho do modelo de classificação.

A Tabela 3.1 apresenta a sumarização das aplicações citadas ao longo desta seção com o propósito de apontar objetivamente como cada aplicação detecta e reage na presença de *conecpt drift*.

Tabela 3.1. Comparação dos Trabalhos Relacionados considerando os Métodos de Deteção e Reação à Deteção de Mudança (*Concept Drift*)

Autores	Método de deteção	Método de Reação
Widmer e Kubat [34]	Precisão do classificador	Usando janelas deslizantes no aprendizado do classificador, amostras mais recentes são contempladas para compor o novo aprendizado e amostras mais antigas são excluídas. Empregam abordagem de **seleção de instâncias** para contornar a mudança.
Schlimmer e Granger [55]	Precisão do classificador	As amostras são descritas por um conjunto de características a partir da atribuição de pesos. Os pesos são ajustados pelo processo de avaliação, que mantém o controle do número de vezes que cada caracterização é combinada ou não com as amostras. A partir das combinações de pesos, é identificado

Autores	Método de deteção	Método de Reação
		o concpet drift e contornada a mudança para o novo aprendizado com **seleção de instâncias**.
Lee et al. [56]	Precisão do classificador	Baseado em Florestas Aleatórias, permite o uso de seleção de características e otimização de parâmetros. A otimização de parâmetros é feita a partir de cada nó e da quantidade de árvores na floresta. Enquanto a seleção de características escolhe as características mais relevantes (**seleção de instâncias**).
Wenerstrom e Giraud-carrier [57]	Precisão do classificador	Apresentam um conjunto de classificadores para deteção de mudanças que utiliza conjuntos de características adaptativas (FAE). FAE é capaz de adaptar-se bem a mudanças, pois utiliza diversos classificadores de diferentes idades (**conjunto de classificadores**) e toma decisões sobre a remoção de um classificador do conjunto com base na precisão individual de cada classificador.
Tsymbal et al. [58]	Precisão do classificador	Usam uma técnica de integração de **conjunto de classificadores** para melhorar o manuseio de deteção de mudança em um nível de instância. Com uma integração dinâmica, cada classificador base recebe um peso proporcional à sua precisão local na vizinhança da instância teste corrente, em vez de usar a precisão global de classificação como em voto de peso normal.
Gu-Hsin et al. [59]	Precisão do classificador	Baseado em um *framework* colaborativo que gera regras de *spam* para troca e gerenciamento. As mudanças são contornadas pelo emprego das regras, ou seja, as regras mudam conforme muda a empregabilidade devido ao comportamento dos ataques spam (**seleção de instâncias**).
Pérez-Díaz et al.[60]	Precisão do classificador	Aplicam a teoria dos conjuntos aproximados para filtro de spam. Contornam o problema de concept drift utilizando a geração dos conjuntos de regras como forma mais eficaz, de acordo com o desempenho do modelo de classificação
Idre Zliobataite [61]	Precisão do classificador	Apresenta uma aplicação com **seleção de amostras relevantes** para o treino baseada na similaridade observando as amostras alvo. O *concept drift* é identificado a cada amostra com a projeção da similaridade de espaço e tempo para ser incluída na base de treino como novo conhecimento.
Angel [62]	Predição do classificador	A partir de um *framework* baseado em um meta-modelo que detecta *drift* recorrente observando a distribuição de probabilidade entre o espaço de características que gerou a modelo base e as características que surgem ao longo tempo. O meta-modelo indica o modelo que melhor representa o aprendizado para o momento corrente.
Lu et al. [63]	Baixa confiabilidade das amostras	Os autores contornam o *concept drift* a partir de uma medida de confiabilidade empregada em conjunto com um sistema CBR (*case-based reasoning*). Sem nenhum conhecimento prévio, apenas observando as medidas de confiabilidade, a atualização do modelo de classificação é efetuada a partir da adição dos casos que possuem baixa confiabilidade.
Proposta desta tese	Limiar de deteção	Baseada no monitoramento da evolução das características, a deteção de mudança é inferida a partir e um limiar pré-determinado que se antecede à degradação do modelo de

Autores	Método de deteção	Método de Reação
		classificação, **selecionando as amostras** que contemplam as características mais relevantes para compor o novo conhecimento.

A próxima seção apresenta aplicações que empregam o conceito de transferência de aprendizado, para atualizar o modelo de classificação.

3.2. Transferência de Aprendizado

Duas importantes questões em muitos dos domínios do mundo real são: a mudança no ambiente que ocorre dinamicamente [65] e a necessidade de se manter a base de conhecimento sempre atualizada. Por exemplo, em uma aplicação AntiSpam, os *spammers* costumam inserir aleatoriamente caracteres no corpo do e-mail ou no cabeçalho, para descaracterizar as assinaturas e ludibriar as aplicações de deteção estática.

Dentre as abordagens de transferência de aprendizado apresentadas no Capítulo 2, seção 2.3.3, as aplicações apresentadas nesta seção compreendem a abordagem de representação de características, visando a transferência do aprendizado baseado na melhor representação de características para descrição do problema de deteção de spam.

Schilmmer e Granger [55] propõem um método de transferência de aprendizagem baseado na representação de conceito distribuído, que é composto por um conjunto de pesos e símbolos. Modificações na descrição do conceito ocorrem em dois níveis: ajuste de pesos e geração de nova caracterização dos símbolos. O aprendizado anterior é utilizado na tarefa de obter conceitos subsequentes por adicionar um atributo de cada conceito adquirido anteriormente na descrição da nova instância.

O sistema proposto por Windmer e Kubat [34] mantém uma janela de exemplos correntes de confiança e hipóteses. Esse método armazena contextos anteriores e reusa-os quando um contexto anterior reaparece. O controle ocorre por meio de uma heurística que monitora constantemente o comportamento do sistema, ou seja, mantendo um tamanho de janelas fixo. As amostras novas que chegam compõem o novo conceito e as amostras mais antigas são excluídas do aprendizado.

Em [66], os autores aplicam o modelo de aprendizado lento (do inglês *lazy learning*) e uma abordagem de raciocínio baseado em instância (do inglês *Instance-Based Reasoning* - IBR) para resolver o problema de rotular e filtrar *spam*. Aprendizado lento

[67] é um método que adia toda a computação essencial até que a tarefa de predição seja completamente determinada. Dessa forma, o sistema é capaz de armazenar uma representação flexível de e-mails e a implementação de um estágio de revisão no processo IBR. O sistema emprega um conhecimento geral na forma de meta-regras que são extraídas do cabeçalho do e-mail. A fase de aprendizado é realizada atualizando-se a estrutura do conhecimento de todo o sistema, considerando a retirada do conhecimento irrelevante por meio da representação de características.

Ling et. al [68] propõem um framework para classificação espectral em que a função objetivo é usada para procurar uma consistência entre o domínio fonte e o domínio alvo. Por meio da otimização da função custo, as informações rotuladas a partir do domínio fonte de dados são transferidas para ajudar na classificação dos dados do domínio alvo. Os autores definem esse método como transferência de aprendizado de domínio. Isso porque a limitação dos métodos tradicionais de classificação espectral considera que os dados do domínio fonte e alvo têm o mesmo espaço de características, hipótese nem sempre verdadeira em cenários do mundo real. Por exemplo, no cenário de filtro de e-mail *spam,* o conjunto de dados de um domínio fonte pode apresentar um espaço de características diferente do domínio alvo. Como solução, a proposta dos autores é transferir, por meio de uma função objetivo, o aprendizado para o domínio alvo.

Meng et.al. [69] calculam a similaridade entre as características específicas alvo e características comuns com método de decomposição do valor singular (SVD – *Singular Value Decomposition*) a fim de aprender uma representação comum de característica. As características comuns se referem ao conjunto de características que é comum entre o domínio fonte e o domínio alvo. A fim de manter mais recursos, enriquecendo as informações relacionadas e de conhecimento entre os dois domínios, é calculada a semelhança entre as características específicas alvo e as características comuns em um espaço de características semântico construído por SVD. Define-se também uma função de características semelhantes por reunir características aparentemente não relacionadas para detetar o conhecimento relevante nos dois domínios. Por meio da função é aprendida uma nova representação de características comuns e é obtido mais conhecimento para aumentar a classificação com precisão na deteção de *spam* e não *spam*.

Duan et. al [70] apresentam um *framework* denominado *Domain Transfer Multiple Kernel Learning* (DTMKL) de aprendizagem unificada *cross-domain* do kernel, no qual o kernel ideal é aprendido por meio da minimização da incompatibilidade de

distribuição entre os domínios auxiliares e alvo, amostras rotuladas e não rotuladas. O aprendizado se consolida por meio da busca por um kernel com desempenho ótimo que, consequentemente, construirá um classificador sempre atualizado. A função kernel é gerada sobre as características que compõem a base de dados. A base de dados é composta pela frequência das caracerísticas que descrevem as amostras de spam e nonspam. A maior dificuldade apresentada pela aplicação é a complexidade computacional, pois para um classificador SVM, a complexidade é normalmente O($n^{2.3}$), equanto que para DTMKL, é *5* x *O*($n^{2.3}$). Portanto, esse método é inviável para problemas de ataques spam.

Gao et. al [71] apresentam um *framework* baseado em conjuntos de classificadores para combinar vários modelos na tarefa de transferência de aprendizagem. O algoritmo SVM é empregado para predição e um método de agrupamento é usado para gerar uma estrutura na forma de grafos. A estrutura de grafos tem como objetivo obter a similaridade entre as estruturas dos modelos de treino em relação à estrutura de uma distribuição de teste. Os pesos são atribuídos dinamicamente de acordo com o poder preditivo de um modelo em cada exemplo de teste e a similaridade entre as estruturas de grafos. A implementação das atribuições de pesos locais é feita mapeando as estruturas de um modelo para as estruturas do domínio de teste, e em seguida, a ponderação de cada modelo localmente de acordo com a coerência da estrutura vizinha em torno do exemplo de teste. O framework é avaliado sobre uma base de dados spam.

Pham et. al [72] apresentam uma arquitetura de sistema baseada em agente. Essa arquitetura é projetada para inferir quais palavras chaves um e-mail spam possui. O conhecimento inicial do agente é construído pelo usuário. O agente construído é aplicado na primeira camada do sistema. A primeira camada é inteiramente baseada no conhecimento inicial do agente sem intervenção do usuário para trazer e-mails legítimos para os usuários. A segunda camada é finalizada pela supervisão do usuário, os e-mails trazidos para os usuários por meio da primeira camada são filtrados pelo usuário e os e-mails definidos pelo usuário como spam, tornam-se dados de treino adicionais para aprendizado incremental do agente. A arquitetura do sistema agente trabalha com uma base de dados, os quais são representados por características extraídas a partir dos termos chaves do assunto e do corpo dos e-mails.

Shi et. al [73] apresentam um princípio para conceber um objetivo de otimização que preserva a estrutura original de dados, enquanto que, ao mesmo tempo, maximiza a semelhança entre os dados. O primeiro passo busca unificar o espaço de dados

heterogêneo, considerando um subespaço de características semelhantes entre o domínio fonte e alvo para preservar a estrutura original. Esse objetivo de otimização é especifico para resolver três questões: (1) dados fontes possivelmente são gerados a partir de espaço de características diferente dos dados alvo (características de texto (fonte) e imagens (alvo)); (2) espaço de características entre dados fonte e alvo são gerados por distribuições diferentes (distribuição de Gauss e distribuição multinominal); (3) dados fonte e alvo têm saída de espaço de características totalmente diferentes. Uma estratégia de seleção de amostra é aplicada para selecionar apenas os exemplos relacionados com o novo treinamento de dados e uma abordagem bayesiana é derivada para modelar a relação entre diferentes espaços de saída. Assim, são extraídos dados de treino a partir de fontes heterogêneas para ajudar a aprender os dados alvo.

Bahadori, Liu e Zhang [74] apresentam um *framework* geral para aprendizado denominado de transferência transdutiva, o qual aprende diferentes mapeamentos de funções a partir dos domínios observados e a função de previsão, ao mesmo tempo, minimiza o erro de reconstrução e o erro de classificação transdutiva de uma função unificada. O classificador utilizado é o mesmo, treinado por meio dos diferentes domínios no espaço latente, enquanto que as funções de mapeamento a partir do espaço de características são desconhecidas e diferem de domínio para domínio. A transferência do aprendizado ocorre por meio das funções de mapeamento sobre o espaço de características do domínio fonte e alvo.

3.2.1. Síntese

A categoria de transferência de aprendizado transdutivo, usando a abordagem de representação de características, mostra-se como a configuração que melhor atende o problema apontado por esta tese, isto é, detetar mudança por meio da evolução de características de ataques utilizando transferência de aprendizado para manter atualizado um repositório de conhecimento. A transferência de aprendizado visa diminuir a diferença entre os dois domínios fonte e alvo. Desta forma, esta tese foca na representação de características para minimizar a diferença entre esses dois domínios, baseada no aprendizado do domínio fonte para predizer um domínio alvo.

A Tabela 2.3 exibe a correlação das abordagens de transferência de aprendizado com as configurações do ambiente de transferência do aprendizado. Para o problema de atualização de aprendizado, a categoria transferência de aprendizado transdutivo com a abordagem de representação de característica é a melhor configuração para o problema

de evolução de características de ataques em redes de computadores por um simples motivo: a abordagem de transferência de aprendizado transdutivo mantém as mesmas tarefas fonte e alvo, enquanto que os domínios fonte e alvos são diferentes, exatamente como o problema abordado nesta tese, isto é, ataques *spam*. E a abordagem de representação de características apresenta a relação necessária com a transferência de aprendizado transdutivo para atender o objetivo proposto por esta tese. Dessa forma, busca-se uma boa representação de características para reduzir a diferença entre os domínios fonte e alvo.

Algumas questões conduzem o andamento desta pesquisa e dos experimentos:

1. Primeira questão: sabe-se que existe uma metamorfose ou evolução de características em problemas de ataques a redes de computadores. Os atacantes sempre evoluem a maneira como os ataques causam danos ao sistema. Logo, características que eram relevantes em um tempo t talvez não sejam mais relevantes em um tempo t+1 e novas características apareçam no domínio alvo. Então surge a pergunta: como identificar essas características novas e determinar qual ou quais são agora relevantes para o problema?
2. Segunda questão: como gerar um novo espaço de características baseado na seleção de características a partir do domínio fonte e alvo e que esse novo espaço de características realmente seja um aprendizado positivo?
3. Terceira questão: as características que não são comuns ao domínio fonte e domínio alvo e que estão presentes somente no domínio alvo, podem representar uma evolução real no comportamento do ataque?

A Tabela 3.2 exibe uma comparação entres as diversas aplicações apresentadas nesta seção visando salientar a abordagem de transferência de aprendizado e a estratégia empregada para compor o novo conhecimento. A maioria utiliza a transferência de parâmetro ou instâncias. Poucas empregam a abordagem de transferência de representação de características.

Com base nas aplicações mais recentes identificas no estado da arte, o objetivo desta tese é construir um método para deteção de ataque *spam* orientada à análise da evolução das características com base nos conceitos de deteção de mudança e transferência de aprendizado. Para compor o novo conhecimento a representação das características mais relevantes devem ser empregadas. Como resultado, espera-se manter as taxas de precisão dos sistemas estáveis e elevadas, diminuir as taxas de falso alarme e

minimizar a interferência humana. O próximo capítulo discute os métodos propostos nesta tese.

Tabela 3.2. Comparação dos Trabalhos Relacionados considerando os Abordagem de Transferência de Aprendizado e Estratégia empregada

Autores	Abordagem de transferência de aprendizado	Estratégia
Schilmmer e Granger [59]	Transferência de representação de características	Baseado na representação de conceito distribuído, o aprendizado é composto por um conjunto de pesos e símbolos atribuídos às características que descrevem o problema.
Windmer e Kubat [34]	Transferência de instâncias	Baseado em tamanho de janelas fixas, o aprendizado é construído sobre a chegada de novas amostras e a exclusão de amostras mais antigas.
Fdez-Riverola at. all [66]	Transferência de instâncias	Baseado em uma técnica de remoção de aprendizado irrelevante observando a frequência dos termos irrelevantes em mensagens spam e legítimas. A pequena diferença entre estas frequências mostra pobre relevância, enquanto as grandes diferenças são indicativos de condições adequadas para a classificação de spam.
Ling at. all [68]	Transferência de parâmetros	Baseado em um método de transferência de aprendizado de domínio, uma função objetiva é empregada para procurar uma consistência entre domínio fonte e alvo.
Meng et.al. [69]	Transferência de representação de características	Baseado no cálculo de similaridade entre as características especificas alvo e características comuns com método de decomposição do valor singular (SVD) a fim de aprender uma representação comum de característica entre os dois domínios.
Duan at. all [70]	Transferência de parâmetros	Baseado em [70][70][70][70][70][71][71][71][71][71][70][69]um framework denominado *Domain Transfer Multiple Kernel Learning* (DTMKL) de aprendizagem unificada *cross-domain* do *kernel*. O aprendizado se consolida por meio da busca de um *kernel* com desempenho ótimo que, consequentemente, construirá um classificador sempre atualizado.
Gao at. all [71]	Transferência de parâmetros	Baseado em um framework de conjuntos para combinar vários modelos na aprendizagem de transferência. Através de uma estrutura de grafos, o objetivo é obter a similaridade entre as estruturas dos modelos de treino em relação à estrutura de uma distribuição de teste. A transferência do aprendizado ocorre por meio da observação do peso local da estrutura dos grafos.
Pham at. all [72]	Transferência de instâncias	Considerando a semântica e associações estatísticas, um proxy pode agregá-los em uma estrutura hierárquica. Dado um conjunto de e-mails spam marcados por vários usuários de e-mail, o servidor proxy pode extrair dois tipos de recursos textuais, que são termos a priori e termos de conceito com base em frases-chave. A transferência do conhecimento se efetua através da ontologia de mensagem spam.

Autores	Abordagem de transferência de aprendizado	Estratégia
Shi at. all [73]	Transferência de instâncias	Baseado no princípio de conceber um objectivo de optimização para preservar a estrutura original de dados, enquanto que, ao mesmo tempo, maximiza a semelhança entre os dados. Unificando o espaço de características entre domínios heterogêneos, a transferência de aprendizado é feita pela seleção de instâncias unificadas.
Bahadori, Liu e Zhang [74]	Transferência de parâmetros	Baseado em um *framework* geral para aprendizado de diferentes mapeamentos de funções a partir dos domínios observados. A transferência do aprendizado ocorre por meio das funções de mapeamento sobre o espaço de características do domínio fonte e alvo.
Proposta desta Tese	Transferência de instâncias	A transferência do aprendizado ocorre por meio das instâncias que contemplam as características de melhor representatividade para o problema de deteção de ataques spam. Observando o período de tempo seguinte ao da construção do modelo de classificação, e considerando os seguintes critérios: o que, como e quando transferir as novas amostras.

Capítulo 4

Método de Deteção de Spam Baseado na Evolução das Características de Ataque

Os modelos de classificação dos tradicionais sistemas de deteção de intrusão são comumente gerados a partir de características obtidas de forma estática, ou seja, são gerados por um conjunto de características que representam o problema em um instante *t* do tempo. Como consequência, essas soluções são pouco efetivas para lidar com problemas que evoluem com o tempo, como é o caso da maioria dos ataques atuais.

Para tratar esse problema, este Capítulo apresenta uma nova abordagem para deteção de ataques do tipo spam baseada na evolução das características dos ataques. O capítulo está organizado em 5 seções. A seção 4.1 fornece uma breve visão sobre a arquitetura geral do método de detecção de ataques spam baseado na evolução das características de ataque. A seção 4.2 apresenta o processamento para entrada de dados empregando aprendizagem de máquinas. As três últimas seções apresentam as fases da arquitetura proposta separadamente e com detalhes específicos.

4.1. MECA: Visão Arquitetural

Na arquitetura proposta, denominada de Método de Deteção de ataque *spam*, baseado na Evolução das Características de Ataques *(*MECA*)*, a deteção dinâmica de ataques ocorre por meio do monitoramento da evolução das características dos ataques. Técnicas de monitoramento de evolução de características são utilizadas para identificar o momento em que ocorre mudança no problema investigado. A partir da deteção de mudança, o novo conhecimento descoberto com a evolução das características é transferido para o domínio fonte, mantendo-o sempre atualizado. Manter o domínio fonte atualizado é importante, pois se as mudanças forem suficientemente "significativas", é possível facilmente reconstruir o classificador para manter a taxa de precisão sempre estável e elevada, com

mínimo de interferência humana.

Assim, a principal ideia por trás de método MECA é obter dados rotulados ou conhecimentos extraídos do domínio de interesse para ajudar um algoritmo de aprendizado de máquina a obter maior desempenho sobre o problema.

Para alcançar esse objetivo, a arquitetura proposta foi construída em três etapas: treino e construção do modelo de classificação; deteção de mudança; e transferência de aprendizado, conforme ilustrado na Figura 4.1.

A etapa de treino e construção do modelo de classificação consiste em encontrar um modelo que descreva e assinale classes de dados [71]. Como na maioria dos sistemas de reconhecimento de padrões (o que inclui aplicações empregando AM tradicional), o modelo é primeiramente construído com base na análise de um conjunto de dados de treinamento. Em seguida, esse modelo é usado para examinar características de uma nova instância (para a qual a classe é desconhecida) e atribuí-la uma classe.

No método proposto, uma vez que o modelo de classificação é construído, inicia-se a fase de deteção de mudança, a qual monitora as amostras de dados com o objetivo de identificar quando uma mudança no padrão das características ocorre. Entretanto, detetar mudança nos dados, antes que o desempenho do modelo de classificação seja comprometido, ainda é um desafio, visto que nem toda mudança na distribuição de dados irá degradar o desempenho de um classificador [75][26].

Para tratar esse problema, esta tese propõe duas estratégias de deteção de mudança. Na primeira estratégia, denominada de *Historical-based Features Selection* (HFS), a deteção da mudança é baseada no histórico das características, observando a frequência e a ausência das características no domínio de treino em relação ao domínio de interesse. Mensagens de *spam* são problemas dinâmicos e tem uma natureza abrupta em suas mudanças [6], [26], [76]. Portanto, considerar a frequência das características para monitorar a mudança no padrão de uma mensagem *spam* pode aferir com maior precisão o momento exato para atualizar o domínio fonte sem depender de um *feedback* do modelo de classificação. Na segunda estratégia, denominada *Similarity-based Features Selection* (*SFS*), a deteção de mudança é observada a partir da mínima similaridade obtida pela comparação entre os vetores de características correspondentes ao domínio fonte e ao domínio alvo. Da mesma forma que a estratégia HFS foi motivada pela natureza do problema observando a frequência das características, a estratégia SFS

foi motivada pela mínima similaridade entre as características do domínio fonte e do domínio de interesse. A ausência de similaridade fornece indícios de mudança na distribuição dos dados e a necessidade de armazenagem de um novo conhecimento para alimentar o domínio fonte. Desta forma, as duas estratégias propostas para esta tese apresentam visões diferenciadas sobre a observação da evolução das características de ataques *spam*. A seção 4.4 detalha as estratégias propostas.

A última etapa da arquitetura do MECA corresponde à etapa de reação, onde dados rotulados ou conhecimento extraído da fase de deteção de mudança são empregados para ajudar um algoritmo de aprendizagem de máquina a alcançar um melhor desempenho sobre o domínio de interesse. A transferência do aprendizado ocorre concomitantemente à etapa de deteção de mudança.

Na estratégia HFS, a transferência do aprendizado é feita a partir das características mais frequentes no domínio alvo e ausentes no domínio fonte. As amostras correspondentes às características mais frequentes que contemplam a nova descrição do problema irão compor o novo conhecimento. Na estratégia SFS, a transferência do aprendizado ocorre por meio de uma medida de similaridade, que afere quão relacionados são os elementos do sistema, ou seja, os elementos (amostras) com nenhuma ou pouca similaridade, irão compor um novo conhecimento por apresentarem características não representativas no domínio de treino, mas relevantes no domínio alvo.

Portanto, independentemente da estratégia utilizada, a etapa de transferência de aprendizado utiliza a abordagem de representação de características, ou seja, a construção do novo conhecimento é totalmente baseada nas características que descrevem o problema no instante de tempo t.

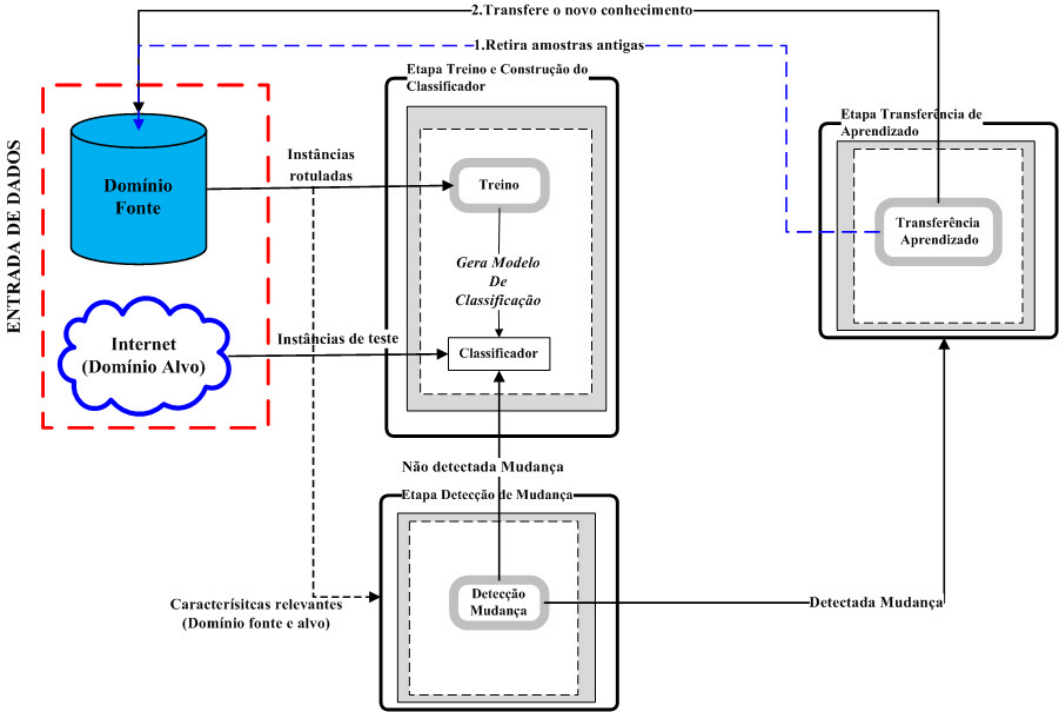

Figura 4.1. Visão geral da Arquitetura MECA. MECA é dividido em três etapas. A etapa de treino e construção do classificador gera o modelo de predição, enquanto a etapa de deteção de mudança observa um limiar para determinar uma mudança e a etapa de Transferência de aprendizado atualiza o domínio fonte.

