

**UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**UTILIZAÇÃO DE ÁRVORES DE DECISÃO PARA
APRIMORAR A CLASSIFICAÇÃO DE FRAGMENTOS**

JULIANO KAZUKI MATSUZAKI OYA

ORIENTADOR: BRUNO WERNECK PINTO HOELZ

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

**PUBLICAÇÃO: PPGENE.DM - 630 A/16
BRASÍLIA/DF: DEZEMBRO/2016**

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UTILIZAÇÃO DE ÁRVORES DE DECISÃO PARA APRIMORAR A
CLASSIFICAÇÃO DE FRAGMENTOS

Juliano Kazuki Matsuzaki Oya

DISSERTAÇÃO DE MESTRADO PROFISSIONAL SUBMETIDA AO DEPARTAMENTO DE
ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE
BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU
DE MESTRE.

APROVADA POR:


BRUNO WERNECK PINTO HOELZ, Dr., UNB
(ORIENTADOR)


CELIA GHEDINI RALHA, Dr., CIC/UNB
(EXAMINADORA INTERNA)


DÍBIO LEANDRO BORGES, Dr., CIC/UNB
(EXAMINADOR INTERNO)

Brasília, 16 de Dezembro de 2016.

FICHA CATALOGRÁFICA

OYA, JULIANO KAZUKI MATSUZAKI

Utilização de árvores de decisão para aprimorar a classificação de fragmentos.

[Distrito Federal] 2016.

xv, 85 p., 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2016).

Dissertação de Mestrado - Universidade de Brasília.

Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Classificação de fragmentos 2. Recuperação de arquivos

3. Aprendizado de máquina 4. Computação forense

I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

Oya, J. K. M. (2016). Utilização de árvores de decisão para aprimorar a classificação de fragmentos. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM-630/16, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 85p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Juliano Kazuki Matsuzaki Oya.

TÍTULO DA DISSERTAÇÃO DE MESTRADO: Utilização de árvores de decisão para aprimorar a classificação de fragmentos.

GRAU / ANO: Mestre / 2016

É concedida à Universidade de Brasília permissão para reproduzir cópias desta dissertação de Mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Do mesmo modo, a Universidade de Brasília tem permissão para divulgar este documento em biblioteca virtual, em formato que permita o acesso via redes de comunicação e a reprodução de cópias, desde que protegida a integridade do conteúdo dessas cópias e proibido o acesso a partes isoladas desse conteúdo. O autor reserva outros direitos de publicação e nenhuma parte deste documento pode ser reproduzida sem a autorização por escrito do autor.



Juliano Kazuki Matsuzaki Oya
Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica
70.910-900 - Brasília - DF - Brasil.

DEDICATÓRIA

À minha esposa e ao meu filho.

AGRADECIMENTOS

Agradeço, em primeiro lugar, à minha esposa. Também ao meu filho, que mesmo sem saber, manteve meu ânimo para concluir este trabalho.

Ao Prof. Dr. Bruno Hoelz, cujo compromisso como orientador foi essencial durante todo o desenvolvimento deste trabalho.

Aos colegas de trabalho do Instituto de Criminalística e aos colegas do Programa de Mestrado em Informática Forense pelo convívio agradável, pelas experiências e conhecimentos compartilhados.

Agradeço também ao Departamento Polícia Federal - DPF, ao Instituto de Criminalística da Polícia Civil do Distrito Federal, à Universidade de Brasília, à Fundação de Peritos em Criminalística Ilaraine Acácio Arce, à Associação Brasiliense de Peritos em Criminalística e aos gestores do Programa Nacional de Segurança Pública com Cidadania - PRONASCI, do Ministério da Justiça, por investirem em seus profissionais e acreditarem no potencial deles.

A todos, os meus sinceros agradecimentos.

Juliano K. M. Oya

RESUMO

UTILIZAÇÃO DE ÁRVORES DE DECISÃO PARA APRIMORAR A CLASSIFICAÇÃO DE FRAGMENTOS

Autor: Juliano Kazuki Matsuzaki Oya

Orientador: Bruno Werneck Pinto Hoelz

Programa de Pós-graduação em Engenharia Elétrica

Brasília, Dezembro de 2016

A classificação de fragmentos de arquivos é uma parte essencial do processo de recuperação de dados em computação forense. Métodos que dependem de assinaturas de arquivo ou de estruturas do sistema de arquivos são amplamente utilizados, mas outros métodos são necessários na ausência de tais informações. Para esse propósito, este trabalho descreve um método flexível para aprimorar a classificação de fragmentos e a recuperação de arquivos por meio da aplicação de árvores de decisão. Arquivos de evidências de casos forenses reais foram utilizados para gerar um conjunto de fragmentos de testes e de validação. Para cada fragmento, 46 atributos foram extraídos e utilizados no treinamento das árvores de decisão. Em média, os modelos resultantes classificaram corretamente 98,78% dos fragmentos em tarefas de classificação binários e de 86,05% em tarefas de classificação multinomiais. Os resultados mostram que as árvores de decisão podem ser aplicadas com sucesso para o problema de classificação de fragmentos e que apresentam bons resultados quando comparadas com outros métodos apresentados na literatura. Por conseguinte, o método proposto pode ser utilizado como um complemento aos métodos usuais de recuperação de arquivo, possibilitando um processo de recuperação de dados mais minucioso.

ABSTRACT

USING DECISION TREES TO IMPROVE FRAGMENT CLASSIFICATION

Author: Juliano Kazuki Matsuzaki Oya

Supervisor: Bruno Werneck Pinto Hoelz

Programa de Pos-graduação em Engenharia Eletrica

Brasília, December of 2016

The classification of file fragments is an essential part of the data recovery process in computer forensics. Methods that rely on file signatures or file system structures are widely employed, but other methods are required in the absence of such information. For this purpose, this paper describes a flexible method to improve fragment classification and recovery using decision trees. Evidence files from real forensic cases were used to generate the training and testing fragments. For each fragment, 46 features were extracted and used in the training of the decision trees. On average, the resulting models correctly classified 98.78% of the fragments in binary classification tasks and 86.05% in multinomial classification tasks. These results show that decision trees can be successfully applied to the fragment classification problem and they yield good results when compared to other methods presented in the literature. Consequently, the proposed method can be used as a complement to the usual file recovery methods, allowing for a more thorough data recovery process.

SUMÁRIO

1 INTRODUÇÃO	1
1.1 Justificativa	3
1.2 Objetivos	3
1.3 Contribuições	3
1.4 Metodologia	4
1.5 Organização do trabalho	4
2 RECUPERAÇÃO DE ARQUIVOS	5
2.1 Noções de sistema de arquivos	5
2.2 Unidade de dado do sistema de arquivos	7
2.3 Recuperação de arquivos	8
2.3.1 Recuperação com metadados	9
2.3.2 Recuperação com <i>data carving</i>	10
2.4 Fragmentação de dados	13
3 CLASSIFICAÇÃO DE FRAGMENTOS	17
3.1 Aprendizagem automática e classificação de fragmento	17
3.2 Atributos dos fragmentos para a aprendizagem automática	19
3.2.1 Atributo de frequência de caracteres ASCII	21
3.2.2 Atributo de percentual de codificação em <i>Base64</i> e em <i>Base85</i> .	21
3.2.3 Atributos estatísticos de média, variação e correlação dos <i>bytes</i> .	21
3.2.4 Atributo de teste de qui-quadrado dos valores dos <i>bytes</i>	22
3.2.5 Atributo de cálculo de <i>Monte Carlo Pi</i>	23
3.2.6 Atributo de percentuais de padrões 0xF8, 0xFF e 0x00	24
3.2.7 Atributo de peso de <i>Hamming</i>	24
3.2.8 Atributo sobre o histograma dos <i>bytes</i>	24
3.2.9 Atributo de vetor de histograma dos <i>bytes</i>	25
3.2.10 Atributo de linguagem utilizada no texto	25
3.2.11 Atributos de processamento de linguagem natural	25
3.2.12 Atributo de linguagem de programação	25
3.2.13 Atributo de entropia dos <i>bytes</i>	26
3.2.14 Atributo de percentual de uso de <i>tag</i>	27
3.2.15 Atributo de taxa de compressão do fragmento	27
3.2.16 Atributo de frequência circular dos <i>bytes</i>	27

3.2.17	Atributo de vetor de escala de cinza	27
3.2.18	Atributo de complexidade <i>Kolmogorov</i>	28
3.2.19	Atributo de subsequência comum mais longa de <i>bytes</i>	28
3.2.20	Atributos de distância de compressão normalizada	28
4	MODELO DE CLASSIFICAÇÃO	29
4.1	Noções de árvores de decisão	29
4.2	Construção de árvores de decisão	30
4.3	Modelo baseado em árvores de decisão	33
4.4	Conjunto de atributos utilizados no modelo	36
5	MÉTODO DE TRABALHO	38
5.1	Descrição do método de trabalho	38
5.1.1	Seleção dos arquivos	38
5.1.2	Tamanhos dos fragmentos	39
5.1.3	Extração dos fragmentos	40
5.1.4	Extração dos atributos	41
5.1.5	Treinamento dos classificadores	42
5.1.6	Validação dos classificadores	42
5.1.7	Tratamento dos atributos	43
5.2	Trabalhos correlatos	45
6	EXPERIMENTOS E RESULTADOS	47
6.1	Execução do experimento	47
6.1.1	Desenvolvimento do protótipo	47
6.1.2	Sobre o formato de arquivo ARFF	48
6.1.3	Seleção do algoritmo de classificação	49
6.1.4	Seleção dos arquivos	50
6.1.5	Extração dos fragmentos e dos atributos	50
6.1.6	Produção dos classificadores	52
6.1.7	Tratamento dos atributos	53
6.2	Resultados obtidos	54
6.2.1	Classificadores binomiais	55
6.2.2	Classificadores multinomiais	57
6.2.3	<i>Ranking</i> dos atributos	60
6.2.4	Aplicação dos filtros	61
6.2.5	Considerações finais	63
7	CONCLUSÕES	65

REFERÊNCIAS BIBLIOGRÁFICAS	67
APÊNDICES	71
A RESUMO DE TRABALHOS RELACIONADOS	72
B ATRIBUTOS DOS CLASSIFICADORES	74
C CLASSIFICADORES BINOMIAIS	76
D CLASSIFICADORES MULTINOMIAIS	78

LISTA DE TABELAS

1.1	Resultado da classificação baseada em assinatura dos fragmentos.	2
4.1	Descrição dos atributos extraídos de cada fragmento.	37
6.1	Descrição dos tipos de arquivo selecionados para o experimento.	51
6.2	Quantitativos dos arquivos selecionados para o experimento.	51
6.3	Comparativo dos percentuais de classificações corretas do classificador binomial.	56
6.4	Comparativo dos percentuais de classificação correta do classificador multinomial.	58
6.5	Matriz de confusão da classificação multinomial do algoritmo J48 (4 <i>Kbytes</i>).	59
6.6	Relação dos atributos com maiores valores de <i>ranking</i> no classificador multinomial, por tamanho de fragmento.	60
6.7	Comparativo dos métodos e resultados dos experimentos de outros autores.	63
A.1	Comparativo dos atributos de fragmentos de arquivo utilizados pelos autores.	72
A.2	Comparativo dos tipos de arquivo utilizados pelos autores.	73
B.1	Resultado do <i>ranking</i> dos atributos para o algoritmo de classificação J48. Resultado ordenado por posição do atributo no <i>ranking</i>	74
B.2	Resultado do <i>ranking</i> dos atributos para o algoritmo de classificação J48. Resultado ordenado por nome do atributo.	75
C.1	Percentuais de classificação correta do classificador binomial (resultados sem a aplicação dos filtros, resultado com a normalização dos atributos).	76
C.2	Percentuais de classificação correta do classificador binomial (resultados com a discretização dos atributos, resultado com a redução dos atributos).	77
D.1	Percentuais de classificação correta do classificador multinomial (resultados sem a aplicação dos filtros, resultado com a normalização dos atributos).	78
D.2	Percentuais de classificação correta do classificador multinomial (resultados com a discretização dos atributos, resultado com a redução dos atributos).	79

D.3	Matriz de confusão do classificador multinomial para fragmentos de 1 <i>Kbyte</i> com dados não normalizados e não discretizados.	80
D.4	Matriz de confusão do classificador multinomial para fragmentos de 2 <i>Kbytes</i> com dados não normalizados e não discretizados.	80
D.5	Matriz de confusão do classificador multinomial para fragmentos de 4 <i>Kbytes</i> com dados não normalizados e não discretizados.	81
D.6	Matriz de confusão do classificador multinomial para fragmentos de 1 <i>Kbyte</i> com dados normalizados e não discretizados.	81
D.7	Matriz de confusão do classificador multinomial para fragmentos de 2 <i>Kbytes</i> com dados normalizados e não discretizados.	82
D.8	Matriz de confusão do classificador multinomial para fragmentos de 4 <i>Kbytes</i> com dados normalizados e não discretizados.	82
D.9	Matriz de confusão do classificador multinomial para fragmentos de 1 <i>Kbyte</i> com dados normalizados e discretizados.	83
D.10	Matriz de confusão do classificador multinomial para fragmentos de 2 <i>Kbytes</i> com dados normalizados e discretizados.	83
D.11	Matriz de confusão do classificador multinomial para fragmentos de 4 <i>Kbytes</i> com dados normalizados e discretizados.	84
D.12	Matriz de confusão do classificador multinomial para fragmentos de 1 <i>Kbyte</i> com dados normalizados, discretizados e com redução de atributos.	84
D.13	Matriz de confusão do classificador multinomial para fragmentos de 2 <i>Kbytes</i> com dados normalizados, discretizados e com redução de atributos.	85
D.14	Matriz de confusão do classificador multinomial para fragmentos de 4 <i>Kbytes</i> com dados normalizados, discretizados e com redução de atributos.	85