4.2. Entrada de Dados

Assim como em outras aplicações que utilizam aprendizagem de máquina (AM), MECA também requer como entrada um conjunto de dados composto por amostras positivas (por exemplo, e-mails *spam)* e amostras negativas (por exemplo, e-mails legítimos, aqui denominados de *nonspam*). Para possibilitar o trabalho destes sistemas que usam AM, cada amostra precisa ser transformada em um vetor de características para ser submetido a algoritmos de aprendizagem. Desta forma, o conjunto de dados de entrada é organizado na forma de uma tabela atributo-valor.

Um dos procedimentos geralmente usados para tratar amostras em formato de documentos texto e adequá-los a um formato de vetor de características é a abordagem "*bag-of-words*"[61]. Nessa abordagem, o texto é representado como um vetor de termos (palavras que ocorrem no texto pelo menos uma vez) com seus respectivos pesos, obtidos a partir da frequência de cada palavra.

Entretanto, a representação de uma coleção de documentos por meio da abordagem *bag-of-words* pode ser muito custosa, pois o vetor que representa cada

documento deve conter todos os termos de toda a coleção de documentos. Para tratar esse problema, diferentes técnicas podem ser empregadas para reduzir a dimensionalidade dos dados durante a fase de pré-processamento como: remoção dos termos comuns ou irrelevantes (ou não discriminantes) para o algoritmo de classificação; exclusão de *stopwords* (artigos, preposições e símbolos); redução das palavras por meio de *stemming* (normalização linguística na qual as formas variantes de um termo são reduzidas a uma forma comum, radical das palavras); categorização de grupos de *stemmings* formando taxonomias, anonimização, ranqueamento dos termos mais importantes segundo algum critério, etc. [77][78]. A Figura 4.2 apresenta o processo de pré-processamento de dados para um MECA.

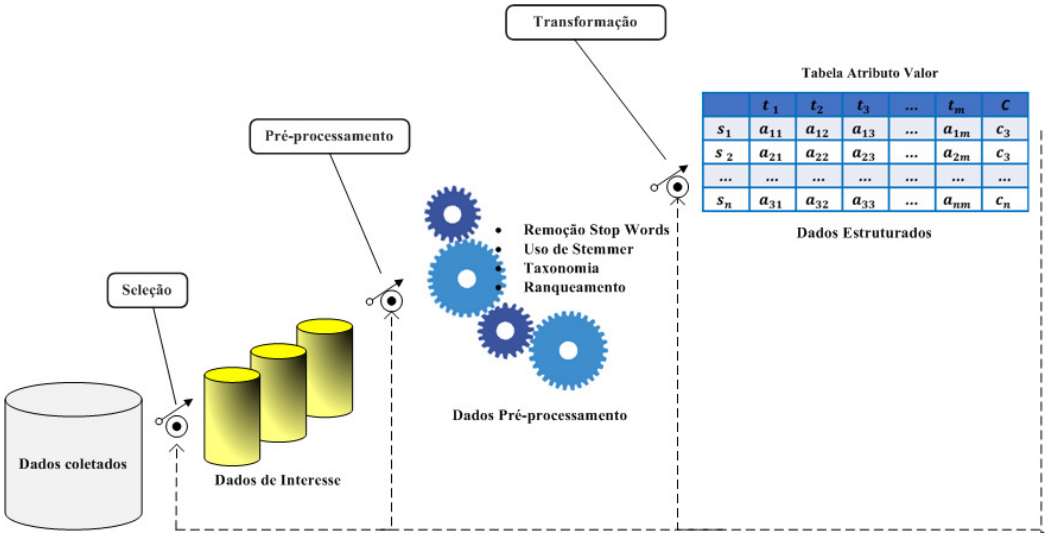

Figura 4.2. O processo de pré-processamento de dados para o MECA. A partir de uma coleção de dados brutos são selecionados dados de interesse para iniciar o pré-processamento aplicando alguma técnica para reduzir a dimensionalidade do espaço de características e transformar tais dados em vetores de características.

As técnicas empregadas na fase de pré-processamento estão diretamente relacionadas com o problema em análise. Por exemplo, nas soluções que tratam o problema de *spam*, por razões de segurança e privacidade, é comum o emprego de técnicas para remoção de nomes, endereços, telefones, e outras informações privadas. O Apêndice A apresenta detalhes do emprego de algumas técnicas de pré-processamento usadas na construção da base de dados utilizada nos experimentos desta tese.

4.3. Etapa de Treino e Construção do Modelo de Classificação

Esta seção apresenta como algoritmos AM constroem modelos de classificação sobre a

entrada de dados apresentada na seção anterior. O funcionamento do MECA na etapa de treino e construção de modelo de classificação não é diferente das aplicações que empregam aprendizagem de máquina tradicional.

O processo de treino e construção do modelo de classificação se resume em aplicar sobre o domínio fonte algum método de avaliação do modelo de classificação, dividindo-o em treino e validação. O conjunto de treino é utilizado como base de entrada do processo de aprendizagem, enquanto que o conjunto de validação é empregado para estimar a capacidade de generalização do modelo gerado durante o treino e para ajustar os parâmetros do algoritmo de aprendizagem. A Figura 4.3 exibe o processo de aprendizagem de máquina. **Figura 4.3**Métodos de avaliação do modelo de classificação têm como objetivo garantir que o modelo de classificação seja treinado sobre amostras diferentes das quais será avaliado e evitar *overffiting*. Alguns métodos mais empregados com este objetivo são baseados em validação cruzada: *k-fold cross validation*, *hold-out* e *live-one-out,* além de *bootstrap* [18].

O objetivo na fase de treino é alcançar bons resultados em termos de generalização do padrão de comportamento identificado para o problema. Para tanto, um algoritmo de AM precisa ser escolhido. Há diversos algoritmos de AM propostos na literatura, tais como: SVM, Árvore de Decisão, *Naive Bayes*, k-NN, dentre outros [79]. Para escolher o algoritmo de AM mais adequado é considerado o tipo de aprendizagem (supervisionada ou não supervisionada) e a distribuição dos dados. Após definido o algoritmo de AM, efetua-se a configuração de parâmetros para o algoritmo. Por exemplo, caso o algoritmo escolhido seja o SVM, dois parâmetros iniciais precisam ser definidos: o tipo de função *kernel* e o parâmetro de regularização *C*. Dependendo da função *kernel* escolhida, outros parâmetros devem ser definidos, como por exemplo, o grau do polinômio para o *kernel* polinomial.

Em geral, o critério de parada para o treinamento de um classificador é definido pelo usuário e é baseado na análise do desempenho do classificador. Geralmente a medida usada para identificar a degradação do modelo de classificação é o declínio na taxa de precisão, ou seja, a falta de capacidade de generalização do classificador sobre novas instâncias. Neste momento é feita a análise do melhor desempenho entre os modelos de classificação gerados com base em seus parâmetros para definir o classificador mais eficiente e eficaz para o problema.

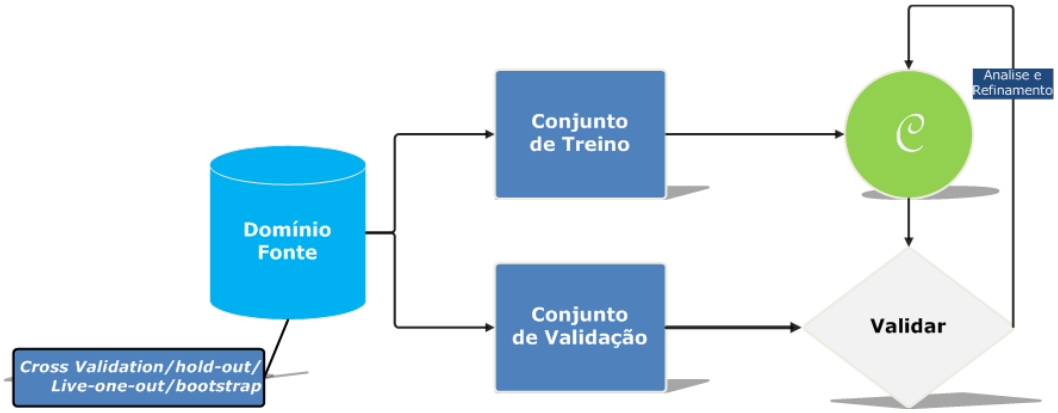

Figura 4.3. Etapa de Treino e Construção Modelo de Classificação. O conjunto de treino analisa e refina a configuração dos parâmetros para o modelo de classificação, medindo seu desempenho sobre o conjunto de validação.

4.4. Etapa de Deteção de Mudança

O objetivo da etapa de deteção de mudança é identificar a degradação do modelo de classificação, construído na etapa anterior, por meio do monitoramento da distribuição de dados do domínio fonte e alvo. Como mencionado no Capítulo 2 desta tese, o processo de deteção de mudança pode ser baseado em mudança na distribuição de probabilidade das classes, no padrão de características consideradas relevantes, na precisão do classificador ou mesmo nos modelos de classificadores utilizados, visto que alguns classificadores são mais sensíveis à mudança na distribuição de dados.

Para manter o aprendizado estável e não comprometer o bom funcionamento do MECA, esta tese considera o processo de deteção de mudança baseado na análise do padrão de características consideradas relevantes num instante *t* do tempo. Características ou mesmo combinações de características que foram relevantes no passado podem não ser suficientemente discriminativas num intervalo de tempo seguinte. Manter o controle sobre a melhor combinação de características que descreve o problema ao longo do tempo possibilita ao sistema antecipar a decisão de atualizar o classificador, visto que torna possível construir um classificador atualizado a partir da distribuição de dados mais recentes.

Visando antecipar o momento em que o modelo de classificação perde a capacidade de generalização, a etapa de deteção de mudança foi concebida para atender os seguintes requisitos:

1. Detetar a mudança tão rápido quanto possível.
2. Distinguir o limiar de tolerância do sistema e o ponto exato da deteção de mudança.
3. Adaptar-se às mudanças.

Para atender estes requisitos duas estratégias de deteção de mudanças são apresentadas: *Historical-based Features Selection* (HFS) e *Similarity-based Features Selection* (SFS). Ambas as estratégias observam a relevância das características sobre janelas de tempo e procuram identificar a relação entre a degradação do modelo de classificação e o conjunto de características que representam o domínio fonte (usado para construir o modelo de classificação) e alvo (domínio de teste). A primeira estratégia analisa as características observando as que são comuns aos dois domínios e aquelas que são ausentes no domínio fonte. Essa análise é baseada na atribuição de pesos às características de acordo com suas relevâncias na distribuição de dados. A segunda estratégia observa as características a partir de uma medida de similaridade entre a distribuição dos dados dos domínios fonte e alvo. A ausência de similaridade entre a distribuição dos dados, além de indicar o surgimento de novas características para a representação do problema, também pode servir como indicador para a atualização do modelo de classificação. Detalhes das duas estratégias propostas são apresentados a seguir.

4.4.1. *Historical-based Features Selection* (HFS)

A estratégia de deteção de mudança HFS tem como objetivo identificar a mudança no comportamento de um determinado ataque observando a representação das características que compõem o domínio fonte e alvo. Esta observação é baseada na análise do histórico das características considerando a frequência de cada uma delas como critério de relevância.

A aplicação da estratégia HFS inicia durante a fase de construção do modelo de classificação, conforme descrito na Seção 3.3, por se tratar também de um método de redução do espaço de dimensionalidade do vetor de características. Nessa fase, o HFS é aplicado para selecionar, usando o critério de frequência, o conjunto de caraterística mais relevantes. O número de características selecionadas para representar os dados de treinamento é determinado pela capacidade de generalização do classificador utilizado na

predição, ou seja, pela capacidade de manter o conhecimento inalterado diante de um conjunto de características reduzido.

Na estratégia HFS, esse conjunto de característica (espaço de dimensionalidade de características $\mathbb{R}^n$) é representado por um vetor de características $\boldsymbol{x_t} = [x_1, \dots, x_n] \in \mathbb{R}^n$, onde *t* representa uma janela de tempo. Como ilustrado na Figura 4.4, os domínios fonte e alvo são representados por um conjunto de amostras $Z = \{z_1, z_2, \dots, z_N\}$, onde cada amostra z é descrita por um conjunto de características $z_j = \{x_1, x_2, \dots, x_n\}$ representada em formato binário (0,1), o qual indica presença (1) e ausência (0) da característica na amostra e pelo conjunto de pesos $\boldsymbol{\omega} = \{\omega_1, \dots, \omega_n\}$, usado para determinar a relevância das características.

O parâmetro peso $\boldsymbol{\omega}$ permite construir um histórico das características com objetivo de facilitar a observação quanto à relevância destas ao longo do tempo. Inicialmente, a cada característica x_n, extraída a partir das amostras que compõem a janela de tempo inicial (domínio fonte), é atribuído o valor igual a 1 ao peso, conforme mostrado na Figura 4.4a.

O processo de deteção de mudança se inicia por meio da análise dos conjuntos de características ao longo do tempo. O vetor $\boldsymbol{x_t}$ é comparado aos vetores correspondentes das janelas de tempo seguintes, ou seja, com os vetores $\boldsymbol{x_{t+1}, x_{t+2}, \dots . x_{t+n}}$. Por exemplo, o vetor $\boldsymbol{x_1}$ representa o conjunto de características da primeira janela de tempo seguinte à geração do modelo de classificação. A análise dos vetores de características $\boldsymbol{x_0}$ e $\boldsymbol{x_1}$ possibilita o monitoramento da evolução das características após essa primeira janela de tempo. Nesta tese, uma janela de tempo pode ser definida em horas, dias ou meses.

A estratégia HFS considera as características diferentes e comuns entre os dois domínios para atribuição de peso $\boldsymbol{\omega}$ e geração de um novo vetor de características relevantes, como mostrado na Figura 4.4 b. O ajuste de pesos ocorre da seguinte maneira: é adicionado +1 (um) ao peso das características que permanecem existentes na janela atual e decrementado -1 (um) do peso das características já existentes na janela anterior, mas ausentes na janela atual. Além disso, para as características novas é atribuído valor 1 (um).

Para gerar uma nova representação do problema, após a análise sobre os dois domínios, a estratégia HFS elimina as características com peso igual à zero, conforme mostrado na Figura 4.4 b. Tal procedimento gera um novo vetor $\boldsymbol{x_{new}}$, contendo todas as

características relevantes ao longo do tempo, que será utilizado numa futura atualização do modelo de classificação.

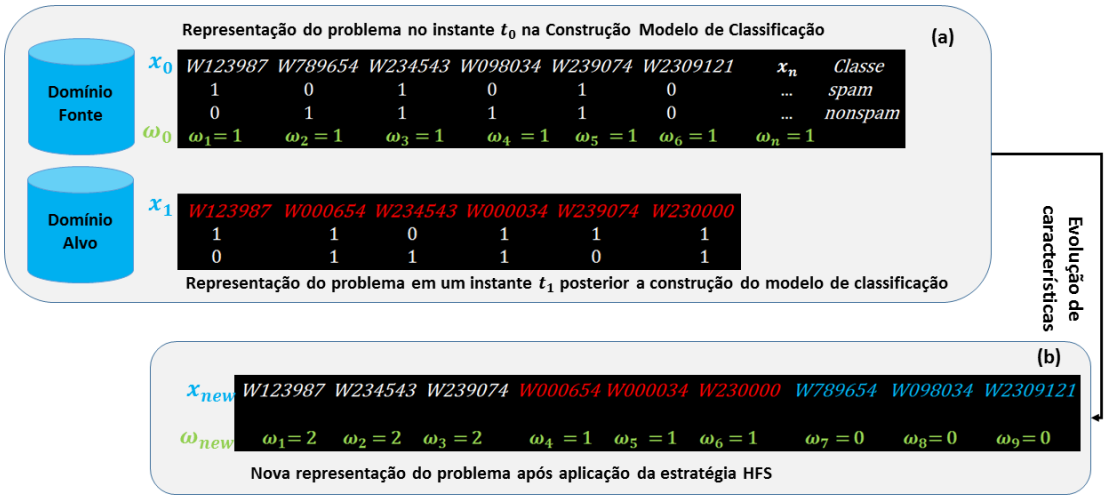

Figura 4.4. Estratégia HFS observa o histórico das características baseado no parâmetro peso para estimar o novo conhecimento; (a) análise entre os vetores ($\boldsymbol{x_0}$, $\boldsymbol{x_1}$) de representação do problema, domínio fonte e alvo; (b) geração do novo vetor de características mais relevantes baseada na frequência e peso.

Um pseudocódigo da estratégia HFS é exibido na Figura 4.5. O algoritmo HFS tem como entrada dois vetores de características correspondentes às amostras do domínio fonte e alvo ($\boldsymbol{x_f}$, $\boldsymbol{x_a}$) e a quantidade *N* de características, que representa as características mais relevantes no momento da geração do modelo de classificação. A partir desta entrada é feita a análise da evolução das características ponderando a igualdade, diferença e inexistência entre os dois vetores de características ($\boldsymbol{x_f}$, $\boldsymbol{x_a}$). As linhas 8-10 atribuem os pesos (1) às características que originaram o modelo de classificação. O momento de análise entre os dois vetores é apresentado nas linhas 12-26. Esta análise observa a relevância das características que se modificam ao longo do tempo gerando um histórico a partir do parâmetro peso. As linhas 28-39 apresentam o momento de normalização do novo vetor de características para gerar o novo conhecimento para uma futura atualização no domínio fonte.

```
1 x_f = vetor de características fonte
2 x_a = vetor de características alvo
3 N = quantidade de características para selecionar
4 i = 0
5 j = 0
6 ω = 0

7 /*Atribuindo peso às N características mais relevantes do domínio fonte*/
8 for (cada x_f [i] em x_f)
9    ω_a [j] = 1
10 endFor

11 /* A partir dos vetores fonte e alvo são atribuídos valores ao parâmetro
      peso de acordo com a representatividade da característica entre os dois
      vetores*/
12 for (cada x_f [i] em x_f)
13   for (cada x_a [j] em x_a)
14     if (x_f [i] != x_a [j])
15       New_Vet_Conhecimento = x_a [j]
16       ω_a [j] = 1
17     else
18       if (          (x_f [i] == x_a [j])
19         New_Vet_Conhecimento = x_f [i]
20         ω_f [i] = pω_i +1
21       else
22           New_Vet_Conhecimento = x_f [i]
23           ω_f [i] = ω_i - 1
24     endIf
25   endFor
26 endFor

27 /* Construindo NOVA representação de características para PROBLEMA*/
28   for (cada x_new[i] em New_Vet_Conhecimento [])
29     if (ω_f [i] != 0)
30       while (x <= N)
31         if (((F x_new[i] ) and ( ω x_new[i] ) )< ((F x_new[i+1] ) and ( ω x_new[i+1] ))then
32           Novo_Conhecimento []= x_new[i+1]
33         else
34           Novo_Conhecimento []= x_new[i]
35       endWhile
36     endIf
37     j++
38   endFor
39 return Novo_Conhecimento
```

Figura 4.5. Algoritmo da estratégia HFS

O Capítulo 4 mostra que a estratégia HFS, por apresentar uma análise da evolução das características empregando um critério de peso e frequências das características, é robusto na tarefa de deteção de mudança gradual. Em essência, o HFS necessita de uma observação maior ao longo do tempo para construir o histórico da relevância das características, como será mostrado no Capítulo 4. Para lidar com problemas que apresentam mudança abrupta na distribuição dos dados, esta tese propõe uma segunda estratégia de deteção de mudança baseada no nível de similaridade das características, conforme descrito a seguir.

4.4.2. *Similarity based Features Selection* (SFS)

A estratégia de deteção de mudança SFS tem como objetivo identificar a mudança no comportamento de um determinado ataque observando a representação das características que compõem os domínios fonte e alvo. Esta observação é baseada na análise da similaridade das características que representam o problema no momento da construção do modelo de classificação em relação ao momento seguinte.

Essa estratégia baseia-se no uso de um modelo vetorial e é inspirada em problemas da área de recuperação da informação. Cada amostra, representada por um vetor de características, é considerada uma palavra. Por exemplo, para problemas de ataques do tipo *spam*, o modelo vetorial estabelece uma relação entre dois vetores (amostras) $z_j = \{x_1, x_2, \dots, x_n\}$ por meio de uma medida de similaridade. A medida de similaridade é uma tentativa de mensurar o quão relacionado são os dois vetores analisados. Para tanto, deve-se calcular o valor de similaridade entre cada amostra alvo $\vec{\boldsymbol{d_j}}$ e a amostra fonte $\vec{\boldsymbol{q}}$.

O cálculo de similaridade pode demandar alta complexidade computacional, pois, a similaridade será calculada entre o conjunto de amostras que representam o domínio alvo D_a e o conjunto de amostras que representam o domínio fonte D_f. Por exemplo, se o domínio fonte for representado por 1000 amostras, as quais originam o modelo de classificação, será feito o cálculo de similaridade para cada documento do domínio alvo em relação ao domínio fonte, originando uma complexidade computacional $\mathbf{1000} \times \boldsymbol{n}$, onde *n* representa o número de amostras do domínio alvo.

Com o objetivo de minimizar está complexidade computacional, o algoritmo *K-means* foi aplicado sobre o domínio fonte D_f para extrair os centroides que melhor representam as classes. Assim, a complexidade computacional é minimizada durante o cálculo do modelo vetorial, pois, se o problema for representado por duas classes, por exemplo, *spam* e nonspam, o cálculo do modelo vetorial torna-se $\mathbf{2} \times \boldsymbol{n}$. Embora a complexidade seja reduzida, o método permanece efetivo, uma vez que será estimada a similaridade entre cada instância nova e o centroide do domínio fonte, a fim se estimar mudanças nos dados.

A fórmula para calcular a similaridade entre dois elementos, proposta por Salton [80][81], consiste em se computar o cosseno do ângulo entre os vetores destes dois elementos. Por exemplo, a similaridade entre uma amostra fonte *q* e uma amostra alvo *d* é dada pela equação 4.1, onde $\boldsymbol{q}$ equivale à consulta e $\boldsymbol{d}$ aos documentos. A correlação

entre os dois vetores é utilizada para medir a proximidade entre os elementos reais modelados.

$$\boldsymbol{sim}(d,q) = \cos\theta \ = \frac{\sum_{i=1}^{t} w(i,d)*w(i,q)}{\sqrt{\Sigma\left((i,d)\right)^2} * \sqrt{\Sigma\left((i,q)\right)^2}} \tag{4.1}$$

A partir do cálculo de similaridade entre as amostras do domínio alvo e as amostras do domínio fonte, busca-se definir um limiar para inferir a mudança. Este limiar está diretamente ligado a um limite de tolerância suportado pelo sistema, ou seja, até que ponto a similaridade entre os dois domínios pode ser reduzida sem que ocorra degradação do desempenho do modelo de classificação? Esse ponto pode ser um indicador para inferir uma mudança real nos dados.

A identificação de uma mudança virtual é facilmente perceptível por meio da estratégia SFS, ou seja, mudança na distribuição de dados que não compromete o desempenho do classificador. As instâncias do domínio alvo que não são similares às instâncias do domínio fonte são armazenas em um repositório de mudança virtual, para que, após a deteção de uma mudança real, estas amostras sejam utilizadas para atualizar o conhecimento do classificador. No entanto, a identificação de uma mudança real é mais complexa, pois é necessário definir o limiar, de forma que não ocorra degradação do desempenho do modelo de classificação.

A Figura 4.6. apresenta a etapa de deteção de mudança. O domínio fonte, $D_f = \{(x_{f_1}, y_{f_1}), \ldots, (x_{fN}, y_{fN})\}$, juntamente com o domínio alvo, $D_a = \{(x_{a_1}, y_{a_1}), \ldots, (x_{aN}, y_{aN})\}$, conforme definidos na seção 2, alimentam o módulo de deteção de mudança. Uma estratégia de deteção de mudança deve ser aplicada neste módulo. Este módulo é bastante flexível, a única premissa considerada comum a todas as possíveis estratégias aplicadas é a ação de detetar mudanças. A estratégia empregada pode ser alterada de acordo com o problema abordado. Enquanto a estratégia de deteção de mudança não determinar a ocorrência de uma mudança, que comprometa o desempenho do modelo de classificação $\mathcal{C}$ a tarefa de classificação continua sendo exercida pelo modelo originado no instante de tempo $\boldsymbol{t_0}$. Por outro lado, para mudanças detectadas na distribuição de dados, as características (ou amostras) que indicam essa mudança são armazenadas em um repositório de mudança virtual. Por fim, quando uma mudança real é detectada, o sistema providencia a construção de um novo modelo de

classificação. Para isso é necessário atualizar o domínio fonte por meio de etapa de transferência de aprendizado.

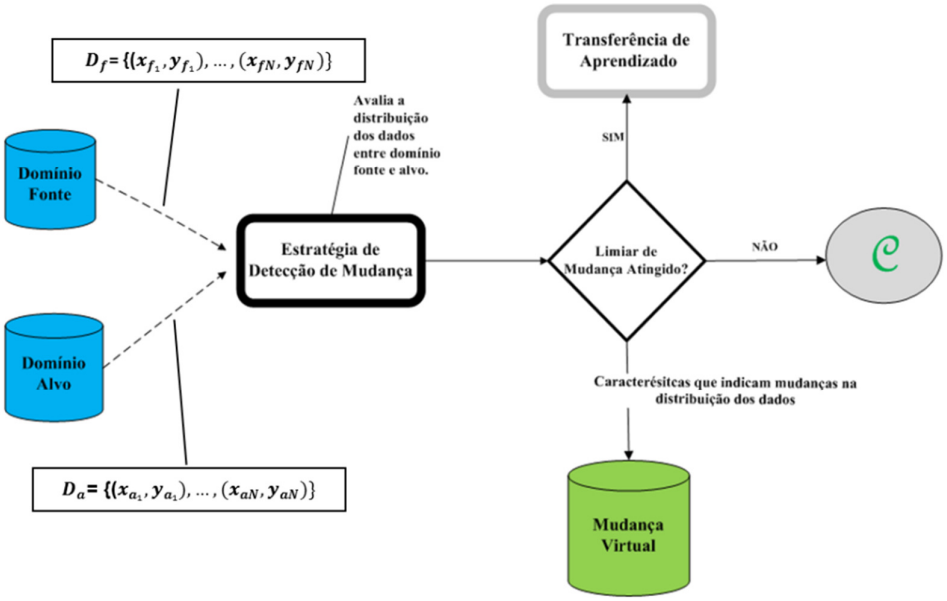

Figura 4.6. Etapa de Deteção de Mudança determina o momento de uma mudança virtual e/ou real. Enquanto a mudança não é detectada, a tarefa de classificação é enviada para o classificador originado no instante de tempo t_0 e as características identificadas com maior relevância entre os dois domínios são armazenadas em um repositório de mudanças virtual que construirá um novo conhecimento.

A próxima etapa do MECA utiliza-se do conceito de transferência de aprendizado, atendendo o critério de transferência positiva, conforme definição na seção 2.3.

4.5. Etapa de Transferência de Aprendizado

O objetivo da etapa de transferência de aprendizado é manter o domínio fonte, ou domínio de construção do modelo de classificação, sempre atualizado para minimizar danos ou prejuízos ao sistema. O maior desafio desta etapa é diminuir o risco de transferência negativa, a qual ocorre quando o desempenho do modelo de classificação não melhora, mesmo que o domínio fonte seja atualizado com novo conhecimento.

Assim, esta etapa apresenta alguns critérios de transferência de aprendizado, definidos anteriormente no Capítulo 2:

- O que transferir?

- Como transferir?
- Quando transferir?

Como mencionado na seção precedente, deteção de mudança e transferência de aprendizado ocorrem simultaneamente. Portanto, os critérios para transferência são atendidos durante a etapa de deteção de mudança. O critério de "o que transferir" é determinado pelas estratégias de deteção de mudanças. Por exemplo, na estratégia HFS, o novo conhecimento é inferido pelo histórico e frequência das características que descrevem as amostras em janelas de tempo, ou seja, serão armazenadas as características mais relevantes considerando o histórico na evolução das características ao longo do tempo. Por outro lado, a estratégia SFS considera a similaridade entre as características que descrevem as amostras entre duas janelas de tempo t_0 e t_1. Quanto menor a similaridade entre os vetores observados, maior a probabilidade de mudança. As amostras que apresentam baixa similaridade são armazenas como novo conhecimento para uma próxima atualização do domínio fonte e construção de uma transferência positiva, ou seja, um conhecimento que melhorará o desempenho do modelo de classificação.

O critério de "como transferir" emprega a abordagem de transferência de representação de características, definida na seção 2, visto que toda a arquitetura MECA é baseada na relevância das características em janelas de tempo. Este critério é atendido pela instância que contempla as características díspares entre os dois vetores avaliados, as quais são detectadas a partir das estratégias de mudanças HFS e SFS. As novas características que surgem ao longo do tempo são transferidas. O último critério "quando transferir" é inerente à etapa de deteção de mudança, isto é, a transferência do novo conhecimento deve ser feita tão logo mudanças que comprometem o desempenho da tarefa de classificação sejam identificadas.

A Figura 4.7 exibe o funcionamento da fase de transferência de aprendizado. Após as novas instâncias serem recebidas, a fase de transferência de aprendizado verifica a quantidade de amostras que precisam ser adicionadas ao domínio de fonte. O novo conhecimento é transferido por meio das amostras que contemplam as características mais atuais que descrevem o problema. Como critério para manter o domínio fonte balanceado com amostras positivas e negativas, as amostras mais antigas são removidas para que as novas amostras sejam inseridas. Uma vez finalizada a transferência, aciona-se a etapa de treino e construção do novo modelo de classificação a partir do novo conhecimento. Desta forma, espera-se gerar um modelo de classificação mais atualizado.

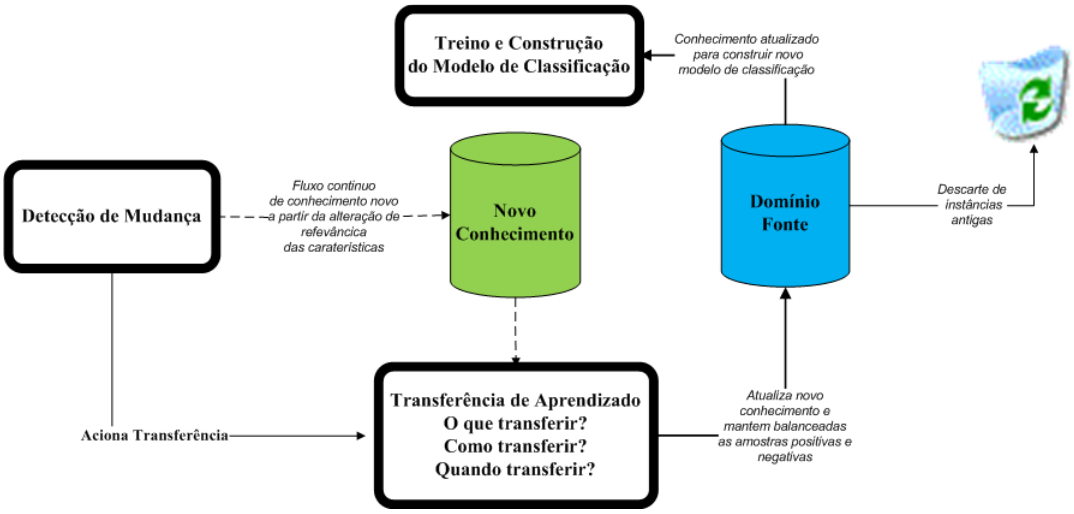

Figura 4.7. Etapa de Transferência de Aprendizado se preocupa em o que, como e quando transferir o novo conhecimento para atualizar o domínio fonte. Instâncias antigas são descartadas para manter o balanceamento das amostras entre as classes da representação do problema.

4.6. Considerações

O maior desafio durante o projeto de uma arquitetura de sistemas de deteção de *spam* baseada na evolução das características ao longo do tempo é determinar o instante necessário em que o sistema de deteção precisa ser atualizado, preservando o seguinte critério: a deteção de mudança no comportamento de ataques deveria acontecer antes de prejudicar ou danificar o sistema. Para isso, a deteção de mudança não deveria se basear na taxa de precisão do modelo de classificação.

Assim, esta tese argumenta que a observação sobre a distribuição dos dados que compõem o vetor x_0 de características usado para construir o modelo de classificação e o vetor de característica x_1 em um instante seguinte, não é suficiente para determinar a necessidade de atualização de um sistema de deteção de *spam*. Uma mudança na distribuição das classes nem sempre compromete o desempenho do modelo de classificação [32], ou seja, como o modelo de classificação apresenta um intervalo de tolerância às mudanças, ou seja, o classificador não reduz drasticamente sua taxa de classificação correta imediatamente após a ocorrência de mudanças no problema. Com isso, um intervalo entre limiar de tolerância e a deteção de mudança, é utilizado para armazenar o conhecimento novo a partir das diferenças apontadas entre as características do instante de tempo t_0 e t_1 por meio das estratégias propostas nesta tese, seção 4.4. Sendo este o maior desafio apresentado na etapa de deteção de mudança, isto é,

determinar o momento exato da ocorrência de um desvio do que era considerado relevante em um momento t_0 e deixa de ser relevante no momento seguinte t_1.

O próximo capítulo apresenta os resultados experimentais obtidos a partir da utilização do MECA. As duas estratégias de deteção de mudança discutidas neste capítulo são investigadas.

Capítulo 5

Experimentos e Resultados

Este Capítulo apresenta os resultados obtidos por meio da avaliação de desempenho das estratégias de deteção de mudança baseadas na evolução das características dos ataques propostas no Capítulo 4. O capítulo inicia (Seção 5.1) com a apresentação do protocolo experimental descrevendo as bases de dados, a métrica usada para estimar a eficiência das estratégias propostas, bem como uma descrição das funções e parâmetros adotados para gerar o modelo de classificação SVM. A Seção 5.2.1 apresenta os resultados da avaliação do comportamento estático de um modelo de classificação (comumente adotado pelas atuais aplicações empregando AM) e a sua relação com a evolução das características. As seções 5.2.2 e 5.2.3 discutem os resultados obtidos pelo classificador utilizando as estratégias HFS e SFS.

5.1. Protocolo Experimental

5.1.1 Bases de Dados

Os experimentos deste capítulo foram realizados utilizando as bases de dados: ECUE (*Email Classification Using Examples*) - base pública disponibilizada na web em http://www.dit.ie/computing/staff/sjdelany/datasets/; e ETSS (*Emerging Technologies and System Security*) - base de dados desenvolvida no âmbito deste projeto de pesquisa. A seguir, essas duas bases são descritas em detalhes.

5.1.1.1. ECUE (Email Classification Using Examples)

Esta base de dados corresponde a uma coleção de *spam* e e-mails legítimos recebidos por um indivíduo no período de Novembro de 2002 a Janeiro de 2004. A base está dividida em dois conjuntos de amostras: base de treino e base de teste. A base de treino corresponde à coleção de e-mails recebidos nos meses de Novembro e Dezembro de 2002 e Janeiro de 2003. A base de teste corresponde à coleção de e-mails recebidos nos meses de Fevereiro de 2003 a Janeiro de 2004. Os e-mails legítimos incluem uma variedade de

correspondências pessoais, empresariais e mensagens de listas de discussões.

As amostras de treino e de teste são representadas por meio de três tipos de características extraídas dos e-mails:

- *Características de palavras* – que representam sequências de caracteres separados por espaço em branco ou pelo delimitador *tag* HTML;
- *Características de caracteres* - que representam a ocorrência de um único caractere no e-mail.
- *Características estruturais* – que representam a estrutura do e-mail, incluindo proporções dos caracteres espaço em branco, pontuação, letra maiúscula, letra minúscula e total de caracteres no e-mail.

Cada característica é representada no seguinte formato: **F99999,** onde **F** indica o tipo de característica - **W** para palavra, **C** para caractere e **S** para uma característica do tipo estrutural. O número 99999 indica o número/índice da característica. Assim, cada e-mail é representado como uma linha estruturada da seguinte forma, Figura 5.1:

dd/mm/aaaa, classe, {característica: ocorrência}

Figura 5.1. Formato das amostras que compõem a base de dados ECUE.

onde, **dd/mm/aaaa** representa a data do e-mail; ***classe***, representa um rótulo que identifica a amostra como *spam* ou não (*nonspam*); e ***{característica: ocorrência}*** representa a característica em valores binários, sendo o valor 1 (um) usado para indicar a presença e o valor 0 (zero) para indicar a ausência da característica na amostra.

A Tabela 5.1 apresenta a distribuição dos dados da base de treino ECUE, composta por 1000 amostras, sendo 500 amostras do tipo *spam* e 500 amostras do tipo *nonspam* (e-mails legítimos). Esse conjunto de amostras será usado para treinar o classificador SVM (domínio fonte).

A Tabela 5.2 apresenta a distribuição dos dados da base de teste ECUE, composta por 10.865 amostras, sendo 9.456 amostras do tipo *spam* (87% das amostras) e 1.409 amostras do tipo *nonspam.* Esse conjunto de amostras será usado como base de teste para avaliar o desempenho do classificador (domínio alvo).

Tabela 5.1. Distribuição de dados da base de treino ECUE. O rótulo S/data indica que a amostra não tem data.

Mensagens	Domínio Fonte				Total
	2002		2003		
	Nov	Dez	Jan	S/ data	
Spam	96	367	37	0	500
Legítima	83	84	84	249	500
Total	**179**	**451**	**121**	**249**	**1.000**

Tabela 5.2. Distribuição de dados da base de teste ECUE.

Mensagens	Domínio alvo												Total
	2003											2004	
	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez	Jan	
Spam	142	391	405	459	406	476	582	1849	1746	1300	954	746	**9.456**
Legítima	151	56	144	234	128	19	30	182	123	113	99	130	**1.409**
Total	**293**	**447**	**549**	**693**	**534**	**495**	**612**	**2.031**	**1.869**	**1.413**	**1.053**	**876**	**10.865**

5.1.1.2. ETSS (Emerging Technologies and Security System)

A base de dados ETSS corresponde a uma coleção de *spam* e e-mails legítimos recebidos por oito indivíduos no período de Junho de 2012 a Outubro de 2013. Como na base de dados ECUE, a base ETSS foi dividida em dois conjuntos de amostras: bases de treino e teste. A base de treino corresponde à coleção de e-mails recebidos nos meses de Junho a Agosto 2012. A base de teste corresponde à coleção de e-mails recebidos nos meses de Setembro de 2012 a Outubro de 2013. Os e-mails legítimos incluem uma variedade de correspondências pessoais, acadêmicas e mensagens de listas de discussões.

O formato das amostras que compõem a base de dados ETSS segue o mesmo padrão da base de dados ECUE, exibido na Figura 5.1. Contudo, diferentemente da base ECUE, a base de dados ETSS é composta somente por características extraídas da estrutura do corpo e assunto dos e-mails. O Apêndice A fornece os detalhes usados na construção da base ETSS, incluindo o método de extração de características e o processo de redução da dimensão do espaço de características adotados.

A Tabela 5.3 apresenta a distribuição dos dados da base de treino ETSS, composto por 1000 amostras, sendo 500 amostras do tipo *spam* e 500 amostras do tipo *nonspam* (e-mails legítimos). Esse conjunto de amostras será usado para treinar o classificador SVM (domínio fonte).

Tabela 5.3. Distribuição de dados da base de treino ETSS.

Mensagem	Domínio Fonte			Total
	2012			
	Jun	Jul	Ago	
Spam	167	167	166	**500**
Nonspam	167	167	166	**500**
Total	**334**	**334**	**332**	**1.000**

A Tabela 5.4 apresenta a distribuição dos dados da base de teste ETSS, composto por 111.280 amostras, sendo 88.997 amostras do tipo *spam* (80% das amostras) e 22.283 amostras do tipo *nonspam.* Esse conjunto de amostras será usado como base de teste para avaliar o desempenho do classificador (domínio alvo).

Tabela 5.4. Distribuição de dados da base de teste ETSS.

Mensagem	Domínio Alvo														Total
	2012				2013										
	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	
Spam	4.669	5.427	4.878	4.229	10.281	4.737	6.511	4.039	7.954	3.741	11.016	8.700	7.476	5.293	**88.997**
Nonspam	1.350	1.715	1.474	1.258	1.282	1.497	1.608	1.915	2.007	1.676	1.910	1.712	1.423	1.454	**22.283**
Total	**6.019**	**7.142**	**6.352**	**5.487**	**11.563**	**6.234**	**8.119**	**5.954**	**9.961**	**5.417**	**12.926**	**10.412**	**8.899**	**6.747**	**111.280**

5.1.2. Medidas de Desempenho

Para o problema de deteção de *spam*, a classificação de e-mails legítimos como *spam* (falso positivo - FP) é mais grave do que ter mensagens com conteúdo *spam* classificadas como não-*spam* (falso negativo - FN), visto que se o filtro de *spam* cometer algum erro, o usuário poderá corrigi-lo e automaticamente o filtro aprenderá os erros cometidos.

A maioria das aplicações sobre detecção de *spam* empregam como medidas de desempenho a taxa de precisão (ou erro) e a revocação - percentual de amostras positivas classificadas corretamente. Em alguns casos, é incorporado um peso maior à taxa de falso-positivos do que à taxa de falso-negativos para efeito de cálculo de taxa de precisão [2], [4], [33], [59].

Esta tese utiliza como métrica de desempenho a taxa de erro considerando as taxas de FP e FN obtidas para avaliar as séries de experimentos apresentadas a seguir. O

objetivo é considerar tanto a taxa de FP quanto de FN como graves, visto que o equilíbrio entre as duas demonstrará um aprendizado homogêneo entre e-mails *spam* e legítimos. Para tanto, é calculado o total de amostras classificadas erroneamente (FP + FN) pelo total de amostras que compõem uma janela de tempo.

A matriz de confusão é usada para aplicar a medida de desempenho escolhida. A referida matriz permite a visualização do desempenho de um algoritmo e é tipicamente usada em experimentos com aprendizagem de máquina. Cada coluna da matriz de confusão representa a classificação da amostra atribuída pelo algoritmo de aprendizagem, enquanto cada linha representa a classe real da amostra.

A Tabela 5.5 apresenta a matriz de confusão, composta pelas seguintes legendas: TP (Verdadeiro Positivo) – número de spams devidamente classificados como *spam*; FP (Falso Positivo) – número de e-mails legítimos classificados como *spam*; FN (Falso Negativo) – número de spams classificados como legítimo e TN (Verdadeiro Negativo) – número de e-mails legítimos devidamente classificados como legítimos.

Tabela 5.5. Matriz de confusão

Classe Atual	Predição da Classe	
	Classe Positiva (Spam)	**Classe Negativa (Legítima)**
Classe Positiva (Spam)	**TP**	**FN**
Classe Negativa (Legítima)	**FP**	**TN**

Para avaliar o período em que um classificador mantém seu conhecimento válido, a taxa de erro é utilizada como métrica, conforme justificado anteriormente, para calcular o desempenho de um modelo treinado em um domínio fonte e testado em um domínio alvo. A Equação 5.1 define a taxa de erro baseada nas legendas da matriz de confusão:

$$\textit{Tx erro} = \frac{FP+FN}{\textit{Total instâncias}} \qquad \textbf{(Equação 5.1)}$$

5.1.3. Classificador

O classificador escolhido nesta tese foi SVM por ser um método originalmente definido para resolver problemas de classificação binária, como a classificação de e-mail em *spam* ou legítimo. Além disso, SVM é um classificador estável, isto é, pequenas mudanças nos

dados não afetam significativamente o desempenho de SVM. Essa característica é importante para reduzir a possibilidade de falsa deteção de mudança em problemas com ambientes dinâmicos.

Dois parâmetros iniciais precisam ser definidos pelo usuário para SVM, o tipo de função kernel e o parâmetro de regularização *C*. Dependendo da função kernel escolhida, outros parâmetros devem ser definidos, como por exemplo, o grau do polinômio quando o kernel polinomial é adotado. A base de treino foi utilizada para a definição desses parâmetros, empregando validação cruzada usando 10 partições (*folds*). A validação cruzada é uma técnica para avaliar a capacidade de generalização de um modelo, a partir de um conjunto de dados. Esta técnica particiona o conjunto de dados em *k* subconjuntos mutualmente exclusivos, e posteriormente, utiliza *k-1* subconjuntos para treinamento, enquanto a partição restante, chamada de base de validação, é usada para estimação dos parâmetros do modelo gerado no treinamento. Esse processo é repetido *k* vezes, sendo que a cada nova iteração, uma nova partição é escolhida como base de validação.

Esse procedimento, incluindo a definição de parâmetros para a base ETSS está descrito em detalhes no apêndice A. Os parâmetros definidos para a base ECUE foram os mesmos apresentados pelos autores da base ECUE, em Delany et al. [12].

Para as bases de dados ECUE e ETSS, os melhores resultados obtidos foram: kernel RBF (*Radial Basis Function*), fator de penalização C=5 com γ=0.01, para base de dados ECUE; e *kernel* RBF (*Radial Basis Function*), fator de penalização C=100 com γ=0.1, para base de dados ETSS.

5.2. Experimentos

Esta Seção apresenta os resultados de três conjuntos de experimentos. O primeiro conjunto de experimentos investiga quão estável um modelo de classificação pode se manter ao longo do tempo sem atualização. Os outros dois conjuntos de experimentos avaliam o comportamento da evolução de características com foco em duas questões distintas: a primeira investiga a evolução das características considerando a relevância em função da frequência da característica em determinada janela de tempo; e a segunda pondera a evolução das características observando o nível de similaridade entre as características do domínio de treino e as características que descrevem as novas amostras (domínio alvo).

5.2.1. Avaliação do Comportamento Estático e da Evolução das Características

Os experimentos descritos nesta seção têm como objetivo determinar o período de tempo em que o modelo de classificação gerado permanece eficiente quando testado em um domínio alvo.

Em particular, duas questões serão investigadas:

1. Qual o desempenho do classificador considerando a taxa de erro nos meses seguintes à construção do modelo de classificação?
2. Qual a relação entre a taxa de erro do classificador e a evolução das características do problema?

Dois conjuntos de experimentos serão realizados utilizando a base de dados ECUE. No primeiro, o modelo de classificação é construído durante a fase de treino (3 meses) e avaliado de forma estática nos dados de teste (12 meses). No segundo, o modelo de classificação é construído e testado mensalmente nos dados que compõem as janelas de tempo do domínio alvo.

Normalmente, em problemas de classificação, o conjunto de dados apresenta uma quantidade de características muito elevada. As técnicas de seleção de características buscam identificar as características mais relevantes, e assim, reduzir a dimensão do espaço de entrada. Neste experimento, a estratégia de seleção de características utilizada foi Ganho de Informação (do inglês *Information Gain* – IG) [82].

5.2.1.1. Experimentos

Os experimentos são divididos em duas séries de seleção de características:

a) ***Série 1 - Information Gain somente no Treino***

Essa primeira série foi aplicada com o objetivo de reduzir a dimensão do espaço de características e obter características mais relevantes para o problema, como normalmente é feito em problemas estáticos, ou seja, o espaço de características reduzido é definido na base de treino. As amostras de teste são representadas pelo mesmo espaço de características definido no treino. Nessa primeira etapa, o classificador SVM é validado e treinado sobre o domínio fonte (base de treino), por meio da estratégia de

validação cruzada e, em seguida, avaliado no domínio alvo (base de teste).

Para a redução da base de dados ECUE, foram escolhidas as 700 características mais relevantes. De acordo com Delany et al. [12], essa é a quantidade de característica que melhor descreve o problema de deteção de *spam* para o conjunto de dados ECUE.

b) Série 2 - Information Gain no Teste

A segunda série de seleção de características visa avaliar o comportamento dinâmico das características. Para isso, o método IG é aplicado em cada janela de tempo mensal do domínio alvo. O classificador é treinado e testado em cada uma dessas janelas de tempo usando validação cruzada com 10 *folds*.

Nesta etapa de experimentos, o objetivo é demonstrar que as características relevantes encontradas nas janelas de tempo do domínio alvo são diferentes das características relevantes identificadas no domínio fonte.

5.2.1.2. Análise dos Resultados

A análise dos resultados inicia pela avaliação da eficiência do modelo de classificação durante a fase de teste (Fev/2003 – Jan/2004), isto é, série 1 de experimentos. A Figura 5.2 apresenta o desempenho do classificador considerando a taxa de erro de classificação obtida por meio da Equação 4.1. Como pode ser observado na Figura 5.2, o classificador apresenta um comportamento esperado, ou seja, um aumento gradativo e ascendente da taxa de erro durante os meses analisados. Contudo, algumas observações requerem uma análise mais cuidadosa. Por exemplo, por que o mês de Fevereiro, mês seguinte ao espaço de treino, apresenta uma taxa de erro alta de 3,07%, enquanto o mês de Abril apresenta uma taxa de erro de 1.09%?

Além disso, a Figura 5.2 mostra que a oscilação na taxa de erro da base de dados ECUE alcança 22.03% em Janeiro de 2004. Esse comportamento leva a outro questionamento: as amostras que compõem o espaço de treino utilizado para gerar o modelo de classificação não têm características representativas para o problema?