LISTA DE FIGURAS

2.1	Interação entre as cinco categorias de dados (adaptada de Carrier (2005)).	6
2.2	Um exemplo de volume no qual o sistema de arquivos é endereçado para grupos de dois setores e parte dos setores não são endereçados (adaptada de Carrier (2005)).	7
2.3	Exemplo de alocação de cinco unidades de dado para dois arquivos (adaptada de Carrier (2005)).	8
2.4	Exemplo de desalocação de entradas de metadado (adaptada de Carrier (2005)).	9
2.5	Processo de visualização do conteúdo de uma entrada de metadado (adaptada de Carrier (2005)).	10
2.6	Utilização das informações das entradas de metadado e de unidades de dado para recuperar o conteúdo do arquivo (adaptada de Carrier (2005)).	10
2.7	Blocos de dados são analisados para extrair um arquivo JPG (adaptada de Carrier (2005)).	12
2.8	Visão geral dos algoritmos de <i>carving</i> (adaptada de Cohen (2007)).	13
2.9	Exemplo de uma função de mapeamento (adaptada de Cohen (2007)).	14
2.10	Possibilidades de mapeamento na ocorrência de fragmentação (adaptada de Cohen (2007)).	16
3.1	Tipos de atributo extraídos dos fragmentos.	20
3.2	Exemplo de cálculo de <i>Monte Carlo Pi</i> .	23
4.1	Exemplo de uma árvore de decisão.	30
4.2	Modelo de recuperação de arquivos com utilização dos classificadores.	33
4.3	Visão geral do processo para produzir os modelos de classificação.	34
4.4	Aplicação dos modelos de classificadores binomiais e multinomiais.	35
4.5	Modelo de reconstrução de arquivos.	36
5.1	Visão geral das fases do método de trabalho.	38
5.2	Visão geral das fases seleção e fragmentação de arquivo.	39
5.3	Visão geral da fragmentação de arquivos.	40
5.4	Visão geral da fase de extração dos atributos dos fragmentos.	41
5.5	Visão geral da fase de treinamento e validação dos classificadores.	42
5.6	Exemplo de validação cruzada do tipo <i>5-fold</i> (adaptada de James et al. (2013)).	43

6.1	Avaliação de desempenho dos algoritmos de árvores de decisão	50
6.2	Visão geral do <i>workflow</i> de aplicação dos filtros de normalização, discretização e seleção de atributo.	54
6.3	Gráfico com os resultados das classificações corretas do classificador binomial, sem aplicação de filtros, por tipos de arquivo e por tamanhos de fragmento.	55
6.4	Gráfico com os resultados das classificações corretas do classificador multinomial, sem aplicação de filtros, por tipos de arquivo e por tamanhos de fragmento.	57
6.5	Gráfico com os custos de tempo para gerar o classificador binomial a cada aplicação de filtros.	61
6.6	Gráfico com os resultados das classificações corretas do classificador binomial após a aplicação dos filtros.	62
6.7	Gráfico com os resultados das classificações corretas do classificador multinomial após a aplicação dos filtros.	62

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

API: Application Programming Interface.

ARFF: Attribute-Relation File Format.

CD: Consolidador de Dado.

CRISP-DM: CRoss Industry Standard Process for Data Mining.

CSV: Comma-Separated Values.

EA: Extrator de Atributos.

Ext2: Second Extended File System.

Ext3: Third Extended File System.

FAT: File Allocation Table.

FA: Fragmentador de Arquivo.

KNN: k-Nearest Neighbor.

MD5: Message-Digest Algorithm 5.

MFT: Master File Table.

MSE: Mean Squared Error.

NTFS: New Technology File System

PAD: Processador de Árvores de Decisão

SVM: Support Vector Machines.

TD: Tratador de Dado.

UFS: Unix File System.

WEKA: Waikato Environment for Knowledge Analysis.

1 INTRODUÇÃO

A classificação de fragmentos é uma importante atividade na decodificação de fragmentos de memória, análise de fragmentos de arquivo em tráfegos de rede, detecção de *software* maliciosos, entre outras (VEENMAN, 2007; ROUSSEV; QUATES, 2013). No caso específico da computação forense, a classificação de fragmentos é uma atividade essencial e indispensável para o propósito de recuperação de arquivos armazenados em mídias sem informações sobre o seu sistema de arquivos (VEENMAN, 2007; LI et al., 2011; FITZGERALD et al., 2012; XIE; ABDULLAH; SULAIMAN, 2013; XU et al., 2014).

O problema da classificação de fragmentos consiste em determinar o tipo do arquivo ao qual pertence o fragmento. Em situações em que o fragmento é parte do cabeçalho ou do final do arquivo, a classificação é uma tarefa mais simples, visto que a maioria dos arquivos armazena assinaturas ou estruturas de dados que indicam o seu tipo nas áreas iniciais ou finais do arquivo. Entretanto, em situações em que os fragmentos são partes do meio do arquivo, onde há pouca ou nenhuma informação que identifiquem o seu tipo, a classificação torna-se uma tarefa desafiadora (VEENMAN, 2007; CALHOUN; COLES, 2008; LI et al., 2011).

A técnica mais utilizada para a classificação e recuperação de arquivos – inclusive fragmentos de arquivo – é por meio da busca por assinaturas conhecidas, as quais ficam localizadas no início e no final de um arquivo. Entretanto, essa técnica tem como pressupostos que o arquivo possui o cabeçalho e o final do arquivo e, ainda, que o arquivo foi alocado de forma contínua na mídia de armazenamento. Como mostrado por Cohen (2007), nem sempre esses pressupostos estão presentes nas mídias de armazenamento.

A fim de verificar a aplicabilidade da técnica de recuperação de arquivos baseadas em assinaturas, foi realizado um teste sobre um conjunto de fragmentos de arquivo sem as informações de cabeçalho e de final do arquivo (o mesmo conjunto de fragmentos, que será apresentado no Capítulo 6, foi utilizado para esse teste). Para tanto, foi utilizado o arquivo de perfis do Foremost (2016), que é uma ferramenta *Linux* tipicamente utilizada para recuperar arquivos, com 80 assinaturas de 40 tipos de arquivo para classificar

os fragmentos de acordo com as assinaturas encontradas dentro dos fragmentos. O resultado, apresentado na Tabela 1.1, são percentuais de acertos de 0,54%, 0,92% e 1,25% para a classificação dos fragmentos, o que mostra que a técnica não pode ser aplicada na recuperação de fragmentos sem as informações de cabeçalho e de final do arquivo.

Tabela 1.1: Resultado da classificação baseada em assinatura dos fragmentos.

	FRAGMENTOS DE 1 KBYTES	FRAGMENTOS DE 2 KBYTES	FRAGMENTOS DE 4 KBYTES
Número de fragmentos	1.215.000	1.215.000	1.210.160
Número de acertos	6.661	11.284	15.245
Número de erros	1.208.339	1.203.716	1.194.915
Não classificados	816.267	696.807	530.402
Percentual de acertos	0,54%	0,92%	1,25%

Alternativamente ao uso de assinaturas na classificação de fragmentos, vários trabalhos foram realizados com a aplicação de técnicas de aprendizagem automática para resolver o problema de classificação de fragmentos (VEENMAN, 2007; CALHOUN; COLES, 2008; AXELSSON, 2010; CONTI et al., 2010; LI et al., 2011; FITZGERALD et al., 2012; XIE; ABDULLAH; SULAIMAN, 2013; XU et al., 2014).

A aprendizagem automática, aplicada na classificação de fragmentos de arquivo, consiste em construir um modelo computacional a partir de dados de atributos de fragmentos cujos tipos são conhecidos e, posteriormente, utilizar o modelo construído para predizer ou classificar novos fragmentos cujos tipos são desconhecidos.

Nos trabalhos citados, de modo geral, a aplicação de técnicas de aprendizagem automática segue os seguintes passos: coleta de arquivos, fragmentação dos arquivos, extração dos atributos dos fragmentos, consolidação dos atributos em dados de treinamento e validação, treinamento do modelo de classificação e, por fim, a avaliação do modelo.

A proposta apresentada neste trabalho inova ao considerar os atributos utilizados por outros pesquisadores e, ainda, novos atributos extraídos a partir dos fragmentos; ao utilizar uma base de dados de 2.830.472 fragmentos extraídos de arquivos de 21 tipos diferentes; ao utilizar a aprendizagem automática baseada em árvores de decisão, cuja precisão média na classificação dos fragmentos se mostrou promissora.

1.1 Justificativa

A classificação de fragmentos de arquivo tem um papel fundamental na recuperação de arquivos, especialmente quando não há informações de metadados ou quando há fragmentação de arquivos. Nesses cenários, as técnicas tradicionais, como o *data carving*, podem ter dificuldades em recuperar um arquivo de interesse.

Portanto, há a necessidade de um modelo de classificação de fragmentos que permita identificar um conjunto de fragmentos de diferentes tipos e tamanhos, a fim de auxiliar a remontagem e a posterior recuperação de arquivos de interesse. Ademais, há a necessidade de um método de trabalho para produzir os modelos de classificadores baseados em atributos que são extraídos dos fragmentos.

1.2 Objetivos

O objetivo primário deste trabalho é apresentar um modelo de classificador de fragmentos de arquivo flexível para diversos tipos e tamanhos de fragmentos. O modelo de classificador deve ser baseado em dados de atributos extraídos dos fragmentos. Dessa forma, um objetivo adicional do trabalho é apresentar os tipos de atributo extraíveis dos fragmentos. Além disso, torna-se necessário a especificação de um método de trabalho para produzir o modelo.

1.3 Contribuições

Ao buscar os objetivos deste trabalho, algumas contribuições serão produzidas, tais como:

- um modelo de classificação de fragmentos de arquivo flexível quanto aos tipos de arquivo, tamanhos dos fragmentos e atributos dos fragmentos;
- um método de trabalho para: seleção dos arquivos, extração dos fragmentos dos arquivos, extração dos atributos dos fragmentos e treinamento e validação dos classificadores;
- um conjunto de atributos de fragmentos para serem utilizados no modelo de classificação de fragmentos de arquivo;
- o uso de aprendizagem automática baseado em árvores de decisão para a classificação de fragmentos.

1.4 Metodologia

Para a realização deste trabalho foram desenvolvidas diversas atividades. Inicialmente foi realizada a revisão bibliográfica sobre os processos de classificação de fragmentos e de recuperação de arquivos.

Foram identificadas diversas abordagens utilizando técnicas de aprendizagem automática, as quais foram estudadas. Experimentos iniciais mostraram que o uso de técnicas baseadas em árvores de decisão têm potencial para realizarem, de forma eficiente, a classificação de fragmentos.

A revisão bibliográfica permitiu, também, identificar os diversos atributos de fragmentos que podem ser utilizados nos modelos de classificadores. Nesse sentido, foi definido um modelo baseado em aprendizagem automática supervisionada, no qual são fornecidos para a máquina exemplos de *inputs* e *outputs* de dados dos atributos dos fragmentos de arquivo.

Para a avaliação do modelo, foram realizados diversos experimentos com os dados de atributos extraídos de fragmentos de arquivo. A execução de cada experimento foi guiada por um método de trabalho. Tal método é descrito e apresentado no no Capítulo 5.

Por fim, com base na análise dos resultados obtidos e na comparação com trabalhos correlatos, foram produzidas as conclusões e as indicações de trabalhos futuros.

1.5 Organização do trabalho

O restante do trabalho está organizado da seguinte forma: no Capítulo 2 são apresentados os principais conceitos relacionados à recuperação de arquivos; no Capítulo 3 são apresentados os principais conceitos relacionados à classificação de fragmentos de arquivo por meio de técnicas de aprendizagem automática; no Capítulo 4 é apresentado um modelo de classificação de fragmentos baseado em árvores de decisão; no Capítulo 5 é descrita a metodologia de trabalho para a realização dos experimentos; no Capítulo 6 são descritos os experimentos realizados, onde são relatadas a seleção dos arquivos, a extração dos fragmentos, a extração dos atributos, a classificação dos fragmentos e o resultado obtido; no Capítulo 7 são apresentadas a conclusão e a indicação de trabalhos futuros.

2 RECUPERAÇÃO DE ARQUIVOS

Este capítulo é dividido em 4 seções. A Seção 2.1 apresenta os principais conceitos relacionados aos sistemas de arquivos. Na Seção 2.2 são descritas as unidades de dado do sistema de arquivos. Na Seção 2.3 são descritas as duas principais técnicas de recuperação de arquivos: com metadados e com o *data carving*. Na Seção 2.4 é discutido o problema da fragmentação de dados para a recuperação de arquivos.

2.1 Noções de sistema de arquivos

Um sistema de arquivos consiste em dados estruturais e em dados de usuário, os quais são organizados de tal forma que o computador saiba onde encontrá-los (CARRIER, 2005). A maioria dos sistemas operacionais possuem seu próprio sistema de arquivos e, na maioria dos casos, o sistema de arquivos é independente da arquitetura do computador que o utiliza.

Carrier (2005) apresenta um modelo de referência para que os diferentes sistemas de arquivos possam ser mais facilmente comparados, tal como a comparação dos sistemas de arquivos FAT (*File Allocation Table*) e Ext3 (*third extended filesystem*). Ainda, um modelo de referência torna mais fácil para determinar onde uma evidência de interesse pode ser localizada. O modelo descrito por Carrier (2005) é dividido em cinco categorias de dados, quais sejam: *sistema de arquivos*, *conteúdo*, *metadados*, *nome de arquivo* e *aplicação*. A Figura 2.1, mostra as interações entre as categorias de dados.