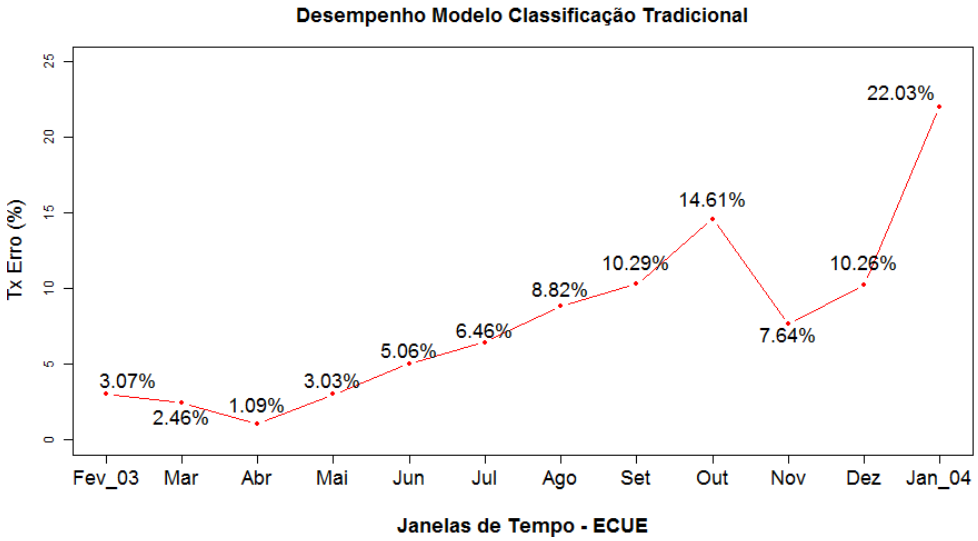

Figura 5.2. Taxa de erro obtida usando um modelo de classificação estático sobre a base de dados ECUE - Série 1

Para investigar tais questões e determinar o período de tempo em que o modelo de classificação gerado permanece eficiente, a segunda série de experimentos foi realizada, na qual o modelo de classificação é construído e testado mensalmente durante o período de teste. É importante destacar que a seleção de características também é realizada mensalmente nestes experimentos.

A Figura 5.3 exibe a comparação entre as taxas de erro obtidas pelo modelo de classificação construído a partir de características selecionadas durante três meses de treino (experimentos denominados de Série 1) e as taxas de erro obtidas por modelos de classificação mensais (Série 2).

Comparando as taxas de erros de ambas as séries, é possível observar que a construção de modelo de classificação a partir da seleção de características mensais é sempre mais eficiente que o modelo "estático" usado na primeira série de experimentos. Na Figura 5.3, em todos os casos, a taxa de erro não foi superior a 2%. A média de erro na Série 1 foi de 7,90%, enquanto na Série 2 foi de 0,98%. Além disso, a taxa de erro de 0,34% obtida no mês fevereiro (Série 2) é bem inferior à taxa obtida pelo modelo estático (Série 1) que foi de 3,07%. Isso demonstra que, primeiro, as características são representativas para o problema e, segundo, que ocorreu uma evolução abrupta nas características que compõem o domínio de treino e o mês de Fevereiro.

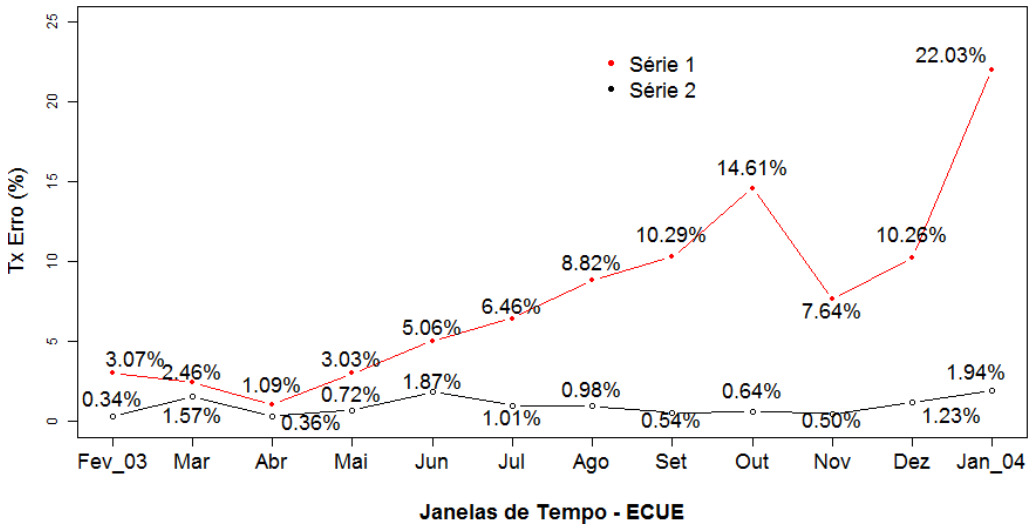

Figura 5.3. Taxas de erros obtidas usando um modelo de classificação estático (Série 1) e um modelo atualizado mensalmente (Série 2).

É importante ressaltar que a variabilidade da taxa de erro nos experimentos da Série 2 é decorrente da distribuição das amostras nos domínios de teste. Como mostrado na Tabela 5.2, os meses de Março, Julho, Agosto e Dezembro apresentam poucas amostras da classe não *spam*. A baixa representatividade desta classe prejudica uma distribuição balanceada de amostras para aplicar estratégia de validação cruzada. Por exemplo, a janela de tempo do mês de Julho apresenta somente 19 amostras do tipo *nonspam*. Assim, ao aplicar a estratégia de validação cruzada com 10 *folds*, alguns *folds* foram compostos por menos de 2 amostras dessa classe. Esta situação compromete o aprendizado do classificador, ocasionando maiores taxas de erro.

A partir das observações realizadas nesses experimentos iniciais, surge mais um questionamento: (1) existe uma relação direta entre evolução das características e taxa de erro do classificador? Para investigar esta relação, uma análise foi feita sobre os vetores de características que representam os domínios de treino e teste. Essa análise teve como objetivo levantar o percentual de características presentes no domínio de teste, porém, ausentes no domínio de treino.

A Figura 5.4 apresenta o percentual de características que estão ausentes no domínio de treino, considerando janelas de tempo mensais, em relação ao domínio de teste. Os resultados mostram que logo no primeiro mês de teste, Figura 5.4 (fevereiro de 2003), o percentual de características ausentes é de 43,43%. Essa evolução das características tem um reflexo direto na taxa de erro do classificador. Considerando os resultados exibidos na Figura 5.2, a taxa de erro apresentada pelo classificador neste mês

foi de 3.07%. Entretanto, esse comportamento não é linear. O mês de abril apresenta uma evolução de característica maior (55% de características ausentes) e uma taxa de erro menor de 1.09%, quando comparado ao mês de fevereiro.

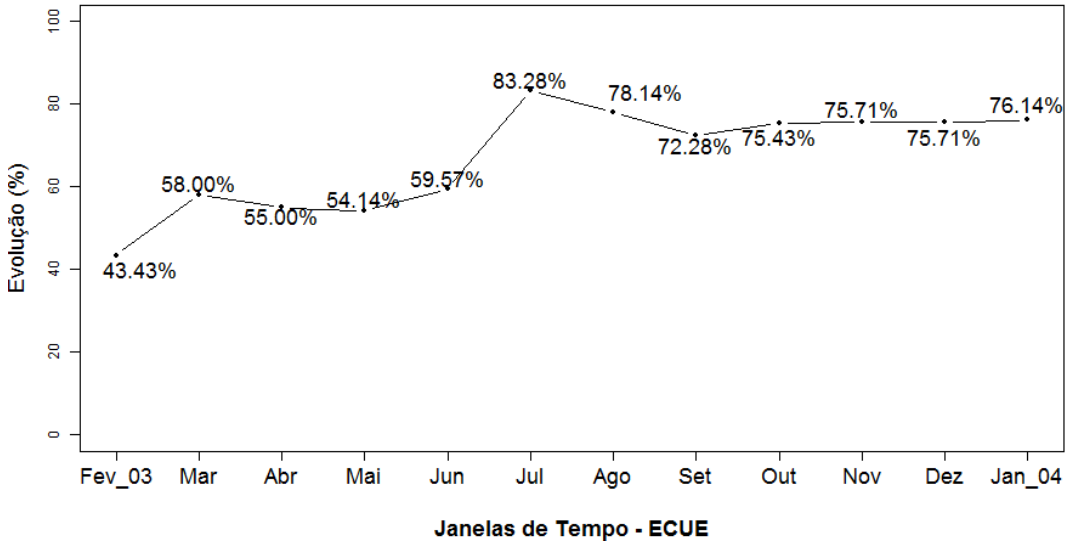

Figura 5.4. Percentual de características ausentes comparando os domínios fonte e alvo para a base de dados ECUE.

Logo, encontrar a validade de um modelo de classificação com base na evolução das características requer uma análise mais detalhada. Apesar de valores percentuais de presença ou ausência de características servirem como bons indicadores para atualizar o modelo de classificação, outros fatores devem ser levados em consideração como a relevância e a frequência das novas características.

5.2.1.3. Discussão

Os resultados apresentados demonstram que a relação da evolução de características de ataques com a taxa de erro do classificador é importante para identificar a efetividade de um modelo de classificação. Porém, a partir desses resultados novos questionamentos surgiram:

1. Os experimentos realizados com a Série 1 comprovam que ataques do tipo *spam*, por serem problemas dinâmicos, não podem ser tratados com modelos de classificação estáticos e que para obter melhores resultados é preciso acompanhar as mudanças que se apresentam nos vetores de atributos que caracterizam e-mails do tipo *spam*.

2. O processo de seleção de atributos realizado nesta tese foi baseado na adoção de da estratégia IG. Os resultados mostram que o melhoramento do processo de seleção de característica pode ser fundamental para a criação de modelos de classificação mais eficientes.

3. A análise temporal realizada mostrou que os valores percentuais de presença ou ausência de características servem como bons indicadores de deteção de mudança. Contudo, para atualizar o modelo de classificação, outros fatores devem ser levados em consideração como a relevância e a frequência das novas características.

Com o propósito de atender os questionamentos quanto à relevância e à frequência das características, esta tese apresenta duas novas estratégias: a primeira, baseada no histórico da relevância de características, denominada HFS (*Historical-based Feature Selection*) é apresentada na seção seguinte; e a segunda, baseada na medida de similaridade das caracerísticas, denominada SFS (*Similarity-based Features Selection*), é descrita na Seção 4.2.3.

5.2.2. HFS (*Historical-based Feature Selection*)

Esta seção apresenta a avaliação da estratégia de seleção HFS proposta na Seção 3.4.1. Para tratar o *concept drift,* o HFS considera a relação histórica da frequência das características para efetuar transferência de aprendizado e manter o domínio fonte sempre atualizado.

Para realizar a avaliação de desempenho do HFS foi necessário fazer uma normalização na base de dados investigada (ECUE) observando a distribuição das amostras. Cada janela de tempo possui características com frequências variando de acordo com a respectiva quantidade de amostras. Por exemplo, a distribuição de dados exibida na Tabela 5.2 mostra uma variação na quantidade de amostras mensais que tem influência na relevância das características, visto que esta relevância é obtida a partir da presença das características no total das amostras.

As Figura 5.5 e 5.6 mostram os vetores de características obtidos em duas janelas de tempo diferentes. A primeira característica (W123987), de ambas as janelas de tempo apresenta relevância máxima, ou seja, esta característica está presente em todas as amostras das janelas de dados. Neste caso, apesar da mesma característica apresentar

frequências diferentes, 293 na Figura 5.5 e 447 na Figura 5.6, a relevância da característica deveria ser a mesma.

	W123987	W789654	W234543	W098034	W239074	...	W999999	CLASS
1	1	0	1	0	1		0	spam
2	1	1	1	0	1		0	spam
3	1	0	0	1	0		0	spam
4	1	1	1	0	0		0	spam
5	1	1	0	0	0		0	spam
	1	1	1	0	1		0	spam
	1	0	0	0	1		0	ham
...	...	...	...	...	...		...	...
	1	0	1	0	0		0	ham
	1	1	0	1	0		0	ham
	1	0	1	0	1		0	spam
	1	0	1	0	0		0	spam
	1	1	1	0	1		0	spam
293	293	289	220	190	150		0	

Figura 5.5. Vetor de características que descreve a janela de tempo de fevereiro composta por 293 e-mail.

	W123987	W789654	W234143	W098234	W249074	...	W999999	CLASS
1	1	0	1	0	1		0	spam
2	1	1	1	0	1		0	spam
3	1	0	0	1	0		0	spam
4	1	1	1	0	0		0	spam
5	1	1	0	0	0		0	spam
	1	1	1	0	1		0	spam
	1	0	0	0	1		0	ham
...	...	...	...	...	...		...	...
	1	0	1	0	0		0	ham
	1	1	0	1	0		0	ham
	1	0	1	0	1		0	spam
	1	0	1	0	0		0	spam
	1	1	1	0	1		0	spam
447	447	389	220	190	150		0	

Figura 5.6. Vetor de características que descreve a janela de tempo de março composta por 447 e-mail.

Para tratar este problema, um procedimento de normalização proposto por Aksoye (2000) é adotado, como segue: dado um limite inferior $\boldsymbol{l}$, e um limite superior $\boldsymbol{u}$, para uma característica $\boldsymbol{x}$, a relevância de $\boldsymbol{x}$ resulta em $\bar{x}$, sendo um intervalo entre 0 e 1, conforme Equação 5.2:

$$\bar{\boldsymbol{x}} = \frac{\boldsymbol{x}-\boldsymbol{l}}{\boldsymbol{u}-\boldsymbol{l}} \qquad \textbf{(Equação 5.2)}$$

Por exemplo, para a primeira característica W123987, a relevância dessa característica é obtida da seguinte forma:

$$\bar{x}_{Figura\ 5.5} = \frac{293-0}{293-0} = \bar{x} = 1 \quad \text{e}$$

$$\bar{x}_{Figura\ 5.6} = \frac{447-0}{447-0} = \bar{x} = 1$$

Enquanto a relevância da terceira característica W234143 é 0.75 na janela de fevereiro e de 0.38 na janela de março.

$$\bar{x}_{Figura\ 5.5} = \frac{220-0}{293-0} = \bar{x} = 0.75 \quad \text{e}$$

$$\bar{x}_{Figura\ 5.6} = \frac{220-0}{447-0} = \bar{x} = 0.49$$

5.2.2.1. Experimentos

A análise do HFS, por ser baseada no histórico das características, requer uma sequência de valores obtidos a partir de repetidas medições no tempo (série temporal). A maneira mais fácil de realizar essa análise é definir uma janela deslizante fixa. Embora a implementação deste modelo seja simples, é propenso a erros caso a escolha da largura da janela não seja adequada. Janelas muito estreitas irão produzir representações muito precisas do estado atual, mas são fortemente afetadas por dados ruidosos, enquanto as janelas muito amplas resultam em resultados mais estáveis, mas igualmente imprecisos, devido aos efeitos do *concept drift* [83].

Para definir o tamanho da janela deslizante fixa, esta Seção apresenta duas séries de experimentos:

a) Série 3 – HFS em janelas de tempo trimestrais

Esta série de experimentos aplica a estratégia HFS em janelas de tempo trimestral. A ideia é avaliar o desempenho do HFS considerando exatamente a mesma janela de tempo usada para gerar o modelo de classificação (treino), a qual é formada por três meses, conforme Tabela 5.1.

b) Série 4 – HFS em janelas de tempo mensais

Esta série de experimentos aplica a estratégia HFS em janelas de tempo mensais.

A escolha de janelas de tempo mensal tem objetivo de produzir representações mais precisas do estado atual e, consequentemente, minimizar as taxas de erros.

5.2.2.2. Análise dos Resultados

A Figura 5.7 apresenta a taxa de erro obtida pelo classificador quando a estratégia HFS foi aplicada considerando janelas de tempo trimestrais (Série 3). Os resultados obtidos mostram que o uso da estratégia HFS melhora o desempenho do classificador consideravelmente quando comparada ao modelo estático, Série 1. Logo após a primeira atualização do modelo (mês de maio), a redução na taxa de erro é claramente percebida. A taxa de erro em maio de 3.03% é reduzida com a estratégia HFS para 0.43%. Esse comportamento se manteve para as demais janelas de tempo, obtendo taxas de erro menores, atingindo em janeiro de 2004, uma redução de 22.03% para 1.48%. Assim, essa considerável redução nas taxas de erro nas janelas de tempo trimestral resultou na redução da taxa de erro média de 7.90% (Série 1) para 1.36% (Série 3).

É importante salientar que o domínio fonte não sofreu nenhum acréscimo na quantidade de amostras, ou seja, foram mantidas as 1.000 amostras (500 *spams* e 500 *nonspam*) iniciais, apesar da estratégia HFS prever inclusão de amostras na base de treinamento. Esse fato pode ser justificado da seguinte forma: HFS faz uma busca em todo o vetor de características do domínio fonte para verificar se as características mais frequentes no domínio alvo estão também presentes no domínio fonte, mesmo que não sejam as características mais frequentes no domínio fonte. Essa ação é importante para o processo de análise de características, pois possibilita a observação de características recorrentes. Além disso, tal medida torna o domínio fonte menos tendencioso para gerar o modelo de classificação. Por fim, possibilita também demonstrar que com o passar do tempo características que eram relevantes deixam de ser relevantes, ou que características apresentam recorrência. Por outro lado, no caso da base ECUE, todas as características identificadas como relevantes no domínio alvo, também estavam presentes no domínio fonte. Portanto, na prática, ocorreu uma manipulação no vetor de características sem a necessidade de manipular amostras, isto é, características inativas foram ativadas e vice-versa.

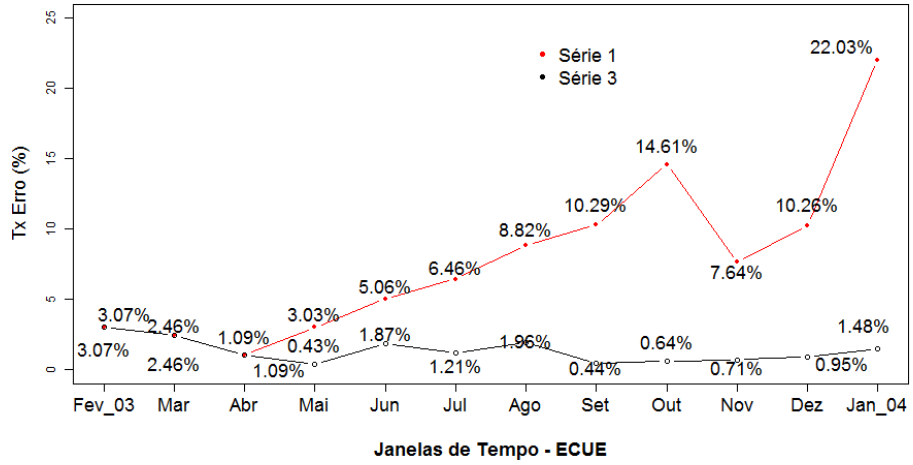

Figura 5.7. Percentuais de taxa de erro obtidos nas Séries de Experimentos 1 e 3 sobre a base de dados ECUE.

Com o objetivo de minimizar a taxa de erro nas janelas de tempo mais próximas do período de treino, a estratégia HFS é empregada em janelas de tempo mensais (Série 4).

Os resultados obtidos com janelas de tempo mensais, exibidos na Figura 5.8, mostram uma pequena melhora no desempenho do modelo de classificação sobre alguns meses. Por exemplo, a taxa de erro da janela de tempo de março obteve uma redução de 0.67, enquanto nas demais janelas de tempo, as taxas de erros apresentaram valores similares aos obtidos empregando janelas de dados trimestrais.

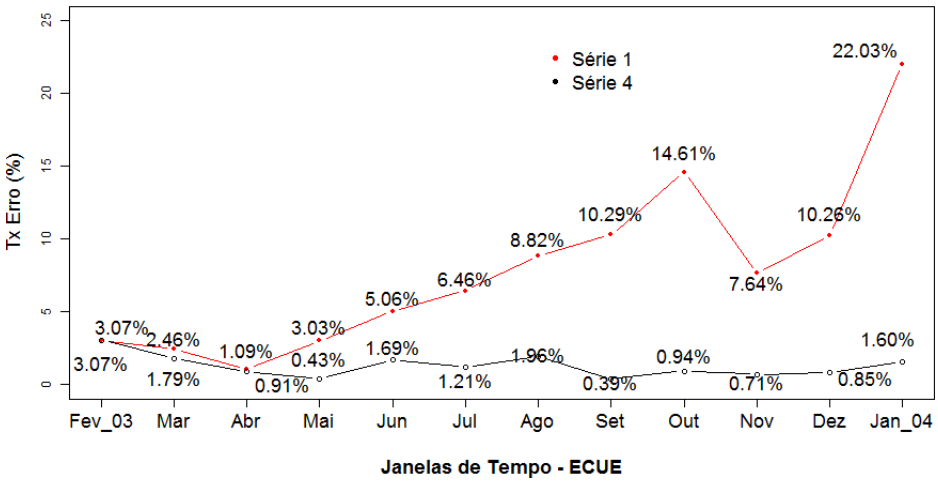

Figura 5.8. Percentuais de taxa de erro obtidos nas Séries de Experimentos 1 e 4 sobre a base de dados ECUE.

A Figura 5.9. apresenta a taxa de erro obtida pelo classificador quando a estratégia HFS considera janelas de tempo trimestrais (Série 3) e mensais (Série 4). Os resultados obtidos mostram que o uso da estratégia HFS nas Séries 3 e 4 apresenta pequena diferença em ganhos de percentuais considerando as taxas de erros obtidas. Entretanto, a redução na taxa de erro não é suficiente para justificar a atualização mensal. Na Série 3 foram realizadas 3 atualizações do modelo de classificação, enquanto na Série 4 foram efetuadas 11 atualizações.

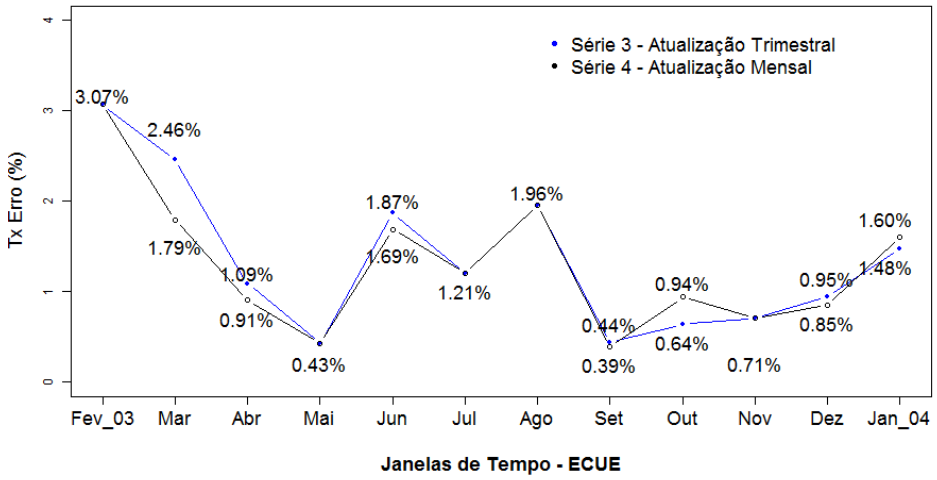

Figura 5.9. Comparação de percentuais de taxas de erro atingidos com a Série 3 e 4 sobre a base de dados ECUE.

O teste de Kruskal-Wallis [84], teste estatísticos não paramétrico, foi aplicado para avaliar a diferença entre os resultados apresentados nas Séries 3 e 4. Os resultados dos testes de Kruskal-Wallis mostram que as Séries 3 e 4 não são significativamente diferentes. Este resultado pode ser comprovado na Figura 5.10, que demonstra a dispersão dos dados com relação à taxa de erro dos modelos de classificação.

Os dados apresentados são dados assimétricos, conforme pode ser observado nos modelos estático (Série 1), dinâmico trimestral (Série 3) e dinâmico mensal (Série 4). Assimétricos porque as linhas medianas dentro das caixas não são equidistantes dos extremos (primeiro quartil e terceiro quartil). Esse comportamento da mediana possibilita identificar a distribuição dos dados. É importante observar que há menor dispersão dos dados no modelo 3 e também menor taxa de erro. Deve-se observar também que a taxa de erro apresenta uma diminuição gradativa com cada um dos modelos, assim como a

dispersão dos dados, embora a série 4 não tenha superado a série 3 em nenhum dos dois fatores.

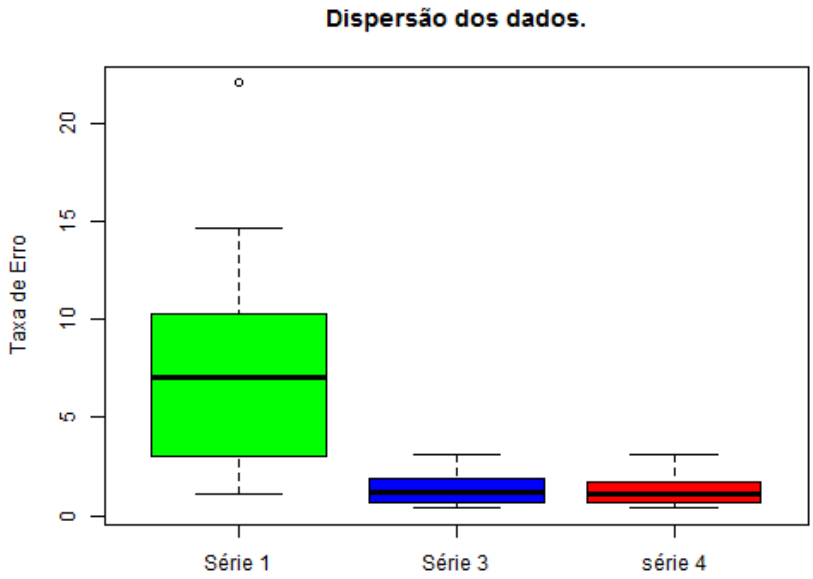

Figura 5.10. Dispersão dos dados das Séries de experimentos 1, 3 e 4 sobre a base de dados ECUE.