Para Carrier (2005), todos os dados em um sistema de arquivos pertencem a uma das cinco categorias, de acordo com o papel que desempenham no sistema de arquivos. Nos parágrafos seguintes, cada uma das categorias são descritas segundo a visão de Carrier (2005).

A *categoria de sistema de arquivos* contém as informações gerais do sistema de arquivos. Todos os sistemas de arquivos têm uma estrutura geral para eles, mas cada instância de um sistema de arquivos é única, pois cada instância possui tamanho e configuração de desempenho próprios.

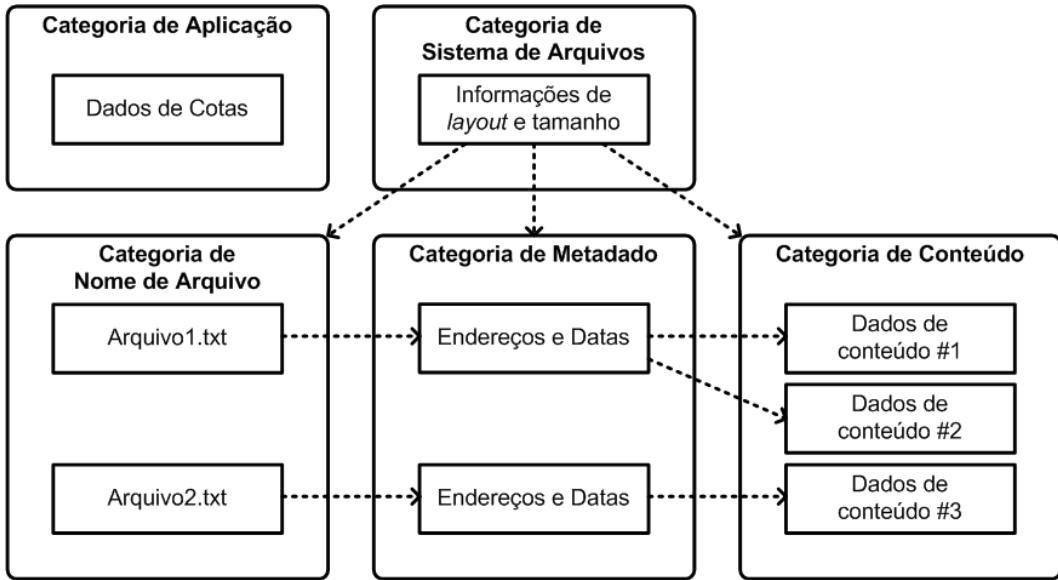


Figura 2.1: Interação entre as cinco categorias de dados (adaptada de Carrier (2005)).

A *categoria de aplicação* contém dados que fornecem características especiais. Estes dados não são necessários durante o processo de leitura ou escrita de um arquivo e, em muitos casos, não necessitam de ser incluídos na especificação do sistema de arquivos. Exemplos de dados nesta categoria incluem as estatísticas de cotas de usuário e o *journal* do sistema de arquivos.

A *categoria de nome de arquivo* contém os dados que atribuem um nome para cada arquivo. Na maioria dos sistemas de arquivos, esses dados estão localizados no conteúdo de um diretório e são uma lista de nomes de arquivos com os endereços de metadados correspondentes.

A *categoria de metadado* contém os dados que descrevem um arquivo, ou seja, eles são os dados que descrevem os dados. Esta categoria contém informações tais como o endereço onde o conteúdo do arquivo é armazenado, o tamanho do arquivo, a data e hora em que o arquivo foi acessado e as informações de controle de acesso. Note que esta categoria não contém o conteúdo do arquivo e, ainda, pode não conter o nome do arquivo. Exemplos de estruturas de dados nesta categoria incluem as entradas de diretórios do FAT, a entrada da MFT (*Master File Table*) e as estruturas de *inode* do UFS (*Unix File System*) e Ext3 (*Third Extended Filesystem*).

A *categoria de conteúdo* contém os dados que compõem o conteúdo real de um arquivo. A maioria dos dados em um sistema de arquivos pertence a esta categoria, e é normalmente organizada em um conjunto de contêineres de tamanho padrão. Cada

sistema de arquivos atribui um nome diferente para os contêineres, como *clusters* e blocos. Neste trabalho, o termo unidade de dado será utilizado para referenciar um contêiner de dado desta categoria.

2.2 Unidade de dado do sistema de arquivos

Os dados da categoria de conteúdo são normalmente organizados em grupos de tamanhos iguais, os quais são chamados de unidades de dado por Carrier (2005). Em sistemas de arquivos específicos, as unidades de dado recebem outros nomes, tais como *clusters* (NTFS) ou blocos (Ext3). Independentemente do tipo de sistema de arquivos, uma unidade de dado pode assumir um estado de alocado ou de não alocado. Normalmente há algum tipo de estrutura de dados que controla o estado da alocação de cada unidade de dado.

A Figura 2.2, adaptada de Carrier (2005), mostra um volume com 17 setores e seus endereços lógicos. Na mesma figura é indicado o mapeamento dos endereços lógicos do volume para os endereços lógicos do sistema de arquivos. Neste sistema de arquivos fictício, cada unidade de dado utiliza um par de setores e, ainda, não faz uso de alguns setores iniciais e finais do volume.

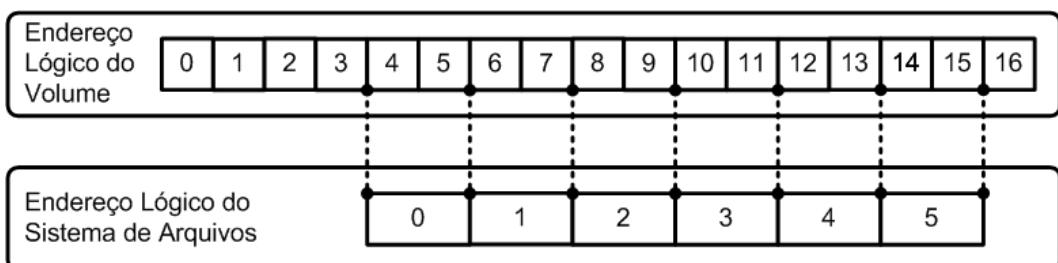


Figura 2.2: Um exemplo de volume no qual o sistema de arquivos é endereçado para grupos de dois setores e parte dos setores não são endereçados (adaptada de Carrier (2005)).

O controle das unidades não alocadas é essencial para que o sistema operacional as localize para o armazenamento de novos arquivos ou expansão de arquivos existentes. Já quando o arquivo é excluído, as unidades de dado utilizadas por esse arquivo devem ser definidas como não alocadas, para que possam ser utilizadas por novos arquivos. Por padrão, como acrescenta Carrier (2005), a maioria dos sistemas operacionais não excluem o conteúdo das unidades de dado marcadas como não alocadas.

Um sistema operacional pode utilizar diferentes estratégias de alocação de unidades de dado e, dentre elas, uma tipicamente empregada é a alocação de unidades

consecutivas. Entretanto, nem sempre é possível a alocação consecutiva de unidades de dado para um arquivo, ocorrendo, nesse caso, a fragmentação dos dados.

Uma unidade de dado que está alocada para um arquivo também possui um endereço lógico de arquivo, que é um endereço relativo ao início do arquivo. Por exemplo, se um arquivo é alocado em duas unidades de dado, a primeira e a segunda unidade de dado possuirão os endereços lógicos de arquivo iguais a 0 e 1, respectivamente. As informações de mapeamento dos endereços lógicos do sistema de arquivos e do arquivo são armazenadas nas entradas de metadado. Pode-se observar um exemplo de mapeamento de endereços lógicos na Figura 2.3, na qual dois arquivos são alocados para cinco unidades de dado.

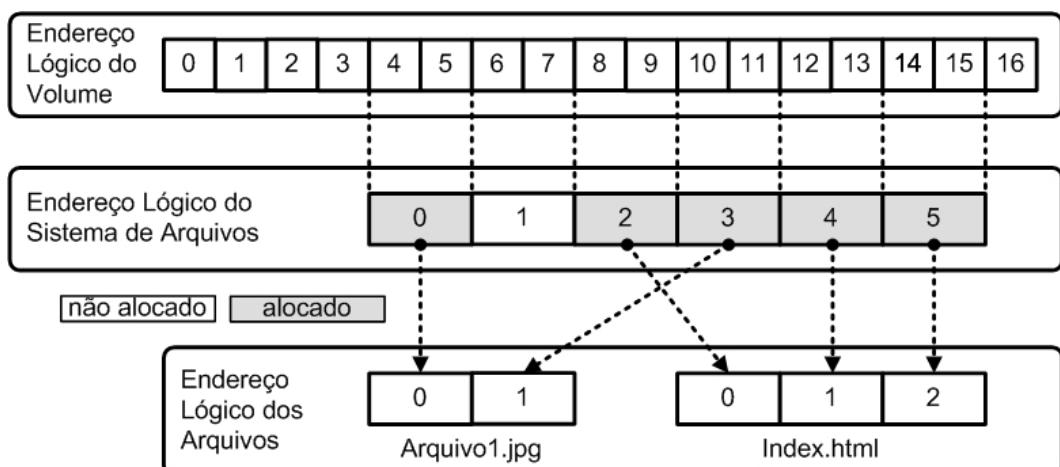


Figura 2.3: Exemplo de alocação de cinco unidades de dado para dois arquivos (adaptada de Carrier (2005)).

Ainda na Figura 2.3, pode-se observar que um endereço específico do sistema de arquivos não é alocado para algum arquivo; e que ocorreu a fragmentação dos dois arquivos, ou seja, os endereços lógicos do sistema de arquivos não foram alocados em sequência.

2.3 Recuperação de arquivos

A recuperação de arquivos é uma atividade essencial na busca de evidências digitais, especialmente aquelas que foram excluídas do sistema de arquivos. Carrier (2005) apresenta dois principais métodos para a recuperação de arquivos apagados: baseados em metadados e baseados em aplicação. O método baseado em aplicação é tipicamente conhecido como *data carving* (COHEN, 2007).

2.3.1 Recuperação com metadados

A recuperação baseada em metadados funciona quando os metadados do arquivo excluído ainda existem. Por exemplo, a Figura 2.4 (A) mostra o caso em que a entrada não alocada de metadados ainda possui os seus endereços de unidade de dado e, portanto, pode ter o conteúdo do arquivo facilmente recuperado. A Figura 2.4 (B), entretanto, mostra que a entrada de metadado foi desalocada e, ainda, teve os seus endereços dos dados excluídos.

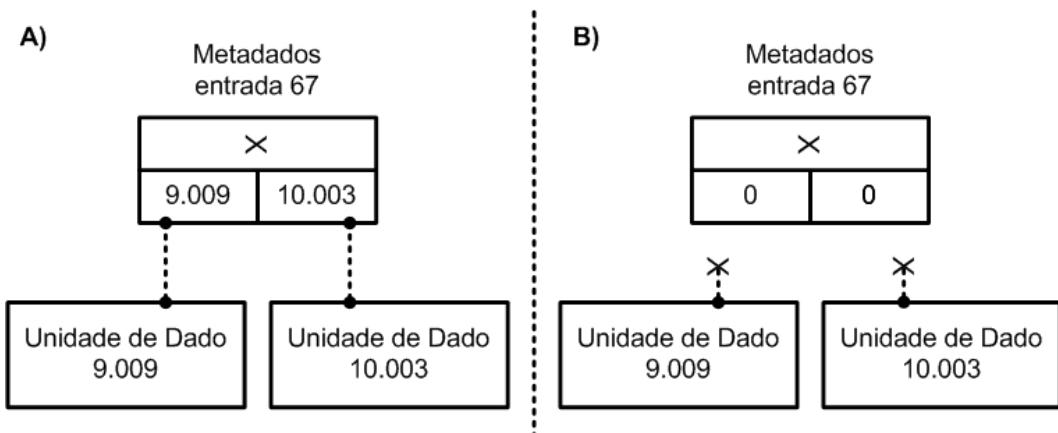


Figura 2.4: Exemplo de desalocação de entradas de metadado (adaptada de Carrier (2005)).

Carrier (2005) alerta para o cuidado ao se realizar a recuperação baseada em metadados, já que pode não haver a sincronia entre os dados da estrutura dos metadados e das unidades de dado (caso em que as unidades de dado são alocadas para novos arquivos). Considerando ainda o exemplo da Figura 2.4 (A), caso houvesse a realocação da unidade de dado 9.009 para outro arquivo, a entrada de metadados 67 ainda permaneceria apontado para a unidade de dado 9.009 e, portanto, a recuperação do arquivo através da entrada 67 resultaria em um arquivo inválido.

Em muitos casos, é necessário o processamento e análise dos metadados que pode apontar para uma estrutura de metadado específica e apresentar mais detalhes sobre o arquivo (CARRIER, 2005). O procedimento exato para esta técnica é dependente do sistema de arquivos, cujos metadados podem ser localizados em locais distintos. Isso pode ser observado na Figura 2.5, onde o sistema de arquivos tem estruturas de metadado localizados na unidade de dado 371, que, após o processamento, mostra o conteúdo de duas entradas de metadado, sendo uma entrada de um arquivo excluído e outra de um diretório alocado.