5.2.2.3. Discussão

Apesar dos resultados apresentados acima serem interessantes, duas questões precisam ser consideradas:

1. O cerne da estratégia HFS está baseado no histórico de características construído ao longo de um período de tempo. Durante a avaliação da estratégia HFS sobre a base de dados ECUE, se constatou uma evolução abrupta das características investigadas mesmo em janelas de tempo trimestrais e mensais por um período de 12 meses. Esse comportamento abrupto prejudicou o desempenho da estratégia HFS, ou seja, o tamanho das janelas de tempo estipuladas (mensal e trimestral) e a mudança repentina nas características que descrevem o domínio fonte e alvo dificultaram a construção eficaz e eficiente de um histórico das características.
2. O número de e-mails *spam* e legítimos na base de dados ECUE não é igual. Como o número de e-mails legítimos é consideravelmente menor que e-mails

spam, essa diferença prejudica a medida de desempenho em refletir uma visão realista da base de dados investigada, ou seja, a medida de desempenho está condicionada à distribuição de amostras de e-mails que compõem a base de dados ECUE.

Outra situação enfrentada devido à distribuição irregular da base de dados ECUE se apresenta durante a construção do modelo de classificação. O aprendizado do modelo de classificação por meio da estratégia de validação cruzada com 10 *folds* é prejudicado. Por exemplo, considerando-se as janelas de tempo de Julho e Agosto, cada uma das janelas apresenta, respectivamente, 19 e 30 amostras de e-mails legítimos. Esta quantidade distribuída em 10 partições, devido à estratégia de validação cruzada, originará, aproximadamente 1, 2 ou 3 amostras de e-mails legítimos para representar a classe em cada partição e treinar o modelo de classificação. Essa é uma situação delicada em aprendizagem de máquina, pois a pouca representatividade dos dados prejudica o aprendizado eficaz e eficiente.

Os resultados obtidos com a estratégia HFS e a distribuição da base de dados ECUE serviram como motivação para o desenvolvimento da estratégia SFS, descrita na próxima seção, e para a construção de uma nova base de dados balanceada, a base de dados ETSS.

5.2.3. Similarity based Features Selection (SFS)

Esta seção apresenta a avaliação da estratégia de seleção de características SFS, proposta na Seção 4.4.2. Para tratar o *concept drift,* o SFS aplica o cálculo do modelo vetorial para analisar a similaridade entre os vetores de características do domínio fonte e alvo. As amostras de e-mails que se distanciarem do domínio de treino, considerando um nível de similaridade, são armazenadas para gerar um novo conhecimento.

5.2.3.1. Experimentos

Os experimentos aplicando a estratégia SFS iniciam com a definição do limiar de deteção de mudança (*concept drift*). Tal limiar é usado para indicar quando o conhecimento adquirido deve ser empegado para gerar um novo modelo de classificação. Em função dos problemas obtidos na seção anterior com a base de dados ECUE, os experimentos de

avaliação da estratégia SFS serão conduzidos somente com a base de dados ETSS.

O limiar para detetar mudança é obtido considerando somente a classe positiva, isto é, a classe *spam*. Esta decisão foi motivada por duas questões. A primeira questão tem base na seguinte observação: para ludibriar os usuários, os atacantes empregam artifícios técnicos para aproximar e-mails *spam* de e-mails legítimos. Por este motivo, muitas das características *spam* estão contidas no conjunto de características legítimas, ou seja, a maioria das amostras de e-mails *spam* tem uma alta similaridade com as amostras legítimas, o que dificulta a identificação da evolução das características da classe *spam*. A segunda questão está relacionada com o modo com que as características são analisadas pelo SFS, o qual emprega a frequência da característica na amostra (TF) de e-mail e a importância da característica na coleção (IDF) de amostras de e-mails. Portanto, uma característica que assume valores pouco variados e que ocorre em muitas amostras de e-mails, tanto *spam* quanto legítimos, não é uma característica representativa para separar as classes. Por essas razões, esta tese irá utilizar somente amostras de e-mails da classe *spam,* visando identificar as características que evoluíram apenas nesta classe.

A investigação do limiar procede da seguinte forma: o cálculo da similaridade é aplicado sobre o domínio de teste em relação ao domínio de treino, gerando uma medida de similaridade para cada amostra de e-mail do domínio de teste. As medidas de similaridades resultantes são representadas em um gráfico de dispersão, que é usado para identificar o limiar. Esse procedimento de definição do limiar para deteção de mudança acontece uma única vez para a base de dados investigada.

Os experimentos realizados com o método SFS aplicado à base de dados ETSS compõem a quinta série de experimentos apresentada nesta tese.

a) Série 5 – SFS

A avaliação da estratégia SFS é baseada no valor do limiar definido e na observação da predição do modelo de classificação. Toda amostra de e-mail que apresentar uma similaridade inferior ao limiar definido e for rotulada como *spam* pelo classificador será armazenada para compor a nova base de conhecimento.

O processo de atualização ocorre adicionando as novas amostras, identificadas de acordo com a regra acima, ao domínio de treino, e retirando as amostras mais antigas para manter a mesma quantidade de amostras no treinamento. Após a atualização do domínio de treino é necessário definir o novo conjunto de características mais relevantes para a

construção do modelo de classificação.

5.2.3.2. Análise dos Resultados

A análise dos resultados inicia pela definição do limiar. A Figura 5.11 apresenta um gráfico de dispersão das amostras relativas às janelas de tempo de setembro a dezembro 2012. Os pontos representam as medidas de similaridades obtidas a partir das amostras de e-mail (domínio alvo) de *spam* (pontos azuis) e legítimos (pontos verdes).

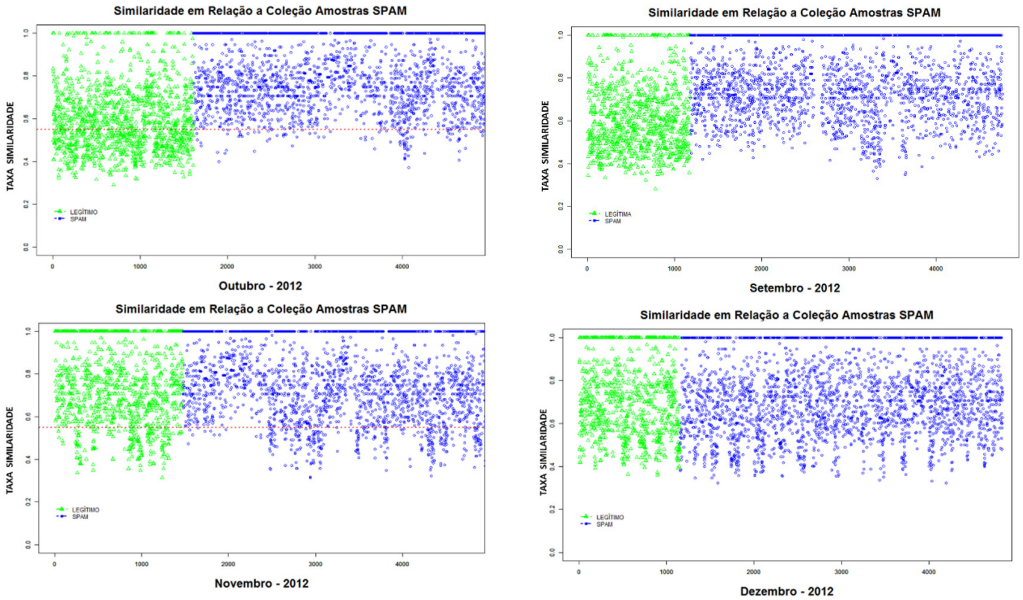

Figura 5.11. Dispersão das amostras de e-mail nas janelas de teste de setembro a dezembro de 2012, em relação as amostras de e-mail *spam* do domínio de treino por meio da medida de similaridade.

Observa-se que a maioria das amostras do domínio alvo (*spam* e legítimos) possuem taxas de similaridade superior a 0.40 para todas as janelas de tempo analisadas. As taxas de similaridades das amostras legítimas para os meses de setembro e outubro concentram-se entre 0.40 a 0.60. Enquanto, nesses mesmos meses, as taxas de similaridades das amostras de *spam* concentram-se entre 0.6 e 1.0. Nos dois meses seguintes, essas taxas são alteradas, as amostras legítimas e *spam* concentram-se acima de 0.5.

Uma análise mais detalhada mostra que no intervalo entre 0.5 e 0.55 concentram-se mais amostras legítimas que amostras *spam*. Por este motivo, para a base de dados ETSS, o limiar definido foi de 0.55, ou seja, toda amostra de e-mail com taxa de similaridade inferior a 0.55 é considerada uma amostra que apresenta alguma evolução

em suas características.

Os experimentos seguem com o uso da estratégia SFS sobre janelas de tempo mensais, considerando o limiar definido acima e a predição do modelo de classificação. As amostras com taxas de similaridade inferiores a 0.55 e rotuladas como sendo da classe *spam* pelo classificador serão armazenadas para serem usadas numa futura atualização do modelo de classificação. Essa abordagem garante duas situações: 1) identificando evolução de características - as amostras que foram rotuladas como *spam* e realmente são da classe *spam* (verdadeiro positivo) são amostras importantes para a transferência do conhecimento, pois apresentam evolução nas características; e 2) reforçando aprendizado a partir da predição incorreta - as amostras que foram incorretamente rotuladas como *spam* (falso negativo) também são transferidas para melhorar o aprendizado do próximo modelo de classificação.

A Figura 5.12 apresenta a taxa de erro obtida pelo classificador quando a estratégia SFS é aplicada considerando janelas de tempo mensais (Série 5). Os resultados obtidos mostram que o uso da estratégia SFS melhora o desempenho do classificador quando comparada ao modelo estático (Série 1).

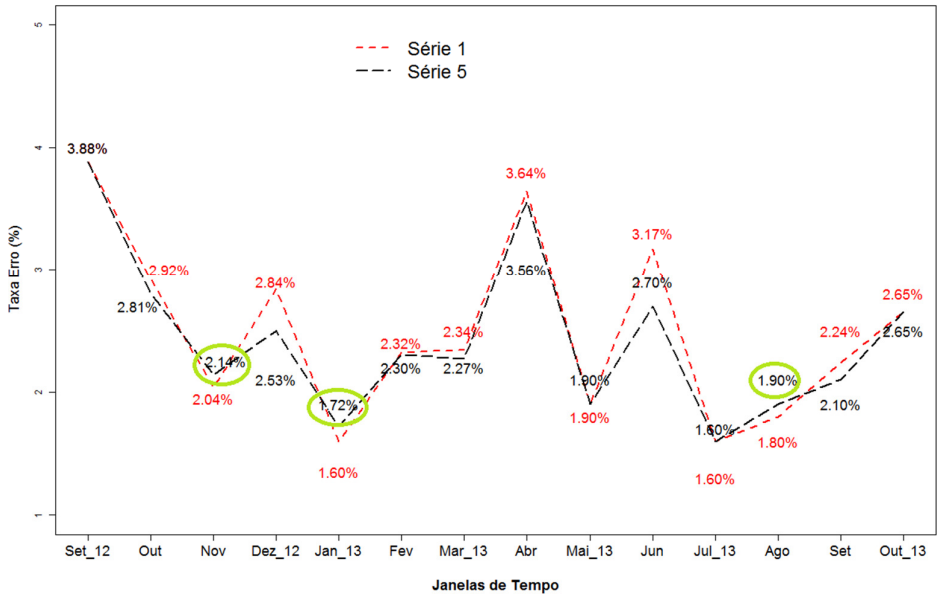

Figura 5.12. Desempenho do modelo de classificação aplicando a Série 1 (aprendizagem de máquina tradicional) e Série 5 (estratégia SFS) sobre a base de dados ETSS.

Os valores destacados por elipses na Figura 5.12 representam comportamento diferente dos demais valores. Para se explicar as três janelas de tempo (novembro de 2012, janeiro

e agosto de 2013) que apresentaram comportamento diferenciado das demais é preciso observar a taxa de similaridade. A Figura 5.13 exibe a comparação entre as taxas de similaridade das séries 1 e 5. A maioria das taxas de similaridades correspondentes à Série 5 apresentam aumento em relação as taxas de similaridade da Série 1. Entretanto, as três taxas de similaridade destacadas por elipses apresentam valores menores em relação à Série 1 e também correspondem às taxas de erro que apresentaram aumento com a Serie 5, comprovando que a hipótese de que sempre que a taxa de similaridade diminui ocorre um aumento na taxa de erro.

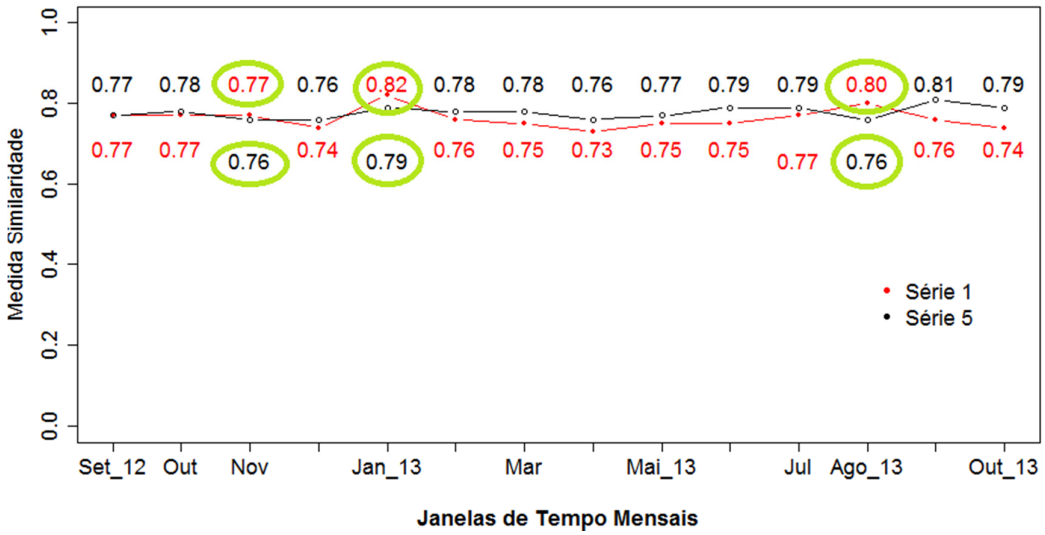

Figura 5.13. Comparação entre as medidas de similaridade resultantes para as Séries 1 e 5.

Apesar da análise dos resultados obtidos com a estratégia SFS sobre janelas de tempo mensais apresentar bons resultados e demonstrar a relação percentual de similaridade e taxa de erro do modelo de classificação, é necessário investigar o comportamento do SFS sobre janelas de tempo diárias.

As Figura 5.14, Figura **5.15** e 5.16 apresentam o desempenho do modelo de classificação exibindo as taxas de erro diárias para os meses de setembro a novembro de 2012. As taxas de erro média obtidas foram de 6.37%, 4.45% e 3.50% para os meses de setembro, outubro e novembro, respectivamente. Enquanto as taxas de erro médias obtidas utilizando janelas de tempo mensais foram de 3.88%, 2.81% e 2.14% para os mesmos meses. A análise das taxas de erro demonstra que a estratégia SFS apresenta um desempenho melhor quando aplicada sobre janelas de tempo mensais para a base de dados

investigada, o que também pode ser comprovado pela quantidade de atualizações efetuadas.

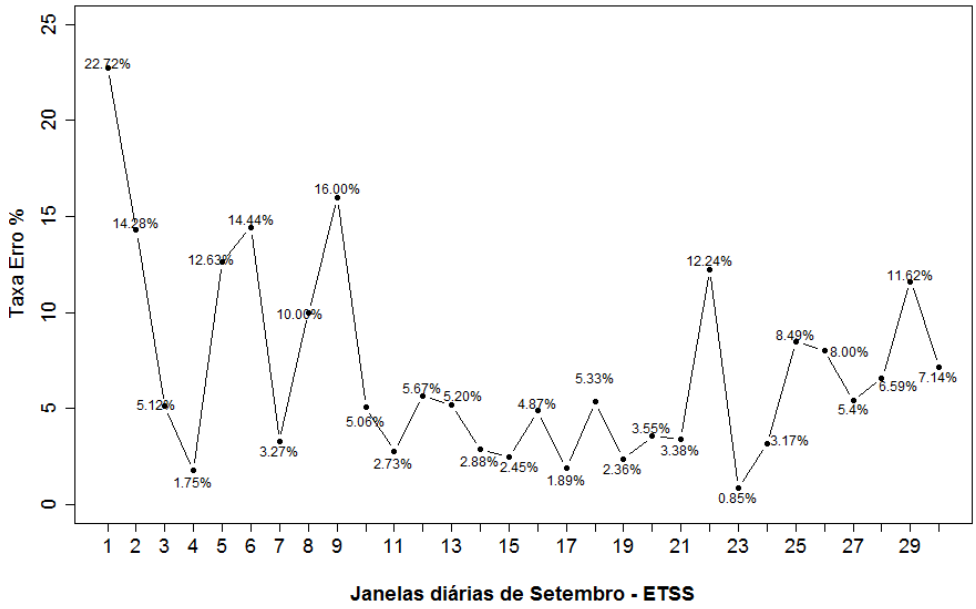

Figura 5.14. Desempenho do modelo de classificação sobre janelas de tempo diárias, com a Série 5 sobre base de dados ETSS – setembro 2012.

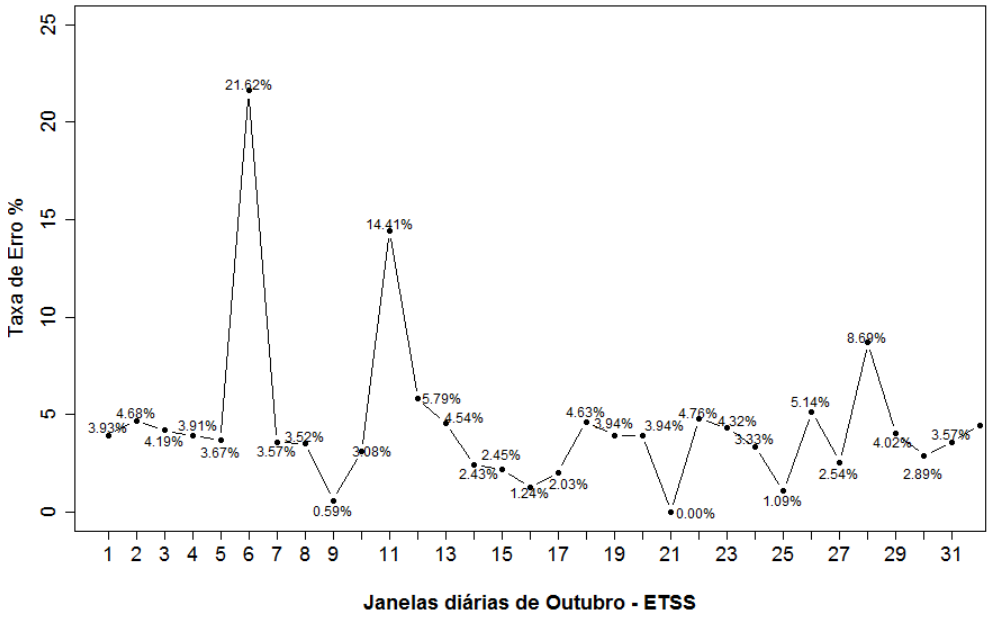

Figura 5.15. Desempenho do modelo de classificação sobre janelas de tempo diárias, com a Série 5 sobre base de dados ETSS – setembro 2012.

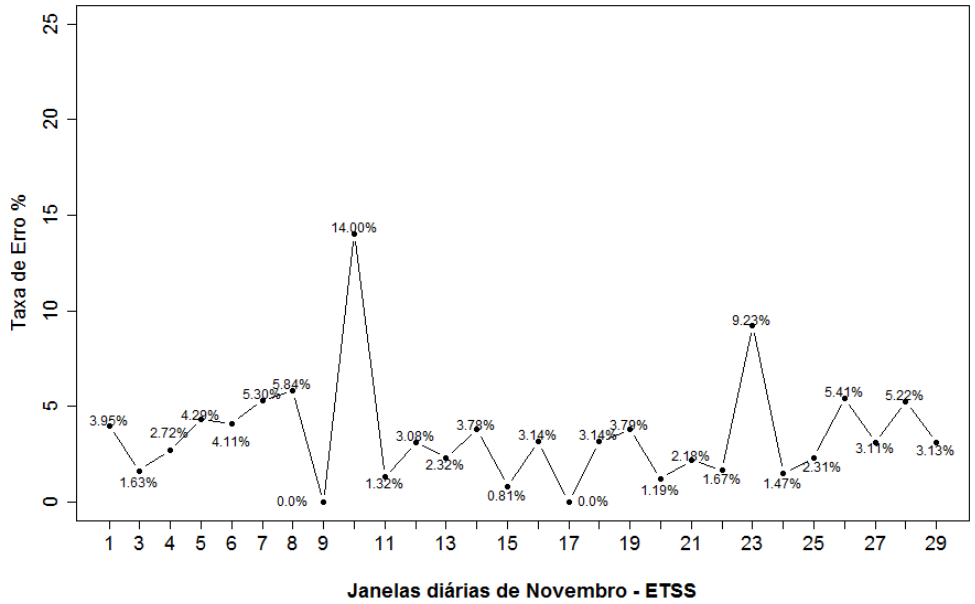

Figura 5.16. Desempenho do modelo de classificação sobre janelas de tempo diárias, com a Série 5 sobre base de dados ETSS – setembro 2012.

As Tabela 5.6, Tabela **5.7** e Tabela **5.8** apresentam o montante de atualizações efetuadas nas janelas de tempo diárias para os meses de setembro, outubro e novembro, respectivamente. As tabelas exibem ainda a quantidade de amostras de e-mails *spam* e

legítimos usada nas atualizações. Foram realizadas 19, 20 e 21 atualizações nos meses citados, totalizando 60 atualizações ao final dos três meses. Enquanto sobre janelas de tempo mensais foram efetuadas somente três atualizações.

Tabela 5.6**.** Quantidade de amostras usadas para atualizar modelo de classificação em janelas de tempo diária considerando classe *spam* e legítima – setembro de 2012.

Atualizações Diárias																				
Mensagem	**up1**	**up2**	**up3**	**up4**	**up5**	**up6**	**up7**	**up8**	**up9**	**up10**	**up11**	**up12**	**up13**	**up14**	**up15**	**up16**	**up17**	**up18**	**up19**	**Total**
Spam	3	7	4	1	6	2	10	0	6	1	8	9	5	45	11	4	10	2	16	150
Nonspam	6	2	3	1	4	2	4	5	0	2	1	7	1	2	3	4	8	10	5	70
Total	**9**	**9**	**7**	**2**	**10**	**4**	**14**	**5**	**6**	**3**	**9**	**16**	**6**	**47**	**14**	**8**	**18**	**12**	**21**	**220**

Tabela 5.7. Quantidade de amostras de e-mails usadas para atualizar modelo de classificação em janelas de tempo Diárias para classe *spam* e legítima – outubro de 2012

Atualizações Mensais																				
Mensagem	**up20**	**up21**	**up22**	**up23**	**up24**	**up25**	**up26**	**up27**	**up28**	**up29**	**up30**	**up31**	**up32**	**up33**	**up34**	**up35**	**up36**	**up37**	**up38**	**up39**	**Total**
Spam	7	6	3	4	4	4	3	14	1	6	6	23	12	4	3	1	0	6	4	5	116
Nonspam	5	7	3	5	4	5	3	5	2	11	4	3	3	9	8	6	3	7	3	6	102
Total	**12**	**13**	**6**	**9**	**8**	**9**	**6**	**19**	**3**	**17**	**10**	**26**	**15**	**13**	**11**	**7**	**3**	**13**	**7**	**11**	**218**

Tabela 5.8. Quantidade de amostras de e-mails usadas para atualizar modelo de classificação em janelas de tempo Diárias para classe *spam* e legítima – novembro de 2012.

Atualizações Mensais																					
Mensagem	**up40**	**up41**	**up42**	**up43**	**up44**	**up45**	**up46**	**up47**	**up48**	**up49**	**up50**	**up51**	**up52**	**up53**	**up54**	**up55**	**up56**	**up57**	**up58**	**up59**	**up60**	**Total**
Spam	6	7	0	3	2	3	0	1	11	1	3	10	0	3	4	8	7	5	3	3	4	84
Nonspam	5	0	1	4	3	10	5	0	2	1	2	4	3	7	0	2	10	5	2	5	0	71
Total	**11**	**7**	**1**	**7**	**5**	**13**	**5**	**1**	**13**	**2**	**5**	**14**	**3**	**10**	**4**	**10**	**17**	**10**	**5**	**8**	**4**	**155**

5.2.3.3. Discussão

A escolha do valor do limiar é uma tarefa muito difícil. A principal dificuldade está na alta dimensionalidade do espaço de características que descrevem e-mails legítimos e *spam*. Observar a taxa de similaridade das amostras do domínio de interesse considerando somente a classe *spam* minimizou esta questão e auxiliou a identificar o limiar. Ao longo das janelas de tempo analisadas, foi possível constatar o distanciamento das taxas de similaridades das amostras de e-mails *spam* de maneira gradativa em relação à primeira

janela de tempo. Esta observação direcionou-nos a uma análise mais aprofundada e possibilitou a seguinte conclusão: para a base de dados investigada, um limiar abaixo de 0.55 aborda um número de características mais representativas da classe *spam*, ou seja, mais relevantes para se identificar a evolução das características que descrevem a classe positiva, classe de interesse desta tese.