Após a análise dos metadados, pode-se visualizar o conteúdo do arquivo através

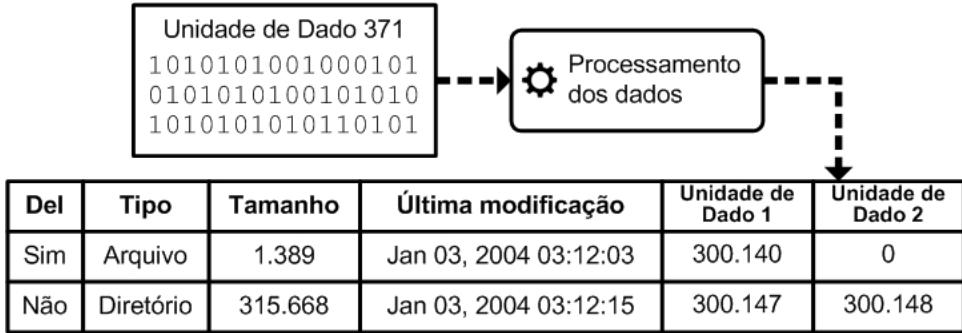


Figura 2.5: Processo de visualização do conteúdo de uma entrada de metadado (adaptada de Carrier (2005)).

da leitura das unidades de dado alocadas para esse arquivo. Esse processo ocorre nos dados das categorias de metadado e de conteúdo; através da técnica de pesquisa de metadado para localizar as unidades de dado alocadas para o arquivo e, em seguida, da técnica de visualização de conteúdo para encontrar o conteúdo real. Pode-se observar isso no exemplo da Figura 2.6, na qual as unidades de dado atribuídas para as entradas de metadado 1 e 2 são identificadas.

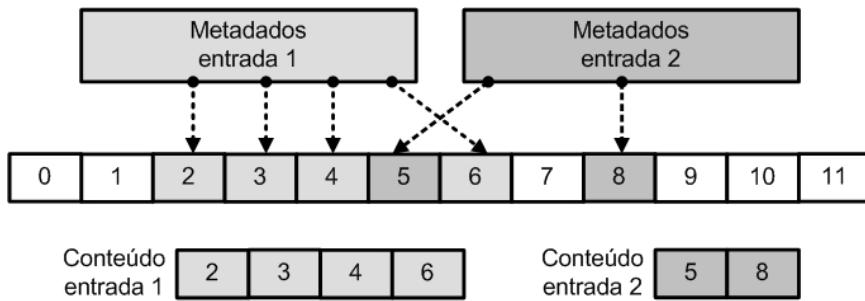


Figura 2.6: Utilização das informações das entradas de metadado e de unidades de dado para recuperar o conteúdo do arquivo (adaptada de Carrier (2005)).

Entretanto, as informações de metadado para a recuperação do arquivo podem inexistir, caso o metadado seja excluído ou caso a estrutura do metadado seja realocada para um novo arquivo (VEENMAN, 2007). Nesse caso, é indicado o uso do método baseado em aplicação (*data carving*) (CARRIER, 2005).

2.3.2 Recuperação com *data carving*

Data carving é o termo mais usado frequentemente para indicar o ato de recuperar um ou mais arquivos de imagens forenses digitais não estruturadas. O termo não estruturado indica que a imagem digital original não contém informação útil do sistema de arquivos que pode ser utilizada para ajudar na recuperação dos arquivos (COHEN, 2007).

Assim, *data carving* é um processo em que são feitas buscas por assinaturas que correspondem ao início e fim dos tipos de arquivo conhecidos. Por exemplo, uma imagem JPEG tem valores de cabeçalho e de rodapé padrões. Para recuperar um arquivo de imagem apagada, pode-se extrair o espaço não alocado do disco e executar uma ferramenta de *data carving* que realize a análise do cabeçalho JPEG e, em seguida, extraia os dados entre o cabeçalho e o final do arquivo (CARRIER, 2005).

O resultado deste processo de análise é uma coleção de arquivos que contêm pelo menos uma das assinaturas pesquisadas. Esse processamento geralmente ocorre em espaços não alocados pelo sistema de arquivos para recuperar arquivos que não possuem entradas de metadado vinculados à eles. Portanto, como explica Garfinkel (2007), o *data carving* reconstrói os arquivos com base em seu conteúdo, em vez de usar metadados que aponta para o mesmo conteúdo.

Dessa forma, o *data carving* é útil tanto para a recuperação de dados como para a análise forense. Para a recuperação de dados, o *data carving* pode recuperar arquivos de um dispositivo que tenha sido danificado como, por exemplo, um disco rígido onde os setores que contêm diretório ou *Master File Table* do disco já não são legíveis. Na análise forense, o *data carving* pode recuperar arquivos que foram apagados e tiveram suas entradas de diretório realocados para outros arquivos, mas ainda não tiveram os setores de dados substituídos.

Uma ferramenta que realiza o *data carving* é o *Foremost* (FOREMOST, 2016), que analisa um sistema de arquivos ou uma imagem de disco com base no conteúdo de um arquivo de configuração, que possui uma lista de assinaturas de tipos de arquivo. Uma assinatura contém o valor do cabeçalho conhecido, o tamanho máximo do arquivo, se o valor do cabeçalho é sensível a maiúsculas, a extensão típica do tipo de arquivo e, por fim, o valor de final de arquivo (CARRIER, 2005). Como exemplo, a linha a seguir é uma entrada de assinatura para arquivos do tipo JPEG:

```
jpg y 200000 xFFxD8 xFFxD9
```

A entrada de assinatura acima mostra que a extensão típica do arquivo é jpg, o cabeçalho e o rodapé são sensíveis ao contexto, o cabeçalho possui a assinatura 0xFFD8 e o rodapé 0xFFD9. Ainda, que o tamanho máximo do arquivo é de 200.000 *bytes* e, se o rodapé não é encontrado depois de ler esta quantidade de dados, deve-se concluir o processamento de *carving* do arquivo.

A Figura 2.7, adaptada de Carrier (2005), exemplifica o processo de *data carving* de um conjunto de dados relacionados à um arquivo do tipo JPEG, no qual o cabeçalho JPEG é encontrado nos primeiros dois *bytes* do setor 902 e o valor de rodapé é encontrada no meio de setor 905. Os conteúdos dos setores 902, 903, 904 e 905 seriam extraídos como uma imagem do tipo JPEG.

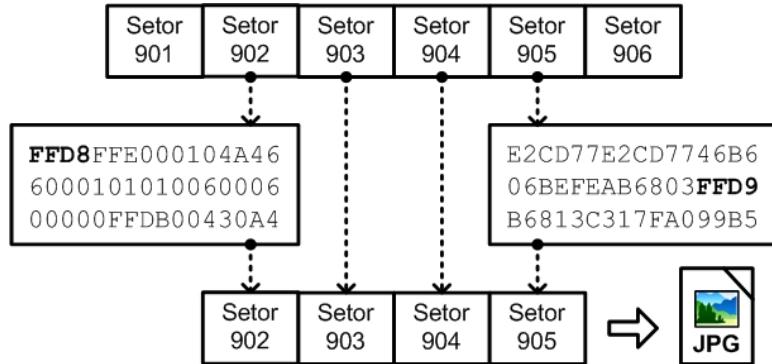


Figura 2.7: Blocos de dados são analisados para extrair um arquivo JPG (adaptada de Carrier (2005)).

Para Garfinkel (2007), o *data carving* apresenta alguns desafios. Em primeiro lugar, os arquivos a serem recuperados devem ser reconhecidos na imagem do disco. Em seguida, algum processo deve estabelecer se os arquivos estão intactos ou não. E finalmente, os arquivos devem ser copiados para fora da imagem de disco e apresentados ao examinador para que seja validado.

Cohen (2007) apresenta os principais componentes dos algoritmos de *data carving*, os quais são mostrados na Figura 2.8. Ainda Cohen (2007) comenta sobre os principais componentes:

- A imagem é processada pela primeira vez em um sistema de pré-processamento que extrai informações sobre o arquivo que está sendo analisado.
- Esta informação é então utilizada por um gerador de função de mapeamento. O gerador pode usar esta informação para restringir as funções de mapeamento produzidas.
- Para cada função de mapeamento em potencial, o gerador produz um arquivo para o discriminador. O discriminador tenta detectar erros no arquivo e, mais importante, estimar o ponto em que o arquivo foi corrompido. Esta informação é enviada de volta para o gerador de função que a utiliza para orientar a seleção de uma futura função de mapeamento.

- O gerador retorna uma ou mais funções de mapeamento consideradas boas pelo discriminador.

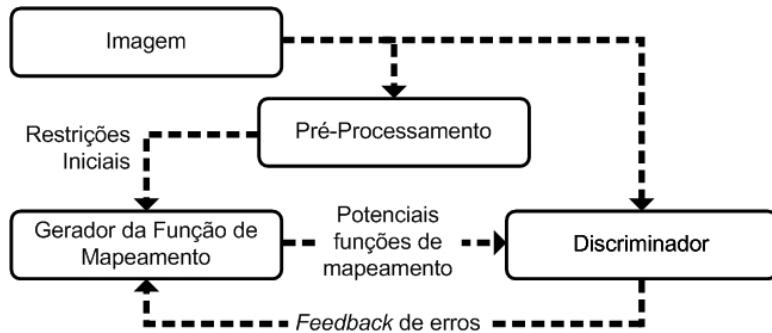


Figura 2.8: Visão geral dos algoritmos de *carving* (adaptada de Cohen (2007)).

As ferramentas de *data carving*, portanto, analisam principalmente o início e o final do arquivo. Quando identificam uma assinatura conhecida, extraem os dados entre as assinaturas identificadas (ou até um tamanho máximo de extração). Essa abordagem funciona relativamente bem, pois os sistemas de arquivos normalmente tentam manter os arquivos alocados de forma contígua e minimizam a fragmentação do arquivo (por exemplo o sistema de arquivos FAT). No entanto, em alguns sistemas de arquivos, blocos indiretos são inseridos no fluxo de dados dos arquivos, expandindo o tamanho dos arquivos e, quase sempre, fragmentando-os (tal como ocorre no sistema de arquivos Ext2 - *second extended filesystem*) (COHEN, 2007).

2.4 Fragmentação de dados

Um sistema operacional pode usar diferentes estratégias para a alocação de unidades de dado. Tipicamente, um sistema operacional aloca unidades de dado consecutivas, mas isso nem sempre é possível. Quando um arquivo não possui unidades de dado consecutivas, é chamado de fragmentação (CARRIER, 2005).

Garfinkel (2007) e Xu et al. (2014) comentam que, embora os sistemas operacionais modernos busquem evitar que os arquivos sejam fragmentados, a presença de arquivos fragmentados ainda é comum. Por exemplo, pode não haver uma região contígua suficiente para armazenar um novo arquivo; ou pode não haver um região desalocada para armazenar novos dados de um arquivo pré-existente.

Um modelo de fragmentação é um conjunto de suposições derivadas da observação de como a fragmentação ocorre na prática. Isso pode depender das características do sistema de arquivos ou de outras generalizações de suposições (COHEN, 2007).

Nesse sentido, Cohen (2007) apresenta um modelo de representação da fragmentação de um arquivo. Na discussão a seguir, o termo arquivo refere-se ao arquivo a ser recuperado (por exemplo PDF, MP3, etc.), enquanto o termo imagem refere-se a imagem de disco no qual o arquivo deve ser recuperado. A imagem contém *bytes* contíguos que representam linearmente os dados obtidos a partir do disco rígido (adquirida, por exemplo, com o programa dd) (COHEN, 2007).

Dessa forma, o processo de *data carving* pode ser definido como a extração ou a cópia dos *bytes* que pertencem a um arquivo a partir de uma imagem. Isto quer dizer que existe um mapeamento entre os *bytes* contidos no arquivo para os *bytes* dentro da própria imagem. Esse mapeamento pode ser descrito como uma função de mapeamento (COHEN, 2007).

Um exemplo de uma função de mapeamento típica é mostrado na Figura 2.9. A figura apresenta dois pontos de descontinuidade da reta, que representa a fragmentação do arquivo. A recomposição do arquivo necessita identificar que as unidades de dado da imagem após o ponto de descontinuidade não fazem parte da sequência do arquivo.

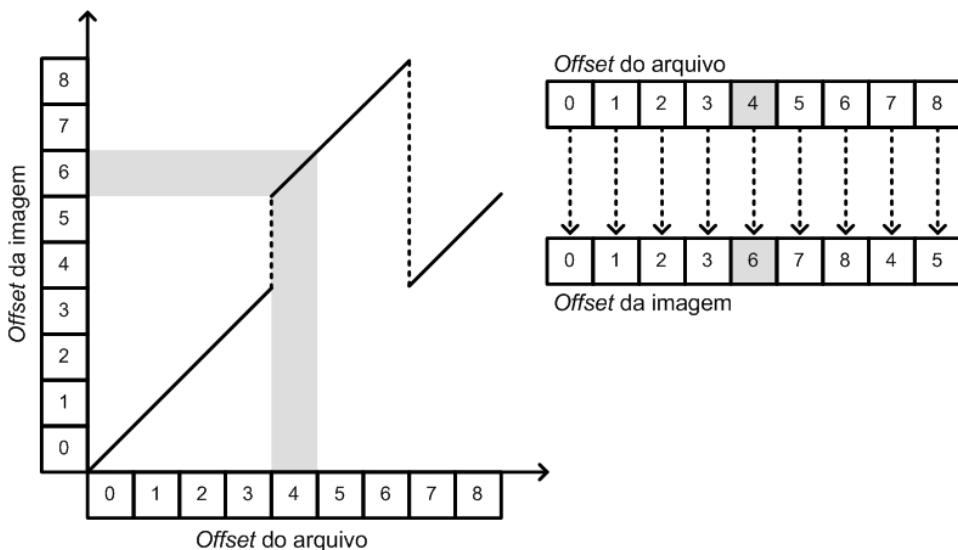


Figura 2.9: Exemplo de uma função de mapeamento (adaptada de Cohen (2007)).