O critério de empregar limiar inferior a 0.55 e predição do modelo de classificação foi um adicional crucial para auxiliar na identificação das amostras de e-mail *spam* que apresentam evolução nas características, assim como identificar amostras classificadas erroneamente (legítimas classificadas como *spam*) que precisam ser inseridas ao novo conhecimento para reforçar o aprendizado do modelo de classificação.

A estratégia SFS respondeu positivamente às atualizações mensais efetuadas empregando a Série 5. Das 14 janelas de tempo, 11 apresentaram redução na taxa de erro e 3 apresentaram aumento determinado pela queda na similaridade durante a transferência de aprendizado para atualização do modelo de classificação. A queda na taxa de similaridade pode ter sido ocasionada por uma transferência negativa, ou seja, as amostras adicionadas não contribuíram para melhorar o desempenho do modelo de classificação. É necessário investigar com mais detalhes as amostras empregadas para transferência de conhecimento.

Observando uma granularidade diária em comparação a janelas de tempo mensais, a Série 5 foi novamente aplicada. Pode-se concluir que, para a distribuição da base de dados investigada, a granularidade diária não apresentou melhora no desempenho do modelo de classificação e necessitou de um montante de 19 a 20 atualizações por mês, totalizando sobre os três meses investigados 60 atualizações. Enquanto que em janelas de tempo mensais foram efetuadas apenas 3 atualizações. Porém, uma análise mais detalhada em cada janela de tempo diária seria importante, visto que ao longo dos três meses analisados a taxa de erro na média diminuiu gradativamente, começando com 6.37% para setembro, 4.45% para outubro e chegando a 3.40% em novembro. Considerando que no mês de outubro uma janela de tempo diária alcançou taxa de erro 0.00% e no último mês, novembro, duas janelas de tempo diárias alcançaram taxas de erro equivalentes a 0.00%. Uma análise mais aprofundada é necessária, pois talvez o critério apresentado para janelas de tempo mensais precise ser revisto para granularidades menores.

O próximo capítulo sintetiza as conclusões e contribuições desta tese, bem como discute vários possíveis caminhos para trabalhos futuros.

Capítulo 6
Conclusões e Trabalhos Futuros

Esta tese demonstrou com base no estado da arte que apesar dos esforços concentrados pelas diferentes abordagens empregadas para mitigar e cessar *spams*, as caixas de mensagens dos usuários ainda são lotadas por e-mails não solicitados ou indesejados. A principal dificuldade em identificar *spam* é devido à constante mudança das características que representam esse tipo de ataque. Os atacantes (s*pammers)* continuamente empregam diferentes artifícios técnicos que aproximam cada vez mais e-mail *spam* de e-mails legítimos, evitando que os filtros de deteção impeçam que tais mensagens indesejadas cheguem aos usuários.

6.1. Resumo dos Trabalhos Realizados

O desafio de monitorar ou detetar mudanças (*concetp drift*) torna os filtros de mensagens um problema incremental de aprendizagem de máquina. Esta tese abordou esse problema provendo a arquitetura de método de **D**eteção de mudança baseada na **E**volução das **C**aracterísticas de **A**taques (MECA) como principal contribuição.

MECA é um método que apresenta três etapas bem distintas e de funcionamento independentes: etapa de treinamento e geração do modelo de classificação, etapa de deteção de mudança e etapa de transferência de aprendizado.

Além da contribuição do método MECA, outra contribuição do trabalho são duas estratégias de deteção de mudança que investigam a evolução das características de ataques na presença de *concept drift* empregadas na etapa de deteção de mudança. As estratégias foram implementadas para manusear *concept drift* e transferir o novo conhecimento identificado a partir da evolução das características de ataque, com o objetivo de contornar o problema e manter o domínio de treino sempre atualizado se antecipando à degradação do modelo de classificação.

A primeira estratégia, denominada HFS (*Historical-based Features Selection*), apresenta uma seleção de características relevantes a partir da construção de um histórico das características ao longo do tempo. O critério de relevância das características é baseado na frequência com que essas características aparecem ou deixam de aparecer ao longo do tempo. Os resultados obtidos sobre janelas de tempo mensais mostraram que o HFS obteve um bom desempenho em relação à aprendizagem de máquina tradicional, chegando a reduzir a taxa de erro média do modelo de classificação com AM tradicional (Série 1) de 7.90% para 1.36% com HFS (Série 3). Com uma investigação mais minuciosa sobre janelas de tempo maiores (trimestrais) foi possível identificar que a alteração era mínima em relação às janelas de tempo mensais. Os resultados obtidos mostraram que a estratégia HFS necessita de um espaço de tempo maior para construir um histórico de características mais eficiente. A mudança abrupta das características apresentadas na base ECUE, corroborou para está conclusão.

Apesar dos resultados interessantes obtidos pela utilização da estratégia FHS, as análises realizadas nas bases de dados adotadas neste projeto de pesquisa mostraram que *spam* é um problema cujo tipo de mudança é abrupta (do inglês *sudden concept drift*). Problemas desta natureza exigem reações rápidas para contornar a necessidade de atualizações sobre o domínio de treino e não se enquadram em um monitoramento de mudança no padrão da distribuição dos dados que exija um espaço de tempo longo para uma análise eficiente da evolução das características.

A partir dessas conclusões, uma nova estratégia foi proposta com o objetivo de observar a evolução abrupta nas características de e-mails *spam*. A nova estratégia, denominada SFS (*Similarity-based Features Selection*), observa as características que descrevem o domínio fonte empregando uma medida de similaridade entre as amostras que compõem o domínio fonte e alvo. Esta medida de similaridade possibilitou identificar as alterações de acordo com a natureza dos dados apresentados, seja ela abrupta ou não.

Observando o comportamento das amostras de e-mails *spam* usando a medida de similaridade proposta foi possível identificar um limiar, para a base de dados utilizada, capaz de determinar o momento em que a evolução das características ocorre. Identificar esse limiar, mesmo que empiricamente, foi uma tarefa árdua. Uma análise na dispersão das medidas de similaridade das amostras do domínio de interesse, mostrou que inicialmente o intervalo para a maioria das amostras de e-mails *spam* se concentrava entre um espaço de 0.60 e 1.0, assim como o intervalo para amostras de e-mails legítimos se

concentrava entre 0.40 e 0.60. É importante observar que uma amostra atinge similaridade de 100% com o domínio de conhecimento inicial quando sua similaridade é igual a 1, ou seja, a escala que proporciona analisar essa medida de similaridade fica entre 0 (zero) e 1 (um). Uma investigação mais detalhada sobre as amostras demonstrou que limiar de 0.55 é adequado para separar as classes de dados *spam* e não *spam* (legítimas) para a base de dados ETSS.

A definição do limiar de deteção de mudança foi crucial para monitorar a evolução das características em paralelo com a tarefa de predição do modelo de classificação. A partir da definição do limiar, foi possível garantir a deteção de amostras de interesse tanto da classe *spam* quanto da classe legítima, para compor o novo conhecimento. Pois, toda amostra com medida de similaridade menor ou igual a 0.55 e rotulada como pertencente à classe *spam* era uma candidata a ser armazenada como novo conhecimento. Esse critério garantia que amostras *spam* com baixa medida de similaridade fossem adicionadas por apresentarem conjunto de características relevantes para o domínio de treino. Enquanto que amostras de e-mails legítimos com medida de similaridade menor e igual a 0.55 e rotuladas como pertencentes à classe *spam*, eram adicionadas ao novo conhecimento com o objetivo de reforçar o aprendizado do modelo de classificação quanto à predição errônea.

Para a etapa de transferência de aprendizado, os critérios de o que, como e quando transferir o novo conhecimento estão conexos com as estratégias propostas na etapa de detecção de mudança, ou seja, ao mesmo temo que detetava-se as mudanças ao longo do tempo, armazenava-se as amostras que compreendiam o novo conhecimento em um repositório denominado novo conhecimento. Entre as 14 janelas de tempo empregadas para avaliar a estratégia SFS, somente em três destas 14 janelas de tempo a transferência do aprendizado efetuada não melhorou o desempenho do modelo de classificação. Essas janelas necessitam uma investigação mais aprofundada para melhorar a transferência de aprendizado e pesquisar alternativas para contornar problemas de transferência de aprendizado negativo.

Outra contribuição desta tese foi uma base de dados construída especificamente para contornar algumas questões deficientes apresentadas pela base de dados ECUE, que foi inicialmente investigada nesta pesquisa. A base de dados ETSS foi construída seguindo os modelos ou etapas referenciadas pelo estado da arte e podem ser conferidas no Apêndice A. Trata-se de uma base de dados *spam* para aplicação de aprendizado de

máquina supervisionado, apresentando aproximadamente 110.000 amostras de e-mails *spam* e legítimos e um espaço de características próximo de 95.000 características.

6.2. Trabalhos Futuros

A arquitetura desenvolvida para o método MECA, nesta tese, possibilita o emprego de outras estratégias diferentes das que foram implementadas e abre, assim, um leque de possibilidades para pesquisas direcionadas a identificar a evolução de características de ataques sem considerar a degradação do modelo de classificação como medida para deteção de mudança.

Diante da definição do conceito de transferência de aprendizado e *concept drift,* a arquitetura MECA pode possibilitar não somente a deteção de um tipo de ataque, mas fazer deteção multi classe. Por exemplo, diante de um aprendizado de máquina supervisionado e um modelo de classificação que deteta *spam*, podem ser identificadas mensagens que contenham outro tipo de ataque, como *phishing*.

Com relação à construção de bases de dados fundamentadas na técnica *bag-of-words*, seria interessante investigar a construção de uma base de dados de mensagens spam e legitimas observando somente o assunto do e-mail. Dessa forma, a arquitetura MECA pode ser avaliada com ambas as estratégias propostas, HFS e SFS.

A estratégia HFS mostrou-se uma estratégia adequada para problemas que apresentam uma mudança gradual. Aplicar a arquitetura MECA empregando a estratégia HFS para avaliar seu comportamento diante de uma deteção de mudança que apresente característica gradual é outro aspecto a ser investigado a partir dos experimentos efetuados nesta tese.

A estratégia SFS (*Similarity based Features Selection*) foi avaliada somente sobre janelas de tempo mensais. Seria interessante avaliar o comportamento da estratégia SFS sobre janelas de tempo trimestrais, pois talvez, obtenha-se um desempenho próximo do atingindo em janelas de tempo mensais, porém com complexidade computacional menor.

Durante os experimentos foi observado que das 14 janelas de tempo que constituem o domínio de teste da base de dados ETSS, três janelas apresentaram comportamento abaixo do percentual de desempenho do modelo de classificação. Esse comportamento é típico de uma transferência de aprendizado negativo. Investigar estratégias para contornar a transferência de aprendizado negativo é outra direção em aberto para dar continuidade às pesquisas para minimizar a complexidade computacional e não efetuar atualizações

desnecessárias.

Esta tese atingiu com êxito a deteção de mudança virtual, ou seja, identificar o momento em que o espaço de distribuição de características sofreu mudanças. A partir desta deteção virtual foi possível atualizar o modelo de classificação antes que este sofresse degradação no desempenho da tarefa de classificação. Contudo, seria interessante investigar o momento em que o modelo de classificação perde o poder de generalização a partir da mudança detectada na distribuição dos dados, ou seja, a mudança real. Sendo esta outra questão em aberto que pode ter continuidade a partir desta tese.

Referências

[1] CERT.br, "Estatísticas de Notificações de Spam Reportadas ao CERT.br," *Cert.br*, 2014. [Online]. Available: http://www.cert.br/stats/spam/.

[2] C. Fung, "Collaborative Intrusion Detection Networks and Insider Attacks," *J. Wirel. Mob. Networks, Ubiquitous Comuting Dependable Appl.*, vol. 2, no. 1, pp. 63–74, 2011.

[3] A. Karimganame, J. Bourgeois, R. Bidou, and F. Spies, "A global security architecture for intrusion detection on computer networks," *Comput. Secur.*, vol. 27, no. 1–2, pp. 30–47, Mar. 2008.

[4] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Appl. Soft Comput.*, vol. 10, no. 1, pp. 1–35, Jan. 2010.

[5] E. Corchado, "Multiagent Systems for Network Intrusion Detection : A Review," *Security*, pp. 143–154, 2009.

[6] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," *Artif. Intell. Rev.*, vol. 29, no. 1, pp. 63–92, Jul. 2009.

[7] F. Fdez-Riverola, E. L. Iglesias, F. Díaz, J. R. Méndez, and J. M. Corchado, "Applying lazy learning algorithms to tackle concept drift in spam filtering," *Expert Syst. Appl.*, vol. 33, no. 1, pp. 36–48, Jul. 2007.

[8] G. Caruana and M. Li, "A survey of emerging approaches to spam filtering," *ACM Comput. Surv.*, vol. 44, no. 2, pp. 1–27, Feb. 2012.

[9] Y. Koren, "Collaborative filtering with temporal dynamics," *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '09*, p. 447, 2009.

[10] P. Medas, G. Castillo, and P. Rodrigues, "Learning with Drift Detection," *Adv. Artif. Intell. Proc. 17th Brazilian Symp. Artif. Intell. (SBIA 2004), Springer*, vol. 3171 of LN, pp. 286–295, 2004.

[11] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains : A Survey," vol. 10, pp. 1633–1685, 2009.

[12] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," *Knowledge-Based Syst.*, vol. 18, no. 4–5, pp. 187–195, Aug. 2005.

[13] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to Spam filtering," *Expert Syst. Appl.*, vol. 36, no. 7, pp. 10206–10222, Sep. 2009.

[14] M. Zi Hayat, J. Basiri, L. Seyedhossein, and A. Shakery, "Content-based concept drift detection for Email spam filtering," *2010 5th Int. Symp. Telecommun.*, pp. 531–536, Dec. 2010.

[15] B. Su and Y. Shen, "Modeling concept drift from the perspective of classifiers," *2008 IEEE Conf. Cybern. Intell. Syst.*, pp. 1055–1060, Sep. 2008.

[16] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, “An Evaluation of Machine Learning-based Methods for Detection of Phishing Sites,” *Mach. Learn.*

[17] H. Xiao, F. Hong, Z. Zhang, and J. Liao, “Intrusion Detection Using Ensemble of SVM Classifiers,” *Fourth Int. Conf. Fuzzy Syst. Knowl. Discov. (FSKD 2007)*, no. Fskd, pp. 45–49, 2007.

[18] E. Alpayd, *Introduction to Machine Learning, 2nd ed.*, vol. 19, no. 02. 2012, pp. 285–288.

[19] U. Alon, “How to choose a good scientific problem.,” *Mol. Cell*, vol. 35, no. 6, pp. 726–8, 2009.

[20] V. N. Vapnik, *Statistical Learning Theory*, vol. 2, no. 4. Wiley & Sons-Interscience, 1998, p. 736.

[21] S. Mukkamala, G. Janoski, and A. Sung, “Intrusion Detection : Support Vector Machines and Neural Networks,” 1998.

[22] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” *Proc. fifth Berkeley Symp.*, vol. 233, no. 233, pp. 281–297, 1967.

[23] V. Chandola, “Anomaly Detection : A Survey,” *Computing*, no. September, pp. 1–72, 2009.

[24] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Blackwell, 2004, p. 376.

[25] F. Roli, J. Kittler, and T. Windeatt, Eds., *Multiple Classifier Systems*, vol. 3077. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1–15.

[26] J. O. Ao, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A Survey on Concept Drift Adaptation ˜,” vol. 46, no. 4, 2014.

[27] I. Zliobaite, “Learning under concept drift: an overview,” *Tech. report, Vilnius Univ. 2009 Tech. Relat. areas, Appl.*, pp. 1–36, 2009.

[28] L. C. Molina, L. Belanche, and A. Nebot, “Feature selection algorithms: a survey and experimental evaluation,” in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, 2002, pp. 306–313.

[29] R. Elwell and R. Polikar, “Incremental learning of concept drift in nonstationary environments.,” *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–31, Oct. 2011.

[30] B. Kurlej and M. Wozniak, “Active learning approach to concept drift problem,” *Log. J. IGPL*, vol. 20, no. 3, pp. 550–559, Feb. 2011.

[31] A. Dries and R. Ulrich, “Adaptive Concept Drift Detection,” *SDM - SIAM, 2009*, pp. 233–244., 2009.

[32] L. I. Kuncheva, “Classifier Ensembles for Detecting Concept Change in Streaming Data : Overview and Perspectives,” *2nd Work. SUEMA*, no. July, pp. 5–10, 2008.

[33] K. O. Stanley, “Learning Concept Drift with a Committee of Decision Trees,” *Tech. report, UT-AI-TR-03-302, Dep. Comput. Sci. Univ. Texas Austin, USA*, pp. 1–17, 2003.

[34] G. Widmer and M. Kubat, "Learning Presence Concept Drift Hidden Context," *Mach. Learn.*, vol. 23, pp. 69–101, 1996.

[35] A. Narasimhamurthy and L. I. Kuncheva, "A Framework for Generating Data to Simulate Chanching," *Proc. 25th IASTED Int. Multi-Conference Artif. Appl. , February.*, pp. 384–389, 2007.

[36] L. L. Minku, S. Member, A. P. White, and X. Yao, "The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift," vol. 22, no. 5, pp. 730–742, 2010.

[37] R. Klinkenberg, "Learning drifting concepts : Example selection vs . example weighting," *Intell. Data Anal. IOS Press*, vol. 8, pp. 281–300, 2004.

[38] J. Z. Kolter and A. M. Maloof, "Dynamic Weighted Majority : An Ensemble Method for Drifting Concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, 2007.

[39] S. J. Delany, "A Comparison of Ensemble and Case-Base Maintenance Techniques for Handling Concept Drift in Spam Filtering," pp. 340–345.

[40] V. Ganti, J. Gehrke, and R. Ramakrishnan, "Mining data streams under block evolution," *ACM SIGKDD Explor. Newsl.*, vol. 3, no. 2, p. 1, Jan. 2002.

[41] M. Ebner, M. O'Neill, A. Ekárt, L. Vanneschi, and A. I. Esparcia-Alcázar, Eds., *Genetic Programming*, vol. 4445. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[42] M. Baena-garc and R. Gavald, "Early Drift Detection Method," *ECML PKDD 2006 Work. Knowl. Dis-covery from Data Streams, 2006.*, pp. 1–10, 2006.

[43] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*, 2003, p. 226.

[44] G. Forman, "Tackling concept drift by temporal inductive transfer," *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR '06*, pp. 252–259, 2006.

[45] K. H. Michael Harries, "Detecting Concept Drift in Financial Time Series Prediction using Symbolic Machine Learning."

[46] "Tracking Context Changes through Meta-Learning," *Mach. Learn.*, vol. 27, no. 3, pp. 259–286.

[47] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," *Proc. seventh ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '01*, pp. 97–106, 2001.

[48] Zhu Qiuming, "Pattern Classification in Dynamic Environments: Tagged Feature-Class Representation and the Classifiers," *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 19, no. 5, pp. 1203–1210, 1989.

[49] T. M. Mitchell and S. B. Thrun, "Explanation-Based Neural Network Learning for Robot Control," *Adv. Neural Inf. Process. Syst. 5*, pp. 287–294, 1993.

[50] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[51] J. Pan, “Features-Bases Transfer Learning with Real-World Applications,” no. September, 2010.

[52] A. Arnold, R. Nallapati, and W. W. Cohen, “A Comparative Study of Methods for Transductive Transfer Learning,” *Seventh IEEE Int. Conf. Data Min. Work. (ICDMW 2007)*, vol. 1, pp. 77–82, Oct. 2007.

[53] R. Raina and A. Y. Ng, “Self-taught Learning : Transfer Learning from Unlabeled Data,” pp. 759–766, 2000.

[54] S. Lee, D. Vickrey, K. Cs, and S. Edu, “Learning a Meta-Level Prior for Feature Relevance from Multiple Related Tasks,” pp. 489–496.

[55] C. J. Schlimmer and H. R. J. Granger, “Incremental Learning from Noisy Data,” *Mach. Learn.*, vol. 1, pp. 317–354, 1986.

[56] S. M. Lee, D. S. Kim, J. H. Kim, and J. S. Park, “Spam Detection Using Feature Selection and Parameters Optimization,” *2010 Int. Conf. Complex, Intell. Softw. Intensive Syst.*, no. i, pp. 883–888, Feb. 2010.

[57] B. Wenerstrom and C. Giraud-carrier, “Temporal Data Mining in Dynamic Feature Spaces,” *Procedings Sixth Int. Conf. Data Min. IEEE. Comput. Soc.*, pp. 0–4, 2006.

[58] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, “Dynamic integration of classifiers for handling concept drift,” *Inf. Fusion*, vol. 9, no. 1, pp. 56–68, Jan. 2008.

[59] G.-H. Lai, C.-M. Chen, C.-S. Laih, and T. Chen, “A collaborative anti-spam system,” *Expert Syst. Appl.*, vol. 36, no. 3, pp. 6645–6653, Apr. 2009.

[60] N. Pérez-Díaz, D. Ruano-Ordás, J. R. Méndez, J. F. Gálvez, and F. Fdez-Riverola, “Rough sets for spam filtering: Selecting appropriate decision rules for boundary e-mail classification,” *Appl. Soft Comput.*, vol. 12, no. 11, pp. 3671–3682, Nov. 2012.

[61] “Combining Time and Space Similarity for Small Size Learning under Concept Drift.”

[62] M. Angel, “Recurring Concept Detection for Spam Filtering.”

[63] N. Lu, G. Zhang, and J. Lu, “Concept drift detection via competence models,” *Artif. Intell.*, vol. 209, pp. 11–28, 2014.

[64] K. Nishida and K. Yamauchi, “Adaptive Classifiers-Ensemble System for Tracking Concept Drift,” *Proc. Sixth International Conf. Mach. Learn. Cyberbetics, Hong Kong, August 2007.*, no. August, pp. 19–22, 2007.

[65] M. G. Kelly, D. J. Hand, and N. M. Adams, “The Impact of Changing Populations on Classifier Performance,” vol. 32, no. 2, pp. 367–371, 1998.

[66] J. R. Méndez, D. Glez-Peña, F. Fdez-Riverola, F. Díaz, and J. M. Corchado, “Managing irrelevant knowledge in CBR models for unsolicited e-mail classification,” *Expert Syst. Appl.*, vol. 36, no. 2, pp. 1601–1614, Mar. 2009.

[67] D. W. Aha, D. Kibler, and M. K. Albert, “Instance-based learning algorithms,” *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, Jan. 1991.

[68] X. Ling, W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Spectral domain-transfer learning," *Proceeding 14th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD 08*, pp. 488–496, 2008.

[69] J. Meng, H. Lin, and Y. Yu, "Transfer Learning Based on SVD for Spam Filtering," *2010 Int. Conf. Intell. Comput. Cogn. Informatics*, pp. 491–494, Jun. 2010.

[70] L. Duan, I. W. Tsang, and D. Xu, "Domain Transfer Multiple Kernel Learning," vol. 34, no. 3, pp. 465–479, 2012.

[71] J. Gao, W. Fan, J. Jiang, and J. Han, "Knowledge Transfer via Multiple Model Local Structure Mapping Categories and Subject Descriptors," pp. 283–291.

[72] X. H. Pham, N.-H. Lee, J. J. Jung, and A. Sadeghi-Niaraki, "Collaborative spam filtering based on incremental ontology learning," *Telecommun. Syst.*, pp. 693–700, 2011.

[73] X. Shi, Q. Liu, W. Fan, and P. S. Yu, "Transfer across completely different feature spaces via spectral embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 906–918, 2013.

[74] M. T. Bahadori, Y. Liu, and D. Zhang, "A general framework for scalable transductive transfer learning," *Knowl. Inf. Syst.*, vol. 38, pp. 61–83, 2014.

[75] L. I. Kuncheva, "Classifier Ensembles for Changing Environments," pp. 1–15, 2004.

[76] G. V. Cormack, "Email Spam Filtering: A Systematic Review," *Found. Trends® Inf. Retr.*, vol. 1, no. 4, pp. 335–455, 2008.

[77] C. D. Manning, "Foundations of Statistical Natural Language Processing."

[78] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," vol. 60, no. 5, 2004.

[79] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, Dec. 2009.

[80] G. Salton, A. Wong, and C. S. Yang, "AVector Space Model for Automatic Indexing," vol. 18, no. 11, 1975.

[81] L. Hamers, Y. Hemeryck, G. Herweyers, M. Janssen, H. Keters, R. Rousseau, and A. Vanhoutte, "SIMILARITY MEASURES IN SCIENTOMETRIC SALTON ' S COSINE FORMULA," vol. 25, no. 3, pp. 315–318, 1989.

[82] J. R. Quinlan, *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann Publishers In, 1992, p. 316.

[83] A. Bifet and R. Gavald, "Learning from Time-Changing Data with Adaptive Windowing a," 2006.

[84] K. Test, "12-5 Kruskal-Wallis Test," *Statistics (Ber).*, vol. 23, no. 1, pp. 57–62, 1986.

[85] C. Caragea and A. Silvescu, "Combining Hashing and Abstraction in Sparse High Dimensional Feature Spaces," pp. 3–9, 2009.

[86] C. Laorden, I. Santos, B. Sanz, G. Alvarez, and P. G. Bringas, "Word sense disambiguation for spam filtering," *Electron. Commer. Res. Appl.*, vol. 11, no. 3, pp. 290–298, May 2012.

[87] R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse, "WEKA Manual for Version 3-7-6," 2012.