Ainda sobre a Figura 2.9, podemos observar que: a inclinação da função é sempre 1 (pois existe um mapeamento um para um entre a imagem e o arquivo); há pontos de descontinuidades na função em vários lugares; a função é inversível, isto é, há no máximo um *offset* do arquivo para cada *offset* da imagem (se existir tal valor). Na prática, isto significa que uma vez que um setor da imagem tenha sido alocado, este não pode ser usado simultaneamente por outro arquivo (COHEN, 2007).

A fragmentação ocorre em discos rígidos como resultado da estratégia de alocação do sistema de arquivos. Essas estratégias são normalmente concebidas para otimizar algumas características do sistema de arquivos (por exemplo, velocidade de acesso mais rápida, melhor eficiência de armazenamento, etc.) e não são deliberadamente concebidas para dificultar a recuperação. Muitas vezes é possível fazer suposições sobre o tipo de fragmentação presente com base nas informações sobre o sistema de arquivos que estava anteriormente no disco e suas estratégias de alocação típicas (COHEN, 2007).

Os sistemas operacionais modernos tentam gravar arquivos sem fragmentação, porque dessa forma os arquivos são mais rápidos para ler e escrever. Entretanto, como descreve (GARFINKEL, 2007), existem três situações em que um sistema operacional deve gravar um arquivo em dois ou mais fragmentos:

- Quando não há região contígua de setores na mídia, grande o suficiente, para armazenar o arquivo sem fragmentação. Isso é provável se a unidade de armazenamento estiver em uso por um longo tempo, for preenchida perto da capacidade e tiver muitos arquivos adicionados e excluídos em uma ordem aleatória ao longo do tempo.
- Se há dados adicionados a um arquivo existente, talvez não haja setores não alocados suficientes no final do arquivo para acomodar os novos dados. Neste caso, alguns sistemas de arquivos podem realocar o arquivo original, entretanto a maioria simplesmente escreverá os novos dados em outro local do disco.
- O próprio sistema de arquivos pode não suportar a gravação de arquivos de um determinado tamanho de forma contígua. Por exemplo, o sistema de arquivos *Unix* irá fragmentar os arquivos longos ou com *bytes* no final do arquivo que não se encaixam em um número par de setores (CARRIER, 2005).

Considere, por exemplo, a função de mapeamento mostrada na Figura 2.10. Esta função tem dois pontos identificados positivamente (P_1 e P_2). Uma vez que os pontos não se encontram na mesma linha de inclinação, deve haver pelo menos uma descontinuidade entre eles. A Figura 2.10 mostra três funções de mapeamento possíveis numeradas 1, 2 e 3. As possibilidades 1 e 3 têm uma única descontinuidade em um setor entre P_1 e P_2 . No entanto, a possibilidade 2 tem duas descontinuidades. Segundo Cohen (2007), as possibilidades 1 e 3 representam uma fragmentação de primeira ordem, enquanto a possibilidade 2 representa uma fragmentação de segunda ordem.

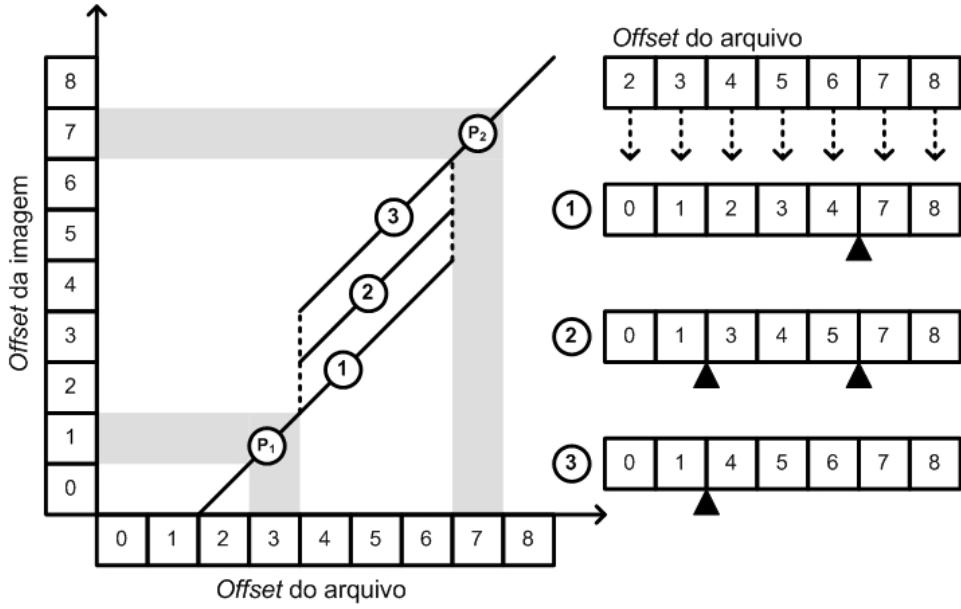


Figura 2.10: Possibilidades de mapeamento na ocorrência de fragmentação (adaptada de Cohen (2007)).

Embora em teoria possa haver várias descontinuidades entre os pontos identificados, na prática a fragmentação é tipicamente mantida em um nível baixo nos sistemas de arquivos modernos. Isto implica que quanto menor a diferença de deslocamento entre pontos identificados no arquivo, mais provável que a fragmentação será de primeira ordem (COHEN, 2007).

Um dos principais problemas com a gravação de arquivos é a fragmentação de arquivos. Se um arquivo nunca foi fragmentado, descobrir o início do arquivo e o final dele seria suficiente para restaurar completamente o arquivo (VEENMAN, 2007).

Eventualmente os metadados sobrevivem para facilitar a reconstrução do arquivo, mas frequentemente o que o investigador possui é uma coleção aleatória de fragmentos de arquivo, que precisam ser reunidos para formar os arquivos originais (ou parcialmente completos) (AXELSSON, 2010).

Descobrir a primeira e a última unidade de dados de um arquivo é bastante fácil para muitos tipos de arquivo, pois as assinaturas são usadas para indicar o tipo no cabeçalho, no rodapé e, até mesmo, nas unidades intermediárias. Em caso de fragmentação de arquivo, as unidades de dado ficam desconectadas e podem estar fora de ordem. Além disso, para vários tipos de arquivo, não há informações de assinatura de cabeçalho disponíveis (VEENMAN, 2007).

3 CLASSIFICAÇÃO DE FRAGMENTOS

Este capítulo é dividido em 2 seções. A Seção 3.1 apresenta os principais conceitos de aprendizagem automática e de classificação de fragmentos. Na Seção 3.2 são descritos os atributos extraíveis dos fragmentos de arquivo para treinar uma máquina de classificação.

3.1 Aprendizagem automática e classificação de fragmento

Comumente, os analistas forenses devem reconstruir um arquivo a partir dos seus fragmentos restantes, após a tentativa de exclusão do arquivo pelo usuário (CALHOUN; COLES, 2008). Normalmente, partes do arquivo terão sido substituídas por outros arquivos mais recentes. No processo de reconstrução do arquivo, Calhoun & Coles (2008) enfatiza a importância de o analista ter conhecimento do tipo de arquivo de um determinado fragmento.

Existe uma maneira simples e confiável de determinar o tipo de arquivo que é verificando o seu cabeçalho. Isto é facilitado nos formatos de código aberto e, mesmo nos formatos proprietários, é possível encontrar informações explícitas de identificação de tipo nos *bytes* iniciais do arquivo (CALHOUN; COLES, 2008).

Uma ferramenta comumente utilizada para a identificação do tipo do arquivo é o programa `file` do *Unix*, que depende da biblioteca de assinatura `libmagic`. A ferramenta usa as assinaturas para reconhecer os tipos de arquivo. Quando um arquivo está completo, esse método é bastante eficiente na identificação de arquivos através da comparação de regiões específicas de um arquivo para identificar correspondências de assinaturas (ROUSSEV; GARFINKEL, 2009).

Entretanto, a classificação de fragmentos de arquivo torna-se complicada quando existem muitos tipos diferentes de arquivo, tais como tipos primitivos simples, como blocos de texto ASCII ou arquivos JPEG; arquivos de contêiner complexos, como arquivo PDF, arquivos compactados, como TAR e ZIP, os quais podem conter outros arquivos (ROUSSEV; GARFINKEL, 2009).

Ademais, em cenários em que partes dos metadados foram perdidos ou corrompidos, os *softwares* forenses atuais podem não ser capazes de identificar corretamente o tipo de um fragmento sem o cabeçalho intacto (CALHOUN; COLES, 2008).

Como descreve Xie, Abdullah & Sulaiman (2013), a classificação de fragmentos visa classificar diferentes categorias de fragmentos de arquivos através de métodos baseados em extensões, métodos baseados em assinaturas e métodos baseados em conteúdo. Dentre estes, o método baseado em conteúdo é o mais desafiador, já que é frequentemente aplicado na classificação de arquivos cujos metadados foram perdidos ou corrompidos.

Os métodos baseados em aprendizagem automática foram explorados para resolver o problema de classificação de fragmentos (XIE; ABDULLAH; SULAIMAN, 2013). A aprendizagem automática é um ramo da Inteligência Artificial na qual, através do uso da computação, são projetados sistemas que podem ser treinados a partir de um conjunto de dados. Tais sistemas podem aprender e melhorar com a experiência e, com o tempo, refinar um modelo que pode ser usado para prever resultados com base na aprendizagem anterior (BELL, 2014).

Nesse sentido, os principais trabalhos sobre a classificação de fragmentos procuram resolver o problema de classificação usando uma combinação de técnicas de aprendizagem automática e de análise estatística (ROUSSEV; GARFINKEL, 2009). Os pesquisadores normalmente selecionam um conjunto de arquivos de diferentes tipos, o qual é dividido em dois grupos, sendo um conjunto de treinamento e um conjunto de teste. Os arquivos no conjunto de treinamento são processados com algum tipo de técnica estatística e os resultados são alimentados em um algoritmo de aprendizagem automática. Os resultados são usados para criar um modelo de classificador. O conjunto de teste é então alimentado no modelo de classificador e sua capacidade de classificação é medida e, finalmente, relatada.

As técnicas de aprendizagem automática tendem a ser mais genéricas, incorporando um número de atributos extraídos dos arquivos (ou fragmentos dos arquivos) e deixando o algoritmo determinar os pesos através do treinamento (ROUSSEV; GARFINKEL, 2009).

A combinação de informações extraídas do conteúdo dos fragmentos de arquivo e da aprendizagem automática produz uma solução de classificação de fragmentos de tal

forma que a precisão da técnica não se baseia nas informações potenciais de metadados, mas sim nos valores dos dados em si. Os atributos extraídos são então aplicados em um classificador para a classificação do tipo de fragmento de arquivo. De acordo com Xie, Abdullah & Sulaiman (2013), depois que os atributos são selecionados, a classificação de fragmentos é um problema comum de aprendizagem da máquina.

3.2 Atributos dos fragmentos para a aprendizagem automática

Para treinar uma máquina de classificação sobre os dados de fragmento de arquivo, é necessário representar cada fragmento de arquivo como um conjunto de atributos. Cada atributo deve ser extraível de qualquer tipo de arquivo ou fragmento (VEENMAN, 2007), ou seja, não deve ser específico para um tipo ou não deve depender de uma estrutura específica de um tipo de arquivo.

Portanto, segundo Veenman (2007), para descrever e diferenciar todos os tipos de arquivo, os atributos não podem ser montados para descrever invariantes de tipos de arquivo específicos, mas em vez disso, invariantes gerais de conteúdo de arquivo, tais como:

- fragmentos de dado que não possuem uma ordem específica (por exemplo: funções em arquivos fonte de linguagem de programação);
- fragmentos de dado que podem ser reordenados sem se tornarem completamente diferentes (por exemplo, imagens não comprimidas);
- dados que podem ser deslocados mantendo-se sintaticamente corretos e semanticamente semelhantes (por exemplo, introdução de *feeds* de linha em arquivos de texto).

Por outro lado, várias características do arquivo são específicas para determinados tipos de arquivos e, portanto, ajudam a diferenciá-los de outros arquivos. Veenman (2007) chama essas características de variantes do tipo de arquivo, tais como:

- dados que possuem uma ordem específica;
- certos símbolos que aparecem frequentemente (por exemplo, os símbolos “<” e “>” em arquivos do tipo HTML).
- arquivos que geralmente contêm informações redundantes e o grau de redundância é muito diferente entre arquivos (por exemplo, arquivos do tipo ZIP vs. HTML).

A extração de atributos também deve considerar técnicas que operam no nível de *bits* individuais e que não assumem esquemas de codificação de *byte* (CONTI et al., 2010). Ainda, como indica Xu et al. (2014), a combinação com outras técnicas, como os métodos baseados em palavras-chave, também permite melhorias na precisão da classificação.

A Figura 3.1 mostra os vários níveis de atributos que podem ser extraídos de um fragmento. Atributos nos níveis de $A2$ a $A4$ podem ser extraídos de fragmentos individuais. Atributos no nível $A1$, entretanto, são baseados no contexto do fragmento e, portanto, a extração necessitaria saber previamente sobre os outros fragmentos relacionados à ele.

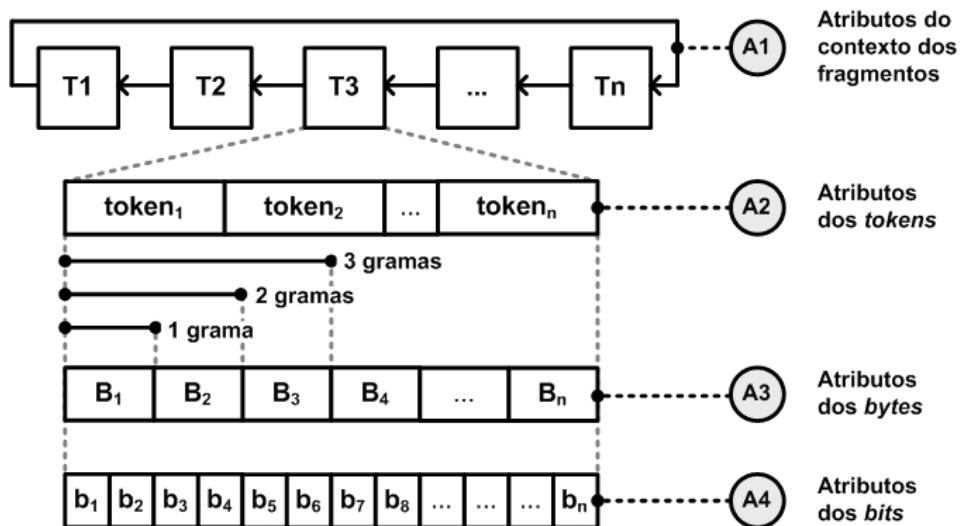


Figura 3.1: Tipos de atributo extraídos dos fragmentos.

Fitzgerald et al. (2012) descreve o tipo de abordagem chamado de *bag-of-words model*, no qual os arquivos são representados como conjunto de palavras não ordenadas. Uma palavra é texto separado por espaços em branco (ou outros tipos de separadores). A contagem de um *token* de palavra é chamada unigrama, ademais, agrupamentos de *tokens* de qualquer número fixo de palavras também podem ser considerados. Assim, dois *tokens* de palavra consecutivos são chamados de bigrama, cujas contagens capturam mais informações sobre a estrutura do fragmentos que as contagens unigramas.

Fitzgerald et al. (2012) descreve, ainda, o tipo de abordagem *bag-of-bytes*, na qual *bytes* individuais são tratados como *tokens*. Dessa forma, as contagens de unigrama, bigram e trigrama de *bytes* consideram, respectivamente, *bytes* individuais e grupos de dois e três *bytes* consecutivos no processamento de extração de atributos dos fragmentos.

Nas subseções seguintes são descritos vários atributos que podem ser extraídos de um fragmento ou um arquivo (ou seja, de uma sequência de *bytes*). Tais atributos foram utilizados individualmente ou em conjunto em experimentos de aprendizagem automática por outros autores.

3.2.1 Atributo de frequência de caracteres ASCII

Em uma dada unidade de dado, é calculada a proporção de caracteres ASCII dentro das seguintes faixas de valores (CALHOUN; COLES, 2008):

- ASCII de baixo valor: frequência dos *bytes* de valor v , onde $0 \leq v < 32$.
- ASCII de médio valor: frequência dos *bytes* de valor v , onde $32 \leq v < 128$.
- ASCII de alto valor: frequência dos *bytes* de valor v , onde $129 \leq v < 256$.

3.2.2 Atributo de percentual de codificação em *Base64* e em *Base85*

As codificações *Base64* e *Base85* são frequentemente utilizadas para transmitir dados binários por meios de transmissão que lidam apenas com texto. Os dados binários são convertidos em caracteres ASCII, sendo que a codificação *Base64* utiliza quatro caracteres para representar três *bytes* de dado, enquanto que a codificação *Base85* utiliza cinco caracteres para representar quatro *bytes*.

A detecção é direta, pois cada codificação usa uma gama bem definida de códigos ASCII (ROUSSEV; QUATES, 2013). São verificados o percentual dos caracteres $0..9$, $a..z$, $A..Z$, $,$, $=$, $/$ (no caso de *Base64*) e dos caracteres $0..9$, $a..z$, $A..Z$, $,$, $-$, $:$, $=$, $,$, $*$, $^$, $!$, $/$, $*$, $?$, $&$, $<$, $>$, $($, $)$, $[$, $]$, ${$, $}$, $@$, $%$, $$$, $#$ (no caso de *Base85*).

3.2.3 Atributos estatísticos de média, variação e correlação dos *bytes*

Considera os valores representados pelos *bytes* para o cálculo dos atributos estatísticos do fragmento. O atributo de média dos *bytes* foi utilizado nos trabalhos de Calhoun & Coles (2008), Conti et al. (2010) e Fitzgerald et al. (2012). Ele é obtido pela soma de todos os *bytes* do fragmento dividido pelo comprimento do arquivo.

Outro atributo é extraído da correlação dos valores dos *bytes* do fragmento, que é extraído através da aplicação da Equação 3.1:

$$\rho = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} = \frac{cov(X, Y)}{\sqrt{var(X).var(Y)}}, \quad (3.1)$$

sobre dois conjuntos de vetores de valores dos *bytes*, os quais são representados por x_1, x_2, \dots, x_n e y_1, y_2, \dots, y_n . Os vetores de *bytes* podem ser gerados de duas formas: (1) dividindo-se o fragmento na metade; ou (2) separando os *bytes* das posições pares e ímpares. Também é utilizado nos cálculos a média aritmética dos valores dos *bytes* de cada vetor na Equação 3.2:

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^n x_i \quad \bar{y} = \frac{1}{n} \cdot \sum_{i=1}^n y_i \quad (3.2)$$

Assim, a análise correlacional indica a relação entre duas variáveis lineares, cujo resultado é um valor entre +1 e -1. O sinal do resultado indica a direção, ou seja, se a correlação é positiva ou negativa, e o tamanho indica a força da correlação, ou seja, se há uma fraca ou forte correlação entre os valores do vetor. Essa atributo mede a extensão em que cada *byte* no fragmento e depende do *byte* anterior (WALKER, 2008).

A correlação de valores de *bytes* não foi utilizada por Conti et al. (2010), entretanto, o mesmo autor indicou a possibilidade do uso de tal atributo na classificação de fragmentos.

3.2.4 Atributo de teste de qui-quadrado dos valores dos *bytes*

O qui-quadrado, simbolizado por X^2 , é um teste de hipóteses que se destina a encontrar o valor da dispersão para duas variáveis nominais, avaliando a associação existente entre variáveis qualitativas. É um teste não paramétrico, ou seja, não depende dos parâmetros populacionais, como média e variância. O princípio básico deste método é comparar proporções, isto é, as possíveis divergências entre as frequências observadas e esperadas para um certo evento. O cálculo de X^2 é feito da seguinte conforme a Equação 3.3:

$$X^2 = \sum \frac{(f_{observada} - f_{esperada})^2}{f_{esperada}}, \quad (3.3)$$

onde $f_{observada}$ = frequência observada para cada classe e $f_{esperada}$ = frequência esperada para aquela classe.

Pode-se dizer que dois grupos se comportam de forma semelhante se as diferenças

entre as frequências observadas e as esperadas em cada categoria forem muito pequenas, próximas a zero (BARBETTA, 2008). Portanto, o teste é utilizado para:

- verificar se a frequência com que um determinado acontecimento observado em uma amostra se desvia significativamente ou não da frequência com que ele é esperado;
- comparar a distribuição de diversos acontecimentos em diferentes amostras, a fim de avaliar se as proporções observadas destes eventos mostram ou não diferenças significativas ou se as amostras diferem significativamente quanto às proporções desses acontecimentos.

O teste de qui-quadrado foi utilizado por Shannon (2004), Conti et al. (2010) e Walker (2008), sendo que este último autor desenvolveu uma ferramenta específica para o cálculo, chamada de `ent`.

3.2.5 Atributo de cálculo de *Monte Carlo Pi*

No cálculo de *Monte Carlo Pi* sobre o fragmento, cada sequência sucessiva de seis *bytes* é usada como coordenadas *X* e *Y* dentro de um quadrado. Se a distância do ponto gerado aleatoriamente for menor que o raio de um círculo inscrito dentro do quadrado, a sequência de seis *bytes* é considerada um *hit*. A porcentagem de acertos (*hits*) pode ser usada para calcular o valor de *Pi* (conforme mostrado na Figura 3.2).

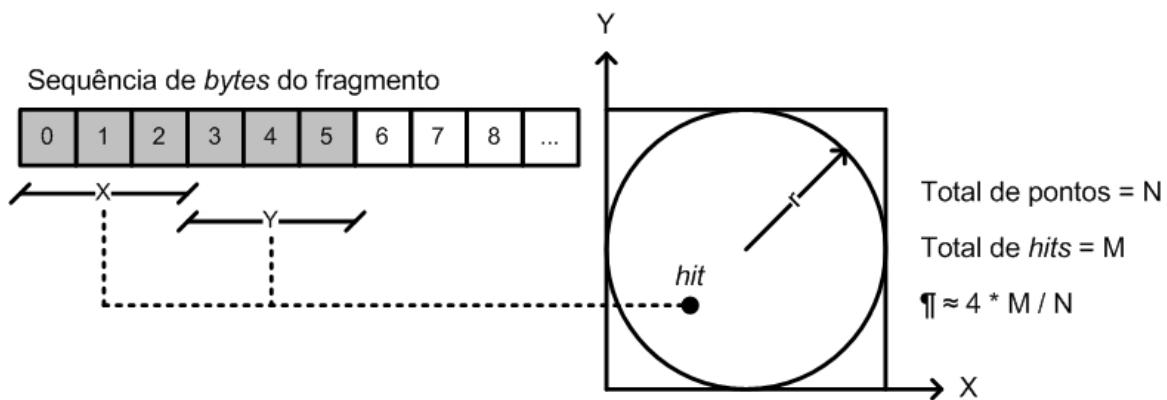


Figura 3.2: Exemplo de cálculo de *Monte Carlo Pi*.

Para fluxos muito grandes, esta aproximação converge muito lentamente e o valor se aproximará do valor correto de *Pi* se a sequência estiver próxima ao aleatório.

O cálculo de *Monte Carlo Pi* foi utilizado por Shannon (2004), Conti et al. (2010) e Walker (2008), sendo que este último autor desenvolveu uma ferramenta específica para o cálculo, chamada de `ent`.

3.2.6 Atributo de percentuais de padrões 0xF8, 0xFF e 0x00

O atributo é resultado do cálculo do percentual de ocorrências dos *bytes* 0xF8, 0xFF e 0x00 no fragmento, sendo possível, por exemplo, a contagem de ocorrências de 1-grama (0xF8, 0xFF e 0x00) e 2-gramas (0xF8F8, 0xFFFF e 0x0000).

Os padrões de *bytes* 0xFF e 0x00 foram identificados por Conti et al. (2010) como padrões que influenciaram de forma negativa o classificador de seu experimento.

Em arquivos no formato JPEG, por exemplo, o *byte* 0xFF é usado como um caractere de escape para todas as *tags* de metadados. Para evitar a ambiguidade e para simplificar o processamento, o codificador armazena também o 0x00 extra após cada *byte* 0xFF no corpo do arquivo, o que produz um padrão muito regular, único e facilmente explorável (ROUSSEV; GARFINKEL, 2009; XIE; ABDULLAH; SULAIMAN, 2013).

3.2.7 Atributo de peso de *Hamming*

O peso de *Hamming* é a contagem do número de símbolos diferentes de zero em um determinado alfabeto. No caso específico dos fragmentos de arquivo, considera-se que o alfabeto contém apenas dados binários, isto é, um alfabeto de zeros e uns. Assim, o peso de *Hamming* é o número total de *bits* 1 dividido pelo número total de *bits* do fragmento (CONTI et al., 2010). O cálculo do peso de *Hamming* foi utilizado por Conti et al. (2010) e por Fitzgerald et al. (2012) como um dos atributos para o modelo de classificação.

3.2.8 Atributo sobre o histograma dos *bytes*

Histograma da frequência de *byte*, isto é, a frequência que cada um dos valores de *byte* ocorre no fragmento de dado, ignorando-se a ordem dos *bytes* no fragmento (VEENMAN, 2007; LI et al., 2011; XIE; ABDULLAH; SULAIMAN, 2013).

Para gerar o histograma dos *bytes*, no caso de uma contagem de 1-grama, conta-se cada *byte* para extrair a distribuição de frequência de seu valor, que corresponde a um número de oito *bits* (LI et al., 2011; XIE; ABDULLAH; SULAIMAN, 2013). É possível, ainda, a contagem de 2-gramas, a qual foi utilizada por Fitzgerald et al. (2012), e de 3-gramas de *byte*.

Sobre o histograma de frequência dos *bytes*, Li et al. (2011) sugere a extração de atributos estatísticos, tais como média, desvio padrão, moda (a soma das maiores frequências dos *bytes*) e correlação sequencial (correlação das frequências dos valores dos *bytes* em m e $m + 1$).

3.2.9 Atributo de vetor de histograma dos *bytes*

No vetor de histograma dos *bytes*, cada frequência de *byte* é considerada um atributo. Em contagens de 1-grama, são gerados 256 atributos (LI et al., 2011); já em contagens de 2-gramas, são produzidos 256^2 atributos, ou seja, um atributo para cada combinação de um par de *bytes* consecutivos (FITZGERALD et al., 2012).

3.2.10 Atributo de linguagem utilizada no texto

Em fragmentos que possuem conteúdo textual, é possível detectar a língua na qual o texto foi escrito (por exemplo, em inglês, japonês ou chinês). Nakatani (2010) desenvolveu uma API específica para a detecção de linguagem em arquivos ou fragmentos, na qual é possível identificar até 50 tipos de língua, sendo que para 49 línguas, o autor indica uma taxa de sucesso de 99,8%.

3.2.11 Atributos de processamento de linguagem natural

Fitzgerald et al. (2012) sugere dois atributos de processamento de linguagem natural (*natural language processing*). O primeiro atributo é o cálculo da média da distância entre os valores de dois *bytes* consecutivos em um fragmento, o qual é realizado, para um fragmento de 512 *bytes*, conforme o Somatório 3.4:

$$\sum_{i=0}^{n=510} \frac{|f[i] - f[i + 1]|}{511}, \quad (3.4)$$

onde $f[i]$ é o valor do *byte* na posição i do fragmento. Já o segundo atributo é a sequência contígua mais longa de *bytes* repetidos de cada fragmento.