[88] Y. Zhu and Y. Tan, "A local-concentration-based feature extraction approach for spam filtering," … *Forensics Secur. IEEE Trans.*, vol. 6, no. 2, pp. 486–497, 2011.

[89] B. Yu and Z. Xu, "A comparative study for content-based dynamic spam classification using four machine learning algorithms," *Knowledge-Based Syst.*, vol. 21, no. 4, pp. 355–362, May 2008.

[90] B. Ribeiro and C. Silva, "RVM Ensemble for Text Classification," *Int. J. Comput. Intell. Res.*, vol. 3, no. 1, pp. 31–35, 2007.

[91] V. Metsis, "Spam Filtering with Naive Bayes – Which Naive Bayes," 2006.

[92] S. J. Delany, "A Comparison of Ensemble and Case-Base Maintenance Techniques for Handling Concept Drift in Spam Filtering."

[93] P. Felber, M. Rajman, and E. Rivi, "S PADS : Publisher Anonymization for DHT Storage," 2010.

[94] E. Minkov, R. C. Wang, and W. W. Cohen, "Extracting Personal Names from Email : Applying Named Entity Recognition to Informal Text," no. October, pp. 443–450, 2005.

[95] W. W. Cohen, "Learning to Extract Signature and Reply Lines from Email," 2004.

[96] D. Harman, "OVERVIEW OF THE SECOND TEXT RETRIEVAL," vol. 31, no. 3, pp. 271–289, 1995.

[97] C. A. Martins, M. C. Monard, and E. T. Matsubara, "PreTexT : uma ferramenta para pré-processamento de textos utilizando a abordagem bag-of-words," São Paulo - SP, 2003.

[98] G. M. Weiss and F. Provost, "Learning When Training Data are Costly : The Effect of Class Distribution on Tree Induction," vol. 19, pp. 315–354, 2003.

Apêndice A

Construção Base de dados

Este apêndice descreve alguns trabalhos que utilizam bases de dados *spam* disponíveis publicamente. Além disso, são apresentados os detalhes da construção de uma nova base de dados, ETSS (*Emerging Technologies and System Security*), coletada no período de junho de 2012 a outubro de 2013. Detalhes como método de extração de características, estratégia de redução do espação de dimensionalidade de características, definição da quantidade de características que melhor representam a descrição do problema e representatividade das características que compõe a base de dados, são exibidos.

A.1. Aplicações empregando bases de dados públicas para deteção de *Spam*

As bases de dados utilizadas em trabalhos que investigam técnicas de deteção de *spam* normalmente são coleções de documentos. Para que as amostras sejam usadas na construção de modelos de classificação com aprendizagem de máquina, normalmente são extraídas características dos documentos na forma de "*bag of words"*. Essa técnica constrói um vocabulário de tamanho *d*, que contém todas as palavras existentes na coleção de documentos. Em seguida, cada documento passa a ser representado como um vetor de características, onde cada entrada *k* do vetor de características recebe a quantidade de ocorrências no documento da *k*-ézima palavra do dicionário.

Essa estratégia pode gerar pelo menos dois grandes problemas. Primeiramente, "*bag of words"* pode produzir espaços de características com dimensão proibitivamente elevada. Além disso, como apenas uma pequena quantidade de palavras ocorre em cada documento, quando comparada ao tamanho do dicionário, a representação do espaço de características de cada amostra é muito esparsa, pois, poucos valores de características serão diferentes de zero [85]. Portanto, estratégias de seleção de características são comumente empregadas na literatura para reduzir o espaço de características, conforme pode ser observado nas bases de dados descritas nesta seção.

Laorden et al. [86] trabalham com duas bases públicas: *Ling Spam*[4] e *TREC*[5]. Os autores exploram o uso de semântica para filtros de *spam,* introduzindo um pré-processamento na busca de palavras de sentido ambíguo (do inglês *Word Sense Disambigation – WSD*). O propósito na aplicação de WSD é recuperar a capacidade de filtragem de métodos baseados em conteúdo. A extração de dados é feita com base em termos de frequência (do inglês *Frequency Term – Inverse Document Frequency – TF-IDF*). Uma etapa de pré-processamento é aplicada antes da construção do vetor de características. Em seguida, a estratégia de seleção de características Information Gain (IG) é usada para reduzir a dimensionalidade do espaço de características. Para a tarefa de classificação, foram usados quatro diferentes algoritmos de aprendizagem de máquina baseados na precisão de classificação de cada algoritmo: SVM (*Support Vector Machine*) [Drucker et al., 1999], *Bayesian Networks* [87], *NB (Naive Bayes), Decision Trees* [82].

Zhu e Tan [88]) aplicam uma abordagem a partir de um sistema de imunidade biológica - BIS *Biological Immune System* - sobre 05 bases de dados: PU1, PU2, PU3, PUA e Enron-Spam[6]. A extração de características é baseada na concentração local das características inspiradas a partir da *Biological Immune System – BIS.* Essa abordagem utiliza *tokenization* com base em espaços em branco e delimitadores. Para seleção características são usados IG, TFV (*Term Frequency Variance*) e DF (*Document Frequency).* Para a tarefa de classificação, o algoritmo de aprendizagem de máquina adotado é SVM e a métrica utilizada é a precisão do classificador. Os autores investigam o desempenho do classificador sobre as três estratégias de seleção de características apresentadas.

Yu e Xu [89] apresentam uma avaliação empírica sobre duas bases de dados, *SpamAssassin* e *Babletext*[7]. A extração de características é suportada pela representação da abordagem *Bag-of-words*. Os autores não definem a estratégia usada para seleção de características, apenas se restringem a dizer que é usada uma seleção de características para escolher as palavras mais importantes e reduzir a dimensionalidade do espaço de características. Quatro métodos são avaliados para a tarefa de classificação por meio da taxa de precisão do classificador: NB [79], Neural Network [14], SVM e RVM (*Relevance*

[4] http://nlp.cs.aueb.gr/software_and_datasets/lingspam_public.tar.gz.
[5] http://plg.uwaterloo.ca/?gvcormac/*spam*.
[6] Todas as cinco bases são encontradas em: http://www.aueb.gr/users/ion/publications.html
[7] http://www.babeltext.com/*spam*/

Vector Machine) [90].

Metsis et al. [91] investigam uma base de dados construída a partir da caixa de e-mail de seis grupos de empregados da empresa ENRON, e mensagens *spam* obtidas de diferentes fontes como: *SpamAssassin*[8], projeto *Honeypot*[9] *e a* coleção de *Bruce Guenter*[10]. A nova base de dados construída a partir da composição apresentada foi baseada somente no assunto e no corpo dos e-mails para não comprometer os proprietários dos e-mails. A extração de características foi fundamentada na frequência de palavras e originou seis novas bases Enron[11] denominadas: Enron1, Enron2, Enron2, Enron4, Enron5 e Enron6. A estratégia para reduzir a dimensionalidade do espaço de características foi novamente IG. Para tarefa de classificação foram usadas diferentes versões de *NB* observando a taxa de precisão do classificador.

Por fim, Delany et al. [92] apresentam a base de dados ECUE (*Email Classification Using Examples – ECUE*), cujas amostras foram coletadas entre novembro de 2002 e janeiro de 2004. Os autores empregam CBR (*Case-Based Reasoning*), que é uma técnica de aprendizado de máquina lenta (do inglês *lazy learning*), para o problema de filtro de *spam*. O conjunto de dados foi construído com base na estrutura, nas palavras e nos caracteres dos e-mails. Os autores também utilizaram a estratégia de seleção de características IG para reduzir a dimensionalidade do espaço de características. O método de aprendizagem de máquina *k-nearest neighbour* foi usado na tarefa de classificação. Os autores também utilizaram a estratégia de seleção de instância para deteção de mudança nos dados. Nesse caso, os e-mails classificados incorretamente como *spam* ou legitimo são inseridos no caso base como forma de atualizar a base de conhecimento. É importante destacar que nessa base de dados, as amostras estão organizadas por data de aquisição.

Portanto, conforme mencionado anteriormente, os trabalhos que investigam técnicas de deteção de *spam* normalmente utilizam bases de dados que são coleções de documentos com espaço de características com dimensionalidade elevada. Pode-se observar também que todos os trabalhos descritos neste apêndice aplicam IG como seleção de características. Entretanto, apenas a base de dados ECUE, utilizada em Delany et al. [92], possui dados coletados por um período relativamente longo de tempo, quinze meses, e também apresenta os dados organizados por data de aquisição. Logo, a base

[8] http://www.spamassassin.org/publiccorpus/
[9] http://www.projecthoneypot.org/
[10] http://untroubled.org/*spam*
[11] http://www.iit.demokritos.gr/skel/i-config/

ECUE é a única base listada nesta seção que pode ser utilizada em nossos experimentos.

Diante desse contexto, optou-se pela criação de uma nova base dados devido aos seguintes fatores: (1) os resultados dos experimentos poderiam não ser muito gerais se obtidos em apenas uma base de dados; ECUE pode ser considerada uma base de dados antiga, logo, a criação de uma base de dados mais recente tornou-se mandatória. A base de dados criada nesta tese está descrita na próxima seção.

A.2. Construção da base de dados ETSS

Para a construção da base de dados ETSS, as características foram extraídas de mensagens *spam* e legítimas (*nonspam*). A coleta das mensagens conservou a periodicidade da chegada de cada amostra e as concentrou em janelas de tempo mensais, correspondendo ao período do ano em que foram recebidas pelos destinatários. É importante destacar que esta base de dados foi organizada a fim de permitir a investigação da evolução das características no contexto de *concept drift*, de acordo com o objetivo desta tese. O conjunto de e-mails *nonspam* e *spam* foi coletado no período que compreende os meses de Junho de 2012 a Outubro de 2013. A partir das seguintes fontes:

1. E-mails *spam* - coletados a partir da classificação de *N* usuários para as mensagens consideradas *spam* de um repositório acadêmico.
2. E-mail *nonspam* – coletados a partir de 07 usuários distintos de um ambiente acadêmico.

A Tabela A.1 apresenta a distribuição da base de dados exibindo o percentual de e-mails *nonspam,* em relação aos e-mails *spam* na sua forma bruta, ou seja, apenas a quantidade de amostras na sua íntegra.

Tabela A.1. Distribuição Base de Dados ETSS Bruta

JANELAS DE TEMPO																		
Mensagem	**2012**							**2013**										**Total**
	Jun	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	
Spam	2521	8714	5718	4884	5609	5061	5020	10830	5069	6829	5807	8345	4027	11584	9363	7789	5483	112653
Nonspam	1123	1020	1204	1432	1761	1501	1299	1333	1530	1688	1979	2064	1726	2005	1772	1455	1475	26447
Total	**3644**	**9734**	**6922**	**6316**	**7370**	**6562**	**6319**	**121163**	**6599**	**8517**	**7786**	**10409**	**5753**	**13589**	**11135**	**9244**	**6958**	**139100**

Para que a base fosse totalmente composta por e-mails em português, foi feita uma busca por e-mails que não estavam em português e/ou que estavam fora do período

desejado, para removê-los da coleção de dados. A Tabela A.2 apresenta a nova composição da base de dados usada na etapa de extração de características.

Tabela A.2. Distribuição da Base de Dados ETSS

JANELAS DE TEMPO																		
Mensagem	**2012**							**2013**										**Total**
	Jun	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	
Spam	2417	8433	5367	4669	5427	4924	4229	10281	4737	6511	4039	7954	3741	11016	8700	7476	5293	**105.214**
Nonspam	1084	960	1157	1350	1715	1476	1258	1282	1497	1608	1915	2007	1676	1910	1712	1423	1454	**25.484**
Total	**3.501**	**939**	**6.524**	**6.019**	**7.142**	**6.400**	**5.487**	**11.563**	**6.234**	**8.119**	**5.954**	**9.961**	**5.417**	**12.926**	**10.412**	**8.899**	**6.747**	**130.698**

Seguindo o processo de construção da base de dados, passou-se à etapa de anonimização das mensagens considerando o corpo e o assunto dos e-mails. Os seguintes pontos para anonimização foram analisados, de acordo com a literatura [93] [94] [95]:

1. Lista de nomes mais comuns incluído alguns sobrenomes;
2. Numerais;
3. Endereços;
4. Símbolos (/, \, |,),(,*,&, $, @, !, ?, etc);

A etapa de anonimização, além de garantir a privacidade dos usuários que disponibilizaram suas mensagens para esta pesquisa, também, diminui a geração de características que não serão relevantes para o aprendizado de um algoritmo em aprendizagem de máquina.

Após a etapa de anonimização, foi aplicada a estratégia *Bag-of-Word* (BoW) considerando remoção de *stopwords* e *stemming* (redução das palavras a radicais). Considerar a remoção de *stopwords* em BoW implica em desconsiderar artigos, pronomes, numerais ordinais, entre outros. O uso da técnica *stemming*, por sua vez, implica em considerar somente os radicais das palavras e desconsiderar prefixos e sufixos [96][97]. Por fim, a base de dados final foi composta por 105.480 características.

Como o espaço de características obtido ainda foi muito elevado, tornou-se necessário reduzi-lo. Contudo, antes de reduzir a dimensionalidade do espaço de características, é imprescindível validar a representatividade das características extraídas para o problema. Com o objetivo de avaliar a representatividade dos dados e das características, cada janela de tempo foi utilizada em um processo de validação cruzada, sendo que todas as 105.480 características foram incialmente empregadas. Portanto, cada

janela de tempo componente da base de dados ETSS, foi utilizada separadamente em um processo de treinamento/teste via validação cruzada. Os resultados obtidos são exibidos na próxima seção.

A.3. Avaliação da Representatividade das Características Extraídas

Os experimentos foram realizados com auxílio da ferramenta WEKA, sendo que foi utilizada a configuração padrão do algoritmo de aprendizagem SVM, isto é, fator de penalização *C*=1.0, *kernel* polinomial de grau 1. A Figural A.1 apresenta as taxas de erro obtidas por SVM em cada janela de tempo via validação cruzada com 10 partições. Pode-se observar que as características que representam as amostras descrevem bem o problema, pois o classificador alcançou taxas de erro menores se comparada com a Figura A.2 que apresenta a taxa de erro do modelo de classificação sendo avaliado com a série 1. A Tabela A.3 apresenta a distribuição da quantidade de características que compõem cada classe.

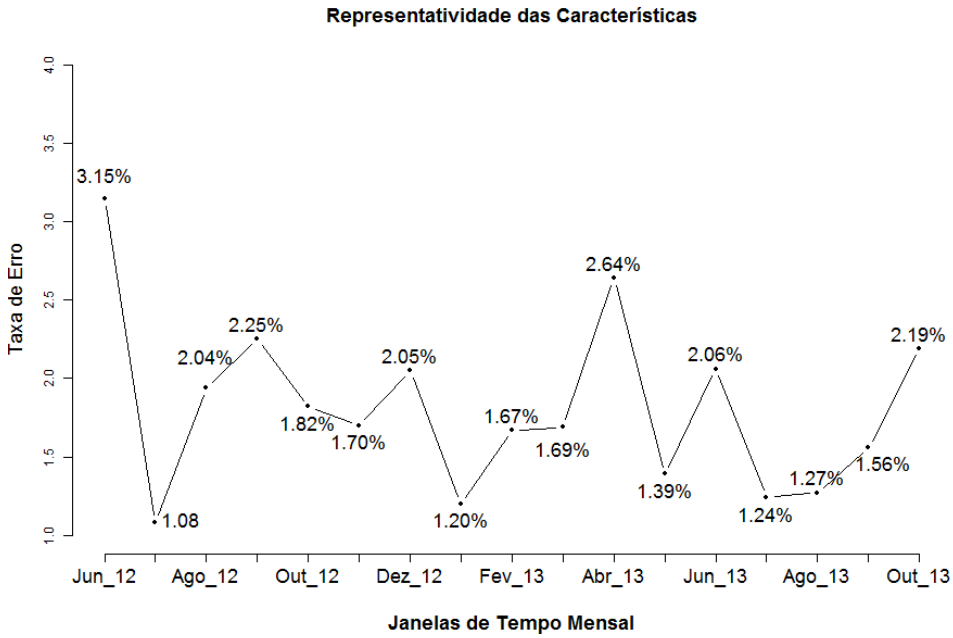

Figura A.1. Desempenho do classificador com todas as características extraídas via BoW. Classificador SVM treinado e testado com estratégia de validação cruzada usando 10 partições.

A distribuição apresentada pela Tabela A.3 auxilia no entendimento do percentual da taxa de erro apresentada na Figura A.1. Observando a primeira janela de tempo, Junho de 2012, que apresenta um percentual de características da classe *spam* presente no vetor de características que descrevem a classe *nonspam* demonstra a dificuldade no

aprendizado do modelo de classificação. Esse comportamento se estende as demais janelas de tempo apresentando uma variabilidade no percentual de características *spam* que se encontram presentes no vetor de características *nonspam*. Por exemplo, a janela de tempo de Janeiro de 2013, apresenta um percentual de erro de 1. 20%, Figura A.1, e um percentual de características *spam* presente no vetor de características nonspam de 27%. Demonstrando que quanto menor esse percentual de características comuns entre as duas classes, melhora o aprendizado do classificador.

Tabela A.3. Representação das classes de Spam e Nonspam por meio da quantidade de Características presente em cada classe e quantidade de características Spam encontradas no vetor de Características Nonspam

Mensagens	**Representatividade das Características por classe**																
	2012							**2013**									
	Jun	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**
Spam	3.280	4.007	5.427	6.032	6.199	4.135	3.611	8.040	4.303	5.270	4.514	6.181	4.329	6.873	5.849	6.064	4.944
Nonspam	11.368	11.946	12.764	12.841	12.877	12.947	10.453	11.446	13.455	13.737	15.801	14.398	735	13.552	13.470	12.703	10.937
Características *spam* encontradas no vetor de nonspam	1.390	1.560	2.117	2.266	1.993	1.658	1.447	2.209	1.570	2.021	1.697	2.153	372	2.396	2.119	2.087	1.832
Percentual	42%	39%	39%	37%	32%	40%	40%	27%	37%	38%	37%	34%	8.5%	34%	36%	34%	37%

A.4. Domínio Fonte e Alvo

Confirmada a boa representatividade das características que compõem a base de dados, o próximo passo envolveu a divisão da base de dados original em base de treino e base teste, respectivamente, domínio fonte e domínio alvo.

A.4.1. Domínio Fonte

O domínio fonte foi composto por 1000 amostras selecionadas aleatoriamente a partir das três janelas de tempo iniciais – junho, julho e agosto de 2012. Para seguir orientações da literatura [98], a base de treinamento foi balanceada, isto é, 50% de amostras *spam* e 50% de amostras *nonspam*. Tabela A.4 apresenta a distribuição de treino domínio fonte. É importante salientar que está composição originou um espaço de características cuja dimensionalidade é 9.355 características.

Tabela A.4. Distribuição das amostras de treino/domínico fonte – ETSS

Mensagem	Domínio Fonte			Total
	2012			
	Jun	Jul	Ago	
Spam	167	167	166	**500**
Nonspam	167	167	166	**500**
Total	**334**	**334**	**332**	**1.000**

Uma vez definida a base de treinamento, a próxima etapa envolveu a redução de dimensionalidade, também de acordo com a literatura. A estratégia IG foi aplicada para reduzir o espaço de características e identificar as características mais relevantes para o problema. Além disso, também foi necessário definir os melhores parâmetros do classificador.

1) *Busca de menor quantidade de características para representar o problema*

Nesta etapa, a ferramenta WEKA foi utilizada com o algoritmo de classificação SVM em sua configuração de parâmetros padrão (fator de penalização *C=1.0, kernel polinomial de grau 1*) fixa. Apenas a quantidade de características foi alterada. O objetivo desta fase é buscar a menor quantidade de características relevantes que mantenha a taxa de erro baixa. A Tabela A.5 apresenta o desempenho do classificador iniciando com todas as características, 9.355, e chegando ao mínimo de 300 características como a melhor representação.

Tabela A.5. Redução da Dimensionalidade do Espaço de Características

Quantidade Características	Parâmetros	Métricas			
		Precisão	Erro	FP	FN
9.355	*C=1, Kernerl Poly, grau 1*	95.90%	4.10%	7.80%	0.40%
4.675	*C=1, Kernerl Poly, grau 1*	96.20%	3.80%	7.20%	0.40%
2.000	*C=1, Kernerl Poly, grau 1*	96.20%	3.80%	7.20%	0.40%
1.000	*C=1, Kernerl Poly, grau 1*	96.20%	3.80%	7.20%	0.40%
800	*C=1, Kernerl Poly, grau 1*	95.90%	4.10%	7.40%	0.40%
500	*C=1, Kernerl Poly, grau 1*	96.10%	3.90%	7.00%	0.80%
300	***C=1, Kernerl Poly, grau 1***	**96.40%**	**3.60%**	**6.40%**	**0.80%**

2) Busca dos melhores parâmetros para o classificador

Nesta etapa, nosso objetivo é definir os melhores parâmetros de SVM. Para tanto, se manteve a quantidade de características fixa, 300 (trezentas), enquanto as configurações dos parâmetros do classificador SVM foram alteradas, com o objetivo de alcançar o melhor desempenho do classificador a partir do ajuste dos parâmetros. A Tabela A.6 mostra como os valores dos parâmetros foram variados, assim como a melhor configuração de parâmetros (*c=100, kernel RBF com gama de 0.1*) para as 300 características selecionadas na fase anterior.

Tabela A.6. Ajuste Parâmetros para Domínio Fonte

Quantidade	Parâmetro				Métricas			
Características	***C***	**Kernel**	**Grau**	**Gama**	**Precisão**	**Erro**	**FP**	**FN**
300	*3*	Poly	1	-	95.90%	4.10%	7.00%	1.20%
300	10	Poly	1	-	94.70%	5.30%	8.40%	2.20%
300	100	Poly	1	-	94.60%	5.40%	8.20%	2.60%
300	1	RBF	-	0.01	93.80%	6.20%	12.40%	0.00%
300	*5*	RBF	-	0.01	96.30%	3.70%	7.40%	0.00%
300	10	RBF	-	0.01	96.50%	3.50%	7.00%	0.00%
300	100	RBF	-	0.01	96.80	3.20%	5.40%	1.00%
300	200	RBF	-	0.01	95.80%	4.20%	6.60%	1.80%
300	100	RBF	-	1.0	94.00%	6.00%	2.20%	9.80%
300	**100**	**RBF**	**-**	**0.1**	**3.20%**	**96.80%**	**4.60%**	**1.80%**

Essas duas fases de ajuste do número de características e dos parâmetros do classificador indicam as seguintes configurações: base de dados com 300 características e 1000 amostras com classificador SVM de parâmetros conforme destacado na Tabela A.6 (*c=100, kernel RBF com gama de 0.1*).

A.4.2. Domínio Alvo

O domínio alvo é representado pelas 14 janelas de tempo seguintes ao domínio de treino. A Tabela A.7 apresenta a distribuição do domínio alvo.

Tabela A.7 Distribuição das amostras de teste - ETSS

Domínio Alvo															
Mensagem	**2012**				**2013**										**Total**
	Set	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	
Spam	4.669	5.427	4.878	4.229	10.281	4.737	6.511	4.039	7.954	3.741	11.016	8.700	7.476	5.293	**88.997**
Nonspam	1.350	1.715	1.474	1.258	1.282	1.497	1.608	1.915	2.007	1.676	1.910	1.712	1.423	1.454	**22.283**
Total	**6.019**	**7.142**	**6.352**	**5.487**	**11.563**	**6.234**	**8.119**	**5.954**	**9.961**	**5.417**	**12.926**	**10.412**	**8.899**	**6.747**	**111.280**

A.5. Avaliação do Comportamento Estático e da Evolução das Características

A metodologia empregada nestes experimentos é a mesma apresentada na seção 4.5, Série 1 e Série 2.

As Tabelas A.4 e A.7 apresentam a distribuição da base de dados ETSS. No entanto a mesma estratégia de redução de espaço de características que foi aplicada na ECUE foi, também, aplicada na base de dados ETSS. Para redução da base de dados ETSS, foram escolhidas as 300 características mais relevantes. Conforme Tabela A.**5**. essa é a quantidade de característica que melhor representa o problema de deteção de *spam* para o conjunto de dados ETSS.

A.5.1. Analise dos Resultados

A análise dos resultados inicia pela avaliação da eficiência do modelo de classificação durante a fase de teste, Tabela A.7 Distribuição das amostras de teste - ETSS. A apresenta o desempenho do classificador considerando a taxa de erro de classificação obtida por meio da equação 4.1. Como pode ser observado o classificador apresenta um comportamento esperado, ou seja, um aumento relativamente gradativo e ascendente da taxa de erro durante os meses analisados. Contudo, algumas observações requerem uma análise mais cuidadosa. Por exemplo, porque o mês de setembro, mês seguinte ao espaço de treino apresenta, da mesma forma, uma taxa de erro alta de 3.88% e no mês de novembro reduz para 2.04%? É visível na Figura 5.2 a oscilação na taxa de erro da base de dados ECUE chegando a uma taxa de 22.03% em janeiro de 2004. No entanto a base de dados ETSS, Figura A.2, mantem uma média de 2.50% na taxa de erro contra 7.90% da base de dados ECUE. Esse comportamento leva ao mesmo questionamento na seção 4.5, se as amostras que compõem o espaço de treino utilizado para gerar o modelo de classificação não têm características representativas para o problema?