3.2.12 Atributo de linguagem de programação

Identifica se o conteúdo textual possui palavras pertencentes ao alfabeto de uma linguagem de programação, tais como as linguagens PHP, C++, C#, Java, Ruby, Python, HTML (*HyperText Markup Language*), SQL (*Structured Query Language*),

Script Bash, Script DOS (Disk Operating System), JavaScript, CSS (Cascading Style Sheets) e XML (eXtensible Markup Language).

A alta frequência de *tokens* de um determinado alfabeto é indício de que o fragmento é um conteúdo textual escrito em uma linguagem específica. Conti et al. (2010) exemplifica que textos codificados em HTML podem ser classificados com identificadores específicos para esse tipos.

3.2.13 Atributo de entropia dos *bytes*

Outra estatística útil é a entropia do fragmento de arquivo. A entropia é uma medida da densidade de informação ou estado de compressão de uma dada unidade de dado. Quanto mais uma unidade pode ser comprimida, menor o valor de sua entropia e, quanto menos uma unidade pode ser comprimida, maior o valor de sua entropia (SHANNON, 2004).

Por exemplo, longas sequências de zeros têm baixa entropia, uma vez que contêm pouca informação. As sequências aleatórias, por outro lado, têm alta entropia. Arquivos de texto têm entropia baixa uma vez que grande parte das informações no arquivo é redundante. Já os arquivos compactados ou criptografados têm alta entropia (CALHOUN; COLES, 2008).

No caso de análise de entropia dos *bytes* de um fragmento, o seu cálculo pode ser realizado através da Equação 3.5:

$$H(X) = - \sum_{i=0}^n p(X_i) \log_b p(X_i), \quad (3.5)$$

onde X é uma variável aleatória com 256 resultados potenciais ($x_i : i = 0, \dots, 255$) e base $b = 10$. A função de massa de probabilidade $p(x_i)$ é a probabilidade do valor do byte i dentro de um dado fragmento (CONTI et al., 2010).

Ademais, pode-se considerar a contagem de 2-gramas de *byte* (256^2 possibilidades de saída) e 3-gramas de *byte* (256^3 possibilidades de saída), conforme indica o trabalho de Conti et al. (2010). Em seus experimentos, Conti et al. (2010) e Fitzgerald et al. (2012) utilizaram apenas a entropia sobre 1 e 2-gramas de *byte*.

3.2.14 Atributo de percentual de uso de *tag*

Veenman (2007) chama essas características de variantes do tipo de arquivo, tais como certos símbolos que aparecem frequentemente (por exemplo, os símbolos “<” e “>” em arquivos do tipo HTML). Dessa forma, são atributos relacionados ao percentual de *bytes* que representam o uso de *tags*, tais como os símbolos: “{ }”, “< >” e “()”.

3.2.15 Atributo de taxa de compressão do fragmento

Atributo extraído a partir da taxa de compressão do fragmento, através da Equação 3.6:

$$T(F) = \frac{\text{tamanho}(F_{\text{original}}) - \text{tamanho}(F_{\text{comprimido}})}{\text{tamanho}(F_{\text{original}})}, \quad (3.6)$$

onde F_{original} é o tamanho inicial do fragmento e $F_{\text{comprimido}}$ é o tamanho do fragmento após a aplicação de algum algoritmo de compressão de arquivo, resultando em fragmentos comprimidos nos padrões ZIP ou GZIP. O cálculo da taxa de compressão do fragmento foi utilizado pelos autores Veenman (2007) e Fitzgerald et al. (2012).

3.2.16 Atributo de frequência circular dos *bytes*

Xie, Abdullah & Sulaiman (2013) propõe a análise da frequência dos *bytes* com representação circular, onde um conjunto de fragmentos de arquivo é dividido em vários blocos usando um esquema de particionamento fixo e, em seguida, para cada bloco é extraído o atributo baseado na análise de frequência de *byte*.

Dessa forma, o esquema de particionamento circular em blocos considera a relação de um fragmento com o fragmento seguinte do arquivo, sendo que para o último fragmento do arquivo a relação é analisada com o primeiro fragmento.

3.2.17 Atributo de vetor de escala de cinza

Xu et al. (2014) analisou cada fragmento como um fragmento de imagem em preto e branco. Assim, cada *byte* do fragmento foi considerado como um *pixel* em uma imagem em escala de cinza e, portanto, foi considerada uma escala de 256 níveis. O resultado do processamento de um fragmento será um vetor de valores em escala de cinza.

3.2.18 Atributo de complexidade *Kolmogorov*

A entropia de *Shannon*, a grosso modo, dá uma estimativa da complexidade de *Kolmogorov*. Tanto a entropia como a complexidade de *Kolmogorov* refletem a estrutura em uma *string* arbitrária. A complexidade de *Kolmogorov* é o comprimento mínimo da descrição (em *bits*) de uma *string*: é um limite inferior na compressibilidade. A entropia de *Shannon* pode ser interpretada como o número de *bits* necessários para uma codificação ideal de uma *string* a partir de uma fonte estocástica (VEENMAN, 2007).

3.2.19 Atributo de subsequência comum mais longa de *bytes*

Atributo utilizado por Calhoun & Coles (2008) para a classificação de tipo de arquivo. Uma subsequência de um fluxo de *bytes* é qualquer sequência de *bytes* obtida por supressões a partir do original. A intuição para considerar as subsequências é que várias pequenas subcadeias de metadados podem ser separadas das longas sequências de dados reais. Dessa forma, considera-se as subsequências sem os dados inúteis (para a predição de tipo), concatenando-se as partes úteis em uma única sequência. Deve-se notar que as subsequências comum mais longas requerem que o fragmento desconhecido seja comparado com cada um dos arquivos de amostra.

3.2.20 Atributos de distância de compressão normalizada

Calcula a distância entre todos os pares de blocos. A distância de compressão normalizada baseia-se na idéia de que, usando um algoritmo de compressão sobre vetores de dados, individualmente e concatenados, pode-se extrair um atributos de medida da distância entre eles (AXELSSON, 2010).

4 MODELO DE CLASSIFICAÇÃO

Neste capítulo é apresentado uma proposta de modelo de classificação de fragmentos de arquivo. Para tanto, este capítulo é dividido em 4 seções. A Seção 4.1 apresenta os principais conceitos de árvores de decisão. Já na Seção 4.2 são descritos os passos para a construção das árvores de decisão. Na Seção 4.3 é apresentado o modelo de classificação de fragmento baseado em árvores de decisão. E na Seção 4.4 são apresentados os atributos de fragmento utilizados no modelo.

4.1 Noções de árvores de decisão

O objetivo de qualquer árvore de decisão é criar um modelo viável que preveja o valor de uma variável de destino com base no conjunto de variáveis de entrada (BELL, 2014). As árvores de decisão podem ser aplicadas a problemas de regressão e de classificação (JAMES et al., 2013; GOLLAPUDI, 2016).

Uma árvore de classificação é muito semelhante a uma árvore de regressão, exceto que ela é usada para prever uma resposta qualitativa e não quantitativa (JAMES et al., 2013). Para usar árvores de classificação, a resposta da variável de destino precisa ser um valor categórico, como *yes/no* ou *true/false*. Por outro lado, as árvores de regressão são usadas para atender aos requisitos de previsão e são sempre usadas quando a variável alvo ou resposta é um valor numérico ou discreto (GOLLAPUDI, 2016).

Os modelos baseados em árvores de decisão apresentam características positivas para a sua escolha, tais como:

- a facilidade de interpretação e análise do modelo, assim como a facilidade de apresentação do modelo para terceiros (JAMES et al., 2013);
- a possibilidade de testes de caixa branca, o que significa que o funcionamento interno pode ser observado e, assim, pode-se ver os passos que estão sendo usados quando a árvore está sendo modelada (BELL, 2014);
- a possibilidade de tratar dados de atributos de tipos numéricos e nominais (BELL, 2014);

- a disponibilidade de algoritmos que permitem resolver tanto problemas binomiais quanto problemas multinomiais (GOLLAPUDI, 2016).

Ademais, a técnica que já mostrou bom desempenho em trabalhos anteriores (AHMED et al., 2011), porém foi pouco explorada quanto aos atributos, algoritmos, tamanhos de fragmento e tipos de arquivo utilizados.

Uma das principais questões das árvores de decisão é que elas podem criar modelos excessivamente complexos, dependendo dos dados apresentados no conjunto de treinamento (BELL, 2014). Para evitar que ocorra o *over-fitting* do algoritmo de aprendizado da máquina, pode-se rever os dados de treinamento e podar os valores para as categorias, o que produzirá um modelo mais refinado e melhor ajustado (BELL, 2014).

4.2 Construção de árvores de decisão

As árvores de decisão classificam instâncias através de uma estrutura de árvore começando da raiz até uma folha. As árvores são construídas e representadas através de dois elementos: nós e arcos que conectam os nós.

Para tomar uma decisão, o fluxo começa no nó raiz, navega através das arestas até chegar a um nó folha e, em seguida, toma uma decisão. Cada nó da árvore denota um teste de um atributo, e os ramos denotam os possíveis valores que o atributo pode tomar (GOLLAPUDI, 2016).

Na Figura 4.1 podemos ver uma representação de uma árvore de classificação de fragmentos do tipo EML, na qual em cada nó intermediário é avaliado um atributo do fragmento e cada aresta há uma regra de decisão sobre os valores dos atributos.

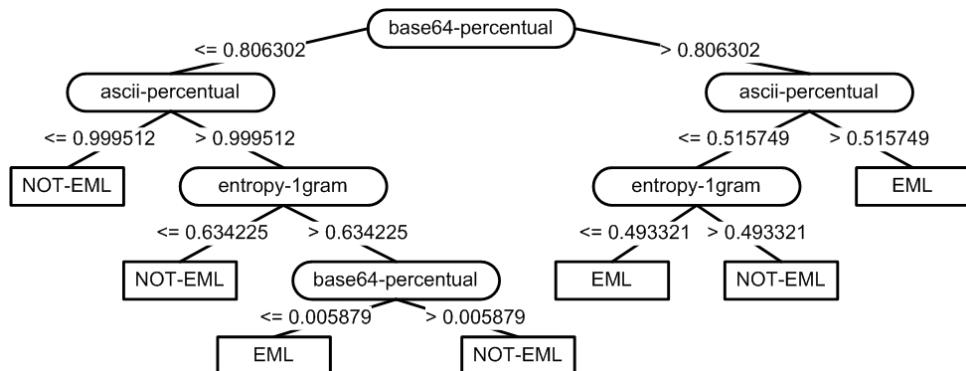


Figura 4.1: Exemplo de uma árvore de decisão.

Alternativamente, as árvores de decisão podem ser expressar através de um conjunto de regras textuais. Assim, a mesma árvore representada na Figura 4.1 pode ser escrita da seguinte forma:

```

base64-percentual <= 0.806302
|   ascii-percentual <= 0.999512:  NOT-EML
|   ascii-percentual > 0.999512
|       |   entropy-1gram <= 0.634225:  NOT-EML
|       |   entropy-1gram > 0.634225
|           |   base64-percentual <= 0.005879:  EML
|           |   base64-percentual > 0.005879:  NOT-EML
base64-percentual > 0.806302
|   entropy-1gram <= 0.515749
|       |   entropy-1gram <= 0.493321:  EML
|       |   entropy-1gram > 0.493321:  NOT-EML
|   entropy-1gram > 0.515749:  EML

```

Em resumo, a representação das árvores de decisão possui as seguintes características (GOLLAPUDI, 2016):

- cada nó não-folha (isto é, um nó de decisão) denota uma representação do valor do atributo;
- cada aresta denota o resto da representação do valor;
- cada nó de folha (ou terminal) representa o valor do atributo de destino;
- o nó de partida é chamado de nó raiz.

As árvores de decisão sempre começam com um nó raiz e terminam em uma folha. É importante observar que as árvores não convergem em nenhum ponto; elas segmentam o seu caminho à medida que os nós são processados (BELL, 2014).

Bell (2014) descreve o algoritmo básico para a construção de uma árvore de decisão:

1. Verifique o modelo para os casos de base.
2. Itere através de todos os atributos (**attr**).
3. Obtenha o ganho de informação normalizado para a divisão em **attr**.
4. Seja **best-attr** o atributo com o maior ganho de informação.
5. Crie um nó de decisão que se divide no atributo **best-attr**.

6. Trabalhe nas sublistas que são obtidas dividindo em `best-attr` e adicione esses nós como nós filhos.

Há diferentes métodos para avaliar a utilização de cada atributo nas regras de decisão da árvore. O método comumente utilizado é através da avaliação do ganho de informação e da entropia (BELL, 2014; GOLLAPUDI, 2016). O método de avaliação do ganho de informação baseado no cálculo da entropia do atributo, antes e depois da seleção do atributo, é o método padrão utilizado nos algoritmos ID3 e C4.5 (QUINLAN, 2014).

Através da análise do ganho de informação e da entropia, busca-se o nó que pode melhor prever o resultado da classificação (BELL, 2014). Na versão original do algoritmo C4.5, um nó raiz é escolhido com base no quanto de entropia total é reduzida caso esse nó seja escolhido (GOLLAPUDI, 2016).