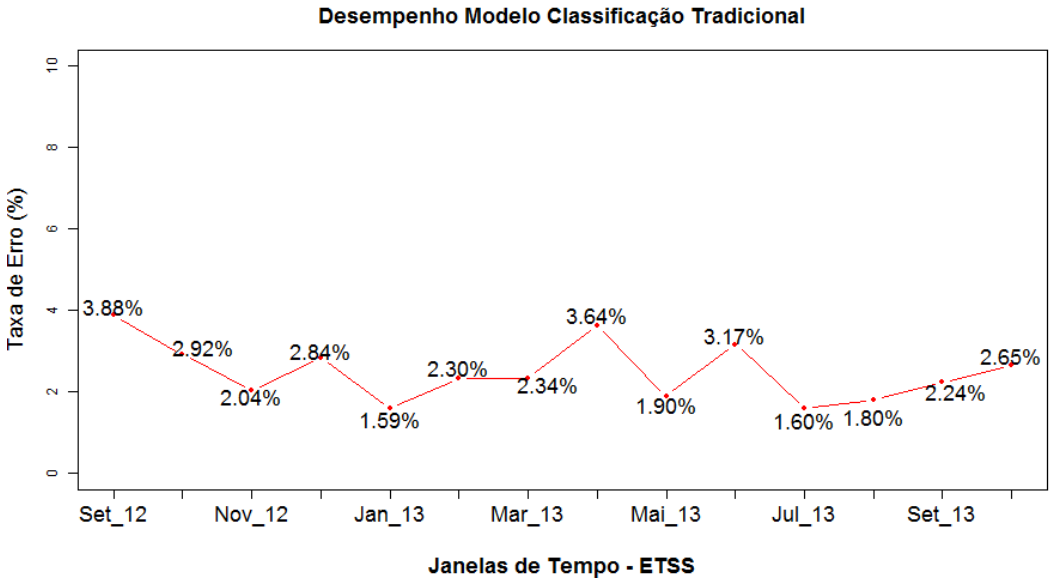

Figura A.2. Taxa de erro obtida usando um modelo de classificação estático sobre a base de dados ETSS - Série 1

No entanto a Série 2 é aplicada, também, sobre a base de dados ETSS. Onde o modelo de classificação é construído e testado mensalmente durante o período de teste. É importante destacar que a seleção de características também é realizada mensalmente nestes experimentos.

A Figura A.3 exibe o modelo de classificação construído a partir de características selecionadas durantes três meses de treino (experimentos denominados de Série 1) e uma comparação das taxas de erro obtidas por modelos de classificação mensais (Série 2).

Comparando as taxas de erros de ambas as séries, é possível observar que a construção de modelo de classificação a partir da seleção de características mensais é sempre mais eficiente que o modelo "estático" usado na primeira série de experimentos. Essa observação se aplica tanto para base de dados ECUE como para a base de dados ETSS, observando-se respectivamente na Figura 5.2 e Figura A.3.

Ainda observando a Figura A.3, enquanto na série 1 se obtém uma taxa média de erro de 2.49%, a série 2 apresenta 1.40%. Além demonstrar a representatividade das características para o problema de maneira satisfatória, também, é possível acompanhar a dinamicidade das características demonstrada com a evolução abrupta no mês seguinte ao período de treino.

A base de dados ETSS apresenta em torno de 130.000 amostras com aproximadamente 95.000 características, Tabela A.2, gerando um espaço de

dimensionalidade de características menor que a base de dados ECUE, porém com uma representatividade de amostras superior, garantido uma maior estabilidade na taxa de erro do modelo de classificação conforme Figura A.3.

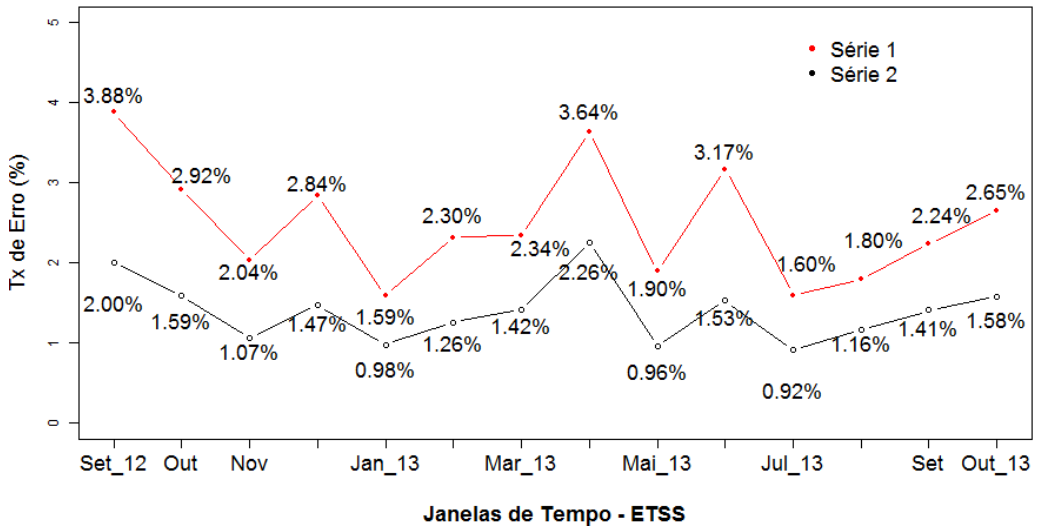

Figura A.3. Taxas de erros obtidas usando um modelo de classificação estático (Série 1) e um modelo atualizado mensalmente (Série 2) sobre base de dados ETSS.

Assim como foi efetuado sobre a base de dados ECUE e a partir das observações realizadas nesses dois experimentos iniciais, se repete os dois questionamentos: existe uma relação direta entre evolução das características e taxa de erro do classificador? Poderia se obter, a partir dessa relação, um limiar para determinar até quando um modelo de classificação continua efetivo? Para investigar esta relação, uma análise é feita sobre o vetor de característica que representa o domínio de treino, gerado via *IG*, e os vetores de características que representam cada domínio de teste, também gerados via *IG*. Esta investigação tem como objetivo levantar o percentual de características ausentes no domínio de teste em relação ao domínio de treino.

A Figura A.4 apresenta o percentual de características que estão ausentes no domínio de treino, considerando janelas de tempo mensais, em relação ao domínio de teste. Os resultados mostram que logo no primeiro mês de teste (setembro de 2012), o percentual de características ausentes é de 19,00% e uma taxa de erro de 3.88%, Figura A.3. Essa evolução das características teve um reflexo direto na taxa de erro do classificador. Contudo se observamos a janela de tempo de outubro de 2012 teve uma evolução de 33% e uma taxa de erro de 2.92% Figura A.3, esta relação fica dúbia.

Percebe-se que quanto mais amostras apresenta a base de dados, mas estável esta base se mostra para gerar um modelo de classificação. Quando se tem uma base com menos amostras as características que representam um e-mail *spam* ou *nonspam*, torna-se limitado. No entanto quando a quantidade de amostras procura representar o mundo real (80 % amostras *spam* e 20% amostras *nonspam*) em uma proporção maior destas amostras de e-mails *spam* e *nonspam,* as características que representam as amostras se apresentam mais estáveis, ou seja, a evolução de características do domínio fonte para o domínio alvo apresenta uma evolução média de aproximadamente 34%, conforme Figura A.4. Enquanto que com menos amostras as características que representando os e-mails *spam* e *nonspam,* a taxa de evolução destas características chega a atingir aproximadamente 67%, Figura A.4. Por outro lado, enquanto se consegue reduzir a evolução de características entre o domínio fonte e domínio alvo, aumenta a taxa de erro do modelo de classificação devido ao aumento na dimensionalidade no espaço de características. Como encontrar um equilíbrio entre evolução de características e taxa de erro para gerar modelos de classificação com confiabilidade maior ao longo do tempo para problemas dinâmicos?

Logo, encontrar a validade de um modelo de classificação baseado na evolução das características requer uma análise mais detalhada. Apesar de valores percentuais de presença ou ausência de características servirem como bons indicadores para atualizar o modelo de classificação, outros fatores devem ser levados em consideração como a relevância e a frequência das novas características.

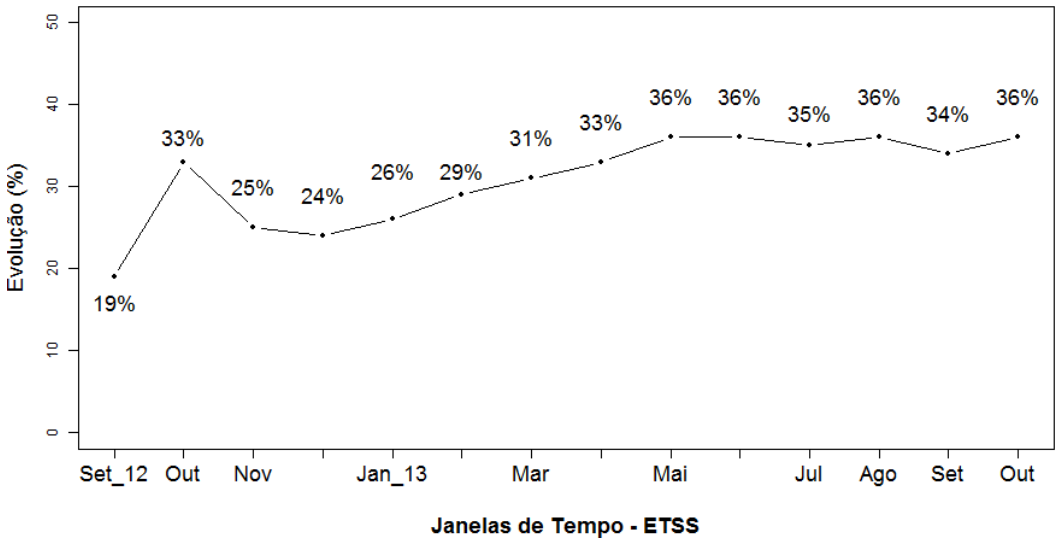

Figura A.4. Percentual de características ausentes comparando os domínios fonte e alvo para a base de dados ETSS.

A.5.2. Discussão

A reaplicação das Séries 1 e 2 sobre a base de dados ETSS reafirmou as considerações da seção 4.5. Os resultados apresentados demonstraram que a relação da evolução de características de ataques e taxa de erro do classificador são importantes para identificar a efetividade de um modelo de classificação, porém, essa relação não é suficiente para determinar um limiar de mudança.

Salientando, ainda, que a partir dos experimentos sobre duas bases de dados de ataques do tipo *spam* podem gerar ou requisitar limiares diferentes. Então conclui-se ainda que determinar um limiar para inferir o momento de atualizar o modelo de classificação é muito especifico e de difícil inferência, pois é dependente da distribuição dos dados que descrevem o problema.

A.6. Dados interessantes sobre base de dados ETSS

Além das informações mencionadas anteriormente, é importante é destacar que na base ETSS, o tamanho do vetor de características que representam as mensagens de *spam* e de *nonspam* é bem diferente. Por exemplo, mensagens *spam* necessitam de 32.311 características para serem representadas, enquanto, mensagens *nonspam* apresentam

número de características três vezes maior, ou seja, 75.169 características. Das 32.311 características detectadas em *spam*, 38% estão presentes no vetor de características *nonspam*, ou seja, 12.284 características são comuns entre *spam* e *nonspam*.

Apêndice B
Similaridade empregando Distância Euclidiana

Este apêndice descreve uma estratégia de análise de similaridade entre o domínio fonte e o domínio alvo em problemas de deteção de *spam*. A estratégia investigada neste apêndice é baseada no uso da técnica de aprendizagem não supervisionada k-means, conforme descrito nas próximas seções.

A partir da investigação da relação entre taxa de erro de classificação e evolução de características discutida na seção 4.2.1 do Capítulo 4, é possível observar uma relação existente, porém, não determinante a ponto de possibilitar a definição de um limiar de evolução de características capaz de indicar deteção de mudança. No intuito de determinar uma medida mais precisa de deteção de mudança, k-means foi utilizado com o objetivo de medir quão distantes as amostras de teste encontram-se em relação às amostras de treino.

Essa estratégia baseia-se na seguinte hipótese: com o passar do tempo, as novas amostras devem se distanciar dos grupos de dados que representam as classes *spam* e *nonspam*, pois, dado que esses grupos são definidos durante o treinamento, a classe *spam* deve se aproximar da classe *nonspam* devido às técnicas e aos artifícios aplicados pelos *spammers*.

B.1. Metodologia

No intuito de investigar a hipótese acima mencionada, o algoritmo de agrupamento K-*means* foi aplicado nos experimentos descritos neste apêndice. Foram utilizados dados obtidos a partir da aplicação de IG e de Frequência para reduzir a dimensionalidade do espaço de características. Apenas a base de dados ETSS foi utilizada, assim como também, apenas janelas de tempo de setembro a dezembro de 2012 foram usadas nos experimentos. Portanto, dois grupos de experimentos foram executados.

a) Empregando estratégia de seleção de características IG:

Primeiramente, a estratégia IG é usada para que sejam selecionadas as características mais relevantes para representar os dados. Em seguida, são calculados os centroides das duas

classes que representam o problema, isto é, *spam* e *nonspam,* a partir dos dados de treinamento. O centroide é o ponto médio entre os membros de um mesmo grupo. Por fim, é calculada a distância euclidiana entre os centroides das duas classes e as amostras que compõem o domínio alvo.

b) Empregando estratégia de seleção de características Frequência:

O mesmo procedimento realizado com IG é empregado aqui. Entretanto, a estratégia de seleção de características aplicada é baseada na frequência, ou seja, as *n* características mais frequentes nos dados de treinamento são usadas para representar tais dados. Os demais passos são idênticos aos acima descritos.

B.2. Análise dos resultados

A Figura B.1 ilustra a distância das amostras de teste em relação aos centroides *nonspam* e *spam*. É possível perceber nessa figura que os valores de distâncias das amostras *nonspam* (dados em azul) apresentam padrão de comportamento inesperado, pois, o valor de distância entre as amostras de teste da classe *nonspam* e o centroide *nonspam* varia entre 4.00 e 4.66, enquanto a distância entre essas mesmas amostras e o centroide *spam* varia entre 1.80 e 5.00. Por outro lado, os valores de distância entre as amostras de teste da classe *nonspam* e o centroide *nonspam* apresentam menor dispersão, isto é, 4.00-4.66 contra 1.80-5.00 de *nonspam* vs centroide *spam*.

Também pode ser observado na Figura B.1 que a distância entre as amostras de teste do tipo *spam* em relação ao centroide *spam* (figuras em vermelho), apresenta comportamento esperado, isto é, novas amostras do tipo *spam* são mais similares ao centroide *spam* do que ao centroide *nonspam*. A variação dos valores das distâncias entre as amostras *spam* em relação ao centroide *spam* fica na faixa de 1.00 a 2.9, enquanto a distância dessas mesmas amostras ao centroide *nonspam* varia entre 4.00 e 5.00.

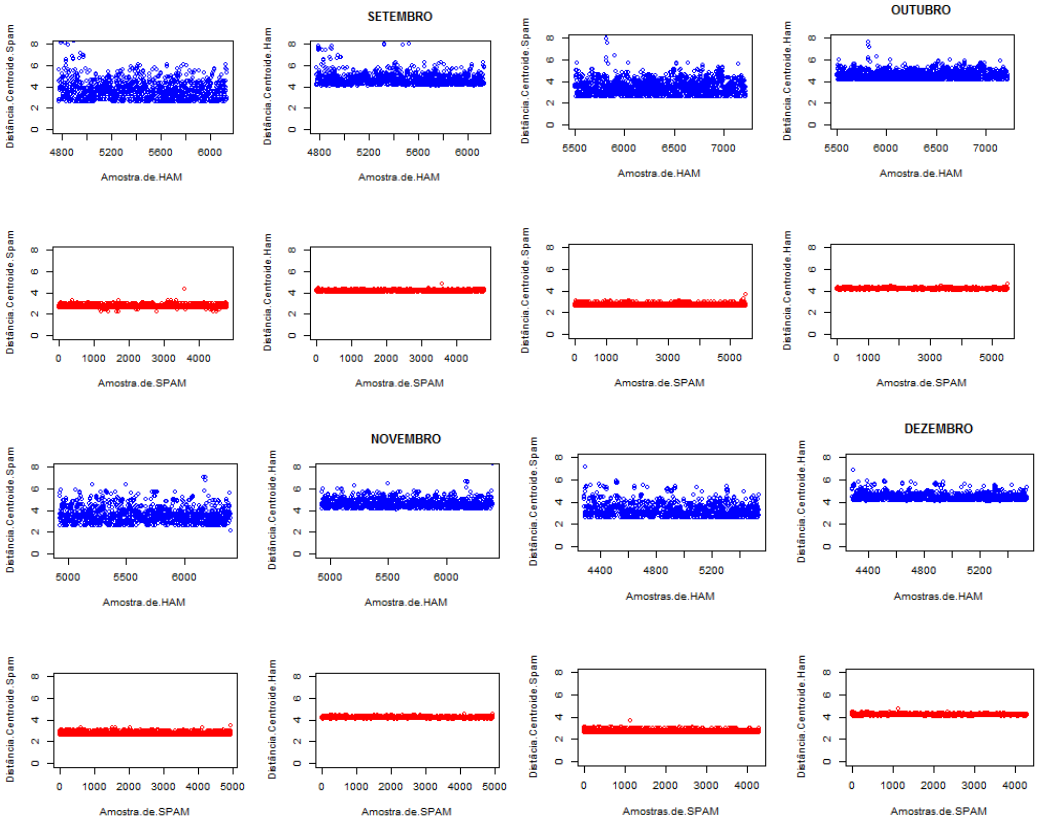

Figura B.1. Amostras de teste da classe Spam e *Nonspam* em relação aos respectivos centroides – IG

A Figura B.2 apresenta a mediana das distâncias entre as amostras de *nonspam* e *spam* em relação aos centroides *nonspam* e *spam*. As amostras são representadas por características selecionadas por IG. É possível observar que a distância entre as amostras de teste *spam* e o centroide *spam* tem aproximadamente metade dos valores inferiores ou iguais a 2.7, enquanto a outra metade tem valor superior ou igual à referida mediana. Já a distância do centroide *nonspam* às amostras *spam* exibe metade dos valores iguais ou inferiores a 4.1, enquanto a outra metade é superior ou igual à mediana. Porém, quanto às amostras *nonspam*, o centroide *nonspam* apresenta mais similaridade com amostras *spam* do que com amostras *nonspam*. Esse comportamento implica em uma taxa de falso positivo alta.

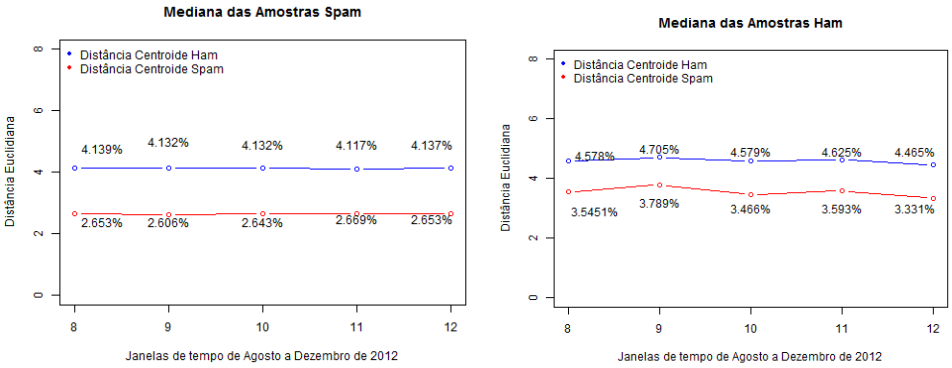

Figura B.2. Mediana em relação aos centroides *Nonspam* e Spam – IG

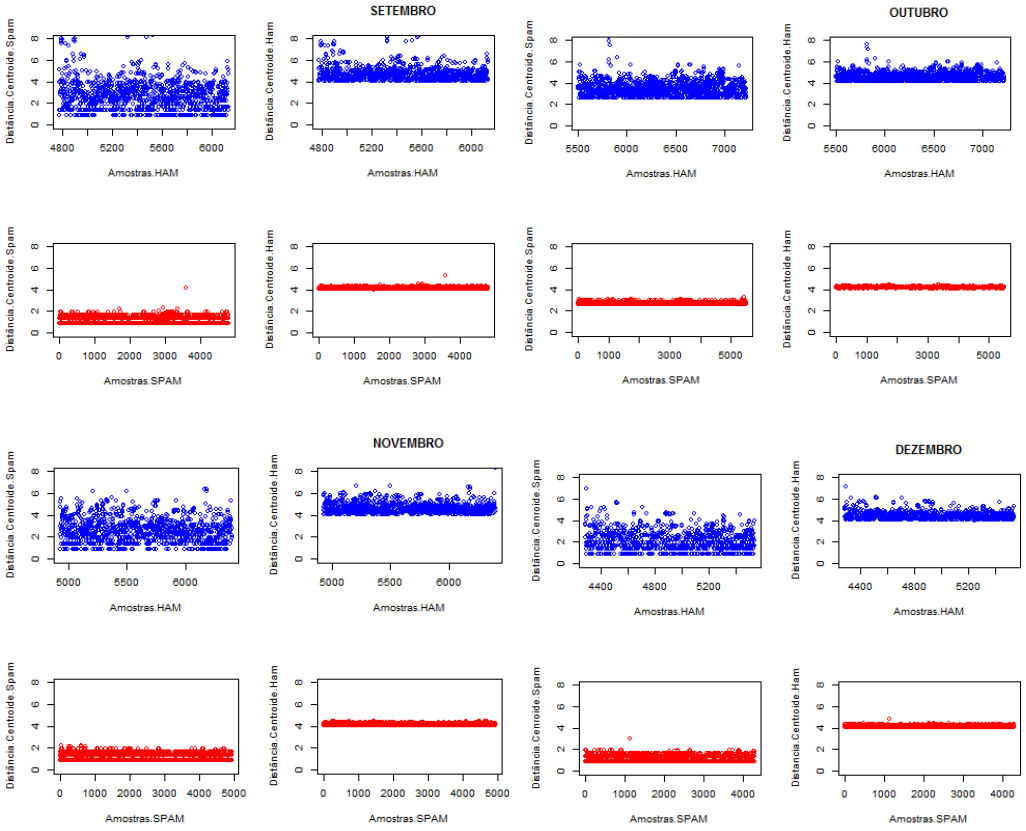

Figura B.3. Amostras de teste da classe Spam e *Nonspam* em relação aos respectivos centroides – Frequência

A Figura B.4 apresenta a mediana do valor das distâncias entre as amostras de teste das classes *nonspam* e *spam* e os centroides *nonspam* e *spam*, sendo que a frequência foi utilizada como método de seleção de atributos. É possível observar que a distância das amostras *spam* em relação ao centroide *spam* tem aproximadamente metade dos valores inferiores ou iguais a 1.1, enquanto a outra metade tem valores superiores ou iguais à

mediana. Em relação à distância do centroide *nonspam* às amostras *spam,* metade dos valores é igual ou inferior a 4.1, enquanto a outra metade é superior ou igual à mediana. Novamente, as amostras *spam* apresentam um comportamento mais esperado em relação à distância dos centroides *spam* e *nonspam*. Pode-se inferir, portanto, que o centroide *spam* possui uma composição de características que consegue representar bem ataques *spam,* fato que faz com que novas amostras *spam* sejam mais similares ao centroide *spam* do que ao centroide *nonspam*, bem como o valor da distância das novas mostras *nonspam* ao centroide *spam* mantém-se estável.

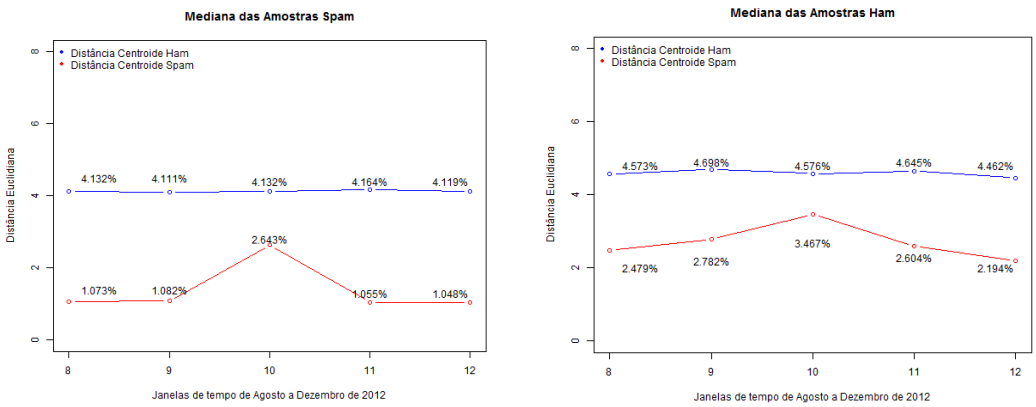

Figura B.4. Mediana em relação aos centroides *Nonspam* e Spam – Frequência

B.3. Considerações Finais

Os experimentos envolvendo o algoritmo k-*means,* cujo objetivo era detetar mudanças no ambiente de treino em relação ao ambiente de teste por meio do cálculo da similaridade entre os dois conjuntos de dados, mostraram que amostras *spam* são facilmente identificadas por meio da similaridade via distância euclidiana. Porém, amostras *nonspam* se aproximam mais do centroide *spam* do que do centroide *nonspam*. Esse comportamento demonstra novamente o quão difícil é identificar amostras *nonspam*, justamente por causa do enorme universo de características que representa essa classe, enquanto amostras *spam* apresentam um universo de características menor e mais constante. Esse comportamento indica que características *spam* são subconjuntos de características *nonspam*, ou seja, é fácil encontrar um conjunto de características *spam* em um vetor de características *nonspam.*