Portanto, para Gollapudi (2016), considera-se que há maior ganho de informação através do atributo que, ao se tornar um nó, produz subconjuntos com menor entropia. Dessa forma, o ganho de entropia é calculado conforme a Equação 4.1:

$$G = E_{antes} - E_{depois}, \quad (4.1)$$

onde G é o ganho de informação obtido, E_{antes} é a entropia do sistema antes da divisão, e E_{depois} é a entropia do sistema após a divisão. A entropia no sistema antes da divisão obtida conforme a Equação 4.2:

$$E = - \sum_{i=1}^m p_i \log_2(p_i) \quad (4.2)$$

Assim, por exemplo, a entropia após usar A para dividir D em v partições para classificar D é dada conforme a Equação 4.3:

$$E_A = \sum_{i=1}^v \frac{D_i}{D} E(D_i) \quad (4.3)$$

4.3 Modelo baseado em árvores de decisão

A Figura 4.2 ilustra como o modelo de classificação pode ser aplicado na classificação de fragmentos e na recuperação de arquivos. Nela é apresentado um cenário no qual inexiste, para alguns fragmentos de arquivo, as informações necessárias para a recuperação do arquivo, tais como os metadados do sistema de arquivos. Os fragmentos não identificados podem ser reconhecidos pelo modelo de classificação (como os fragmentos dos tipos T1 e T2 do exemplo), mesmo sem as informações de assinaturas de arquivo ou de metadados do sistema de arquivos. Em seguida, um programa de recuperação de arquivo pode ser aplicado sobre os fragmentos classificados para recuperar os arquivos.

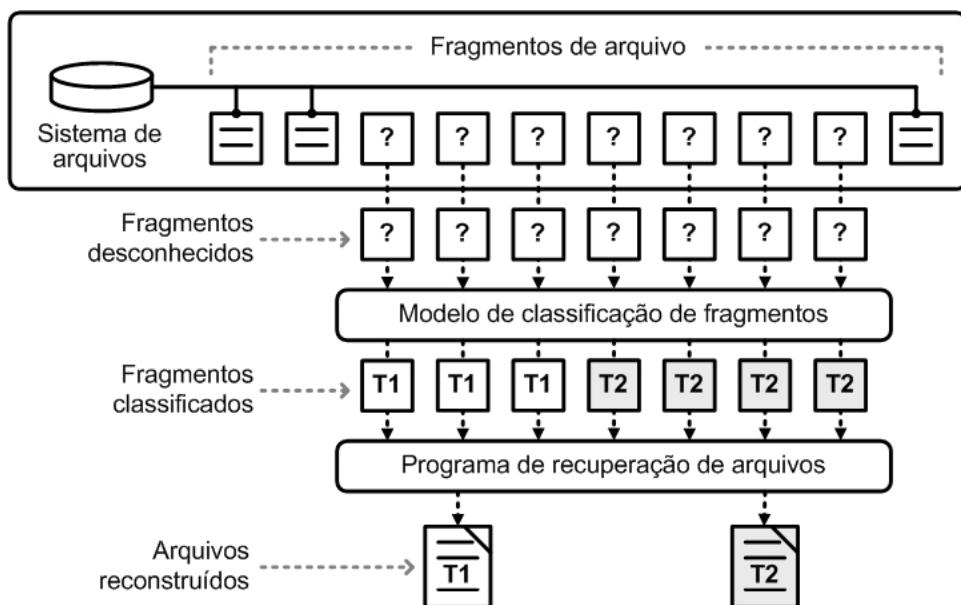


Figura 4.2: Modelo de recuperação de arquivos com utilização dos classificadores.

Dado um fragmento de dado e n tipos de arquivo distintos, existem dois tipos de perguntas que podem ser feitas sobre o tipo de fragmento de dado. O primeiro tipo são de perguntas binárias, nas quais são considerados apenas dois tipos A e B de cada vez e deseja-se saber se o fragmento é do tipo A ou do tipo B. Tais questionamentos são chamados de problemas de classificação binária (ou binomiais) (LI et al., 2011). O segundo tipo de perguntas são aquelas que exigem identificar o fragmento de dado como um dos n tipos, onde $n > 2$. Isto é muitas vezes referido como problemas de classificação multiclasse (ou multinomiais) (LI et al., 2011).

Portanto, os dados para a aprendizagem dos algoritmos podem ser ajustados de modo a produzir dois modelos de classificador:

- *classificador binomial ou binário*: o classificador resultante será capaz de distinguir um tipo dos demais. Dessa forma, é produzido um classificador específico para cada tipo, também chamado de classificador TYPE-X. Esse modelo foi utilizado por Veenman (2007), Calhoun & Coles (2008), que realizaram a classificação de pares de tipos na forma TYPE vs. NOT-TYPE;
- *classificador multinomial ou multiclasse*: o classificador resultante será capaz de distinguir entre múltiplos tipos, ou seja, um conjunto de dois ou mais tipos. Esse modelo foi utilizado por Veenman (2007), Fitzgerald et al. (2012), Xie, Abdullah & Sulaiman (2013) e Xu et al. (2014). Veenman (2007) chama esse modelo de classificador TYPE-ALL.

Para o treinamento e validação do modelo de classificação, pode-se utilizar um conjunto de fragmentos cujos tipos são previamente conhecidos. Após a extração dos atributos, os fragmentos de arquivo passam a ser representados pelos seus vetores de atributos. Uma visão geral do processo é apresentada na Figura 4.3, que mostra a extração de uma base de dados de atributos a partir de um conjunto inicial de fragmentos (T_1 a T_6). Os atributos são ajustados para que o algoritmo de árvores de decisão os processe e gere, no final do fluxo, os modelos de classificação binomial e multinomial.

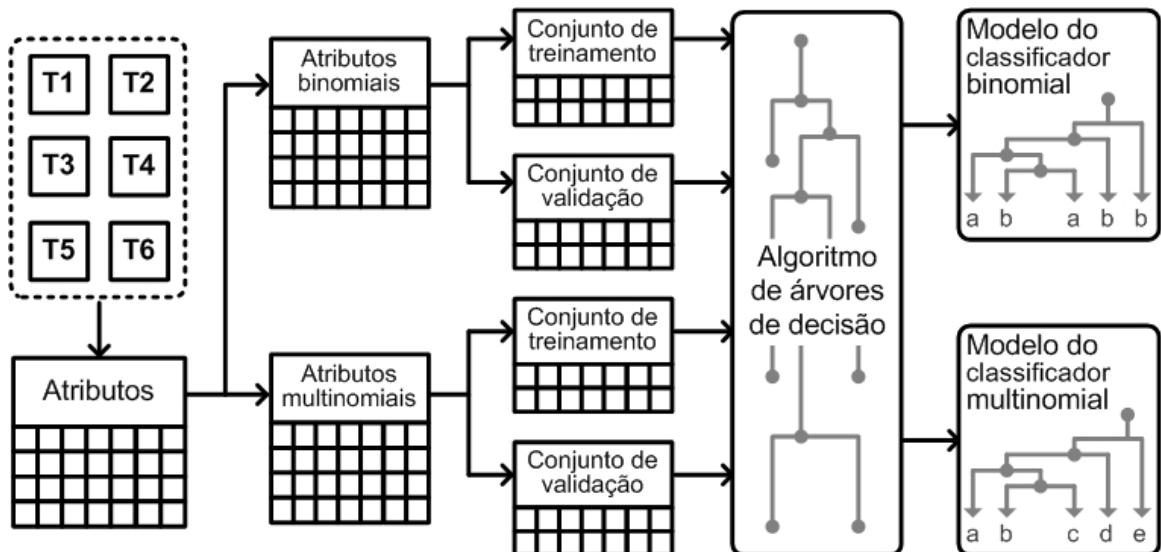


Figura 4.3: Visão geral do processo para produzir os modelos de classificação.

Comumente os classificadores binomiais e multinomiais são utilizados de forma isolada e, nesse sentido, Li et al. (2011) comentam que, em geral, a resposta à perguntas multiclasse não podem ser obtidas de forma confiável e eficiente através de modelos binários e, portanto, é exigido um tratamento diferente. Calhoun & Coles (2008),

ainda, comentam sobre a necessidade de um método que funcione razoavelmente bem para todos os tipos de arquivo.

Roussev & Garfinkel (2009) comentam que o problema mais geral de classificação do fragmento é resolvido pela correlação cruzada dos resultados dos classificadores binomiais. Entretanto, tais resultados podem entrar em conflito, caso em que o método deve sinalizar a discrepância ou deve recusar o fragmento (ROUSSEV; GARFINKEL, 2009).

Uma solução possível é a utilização combinada dos dois modelos de classificador, como ilustrado na Figura 4.4, na qual um classificador multinomial é aplicado sempre que se há dúvidas a respeito do tipo de um fragmento previamente processado por um conjunto de classificadores binários.

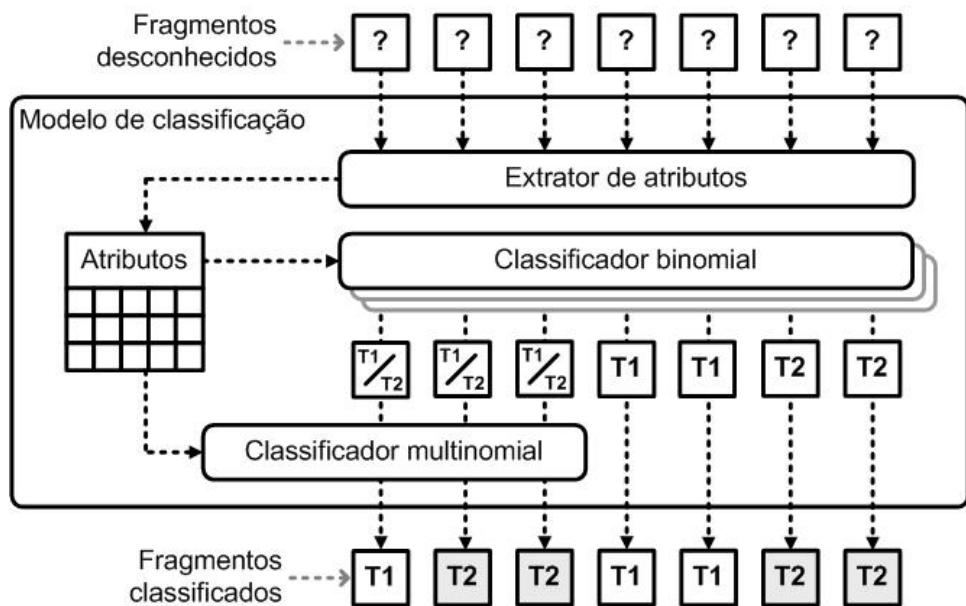


Figura 4.4: Aplicação dos modelos de classificadores binomiais e multinomiais.

O modelo da Figura 4.4 é baseado na solução proposta por Veenman (2007), que sugere o uso de modelos específicos (binomiais) para a análise de um conjunto de *clusters*. Eventualmente alguns *clusters* podem ser rotulados como sendo de vários tipos de arquivo diferentes por causa de falsos positivos dos modelos binomiais. Uma forma de evitar esses rótulos conflitantes, segundo Veenman (2007), é acrescentando o classificador multinomial exclusivo, que pode decidir de forma exclusiva os casos em que alguns tipos são confundidos com os outros.

Em seguida, a partir da identificação correta dos tipos dos fragmentos de interesse,

os quais não foram recuperados através das técnicas tradicionais, como o *data carving*, é possível reconstruir os arquivos através da combinação de suas partes e do uso de validadores para a verificação da reconstrução do arquivo. Na Figura 4.5 é ilustrado o procedimento de reconstrução do arquivo a partir dos fragmentos identificados de cada tipo específico.

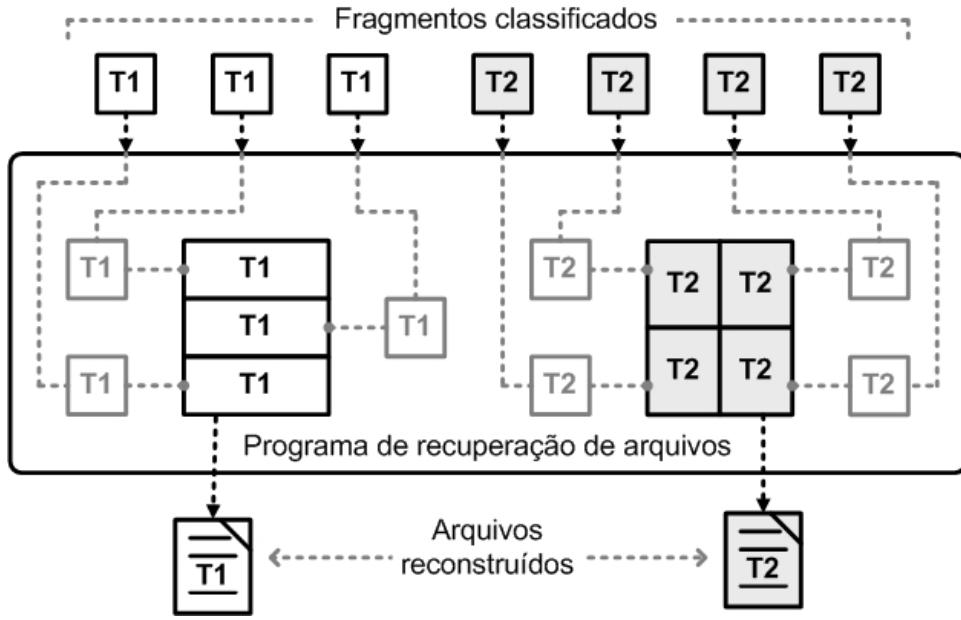


Figura 4.5: Modelo de reconstrução de arquivos.

Esta abordagem fornece uma fonte adicional de informação que pode ser usada no processo de recuperação do arquivo (VEENMAN, 2007). Nesse sentido, os trabalhos de Pal, Shanmugasundaram & Memon (2003), Pal, Sencar & Memon (2008) demonstraram que é possível remontar os arquivos usando modelos de contexto para sequenciar fragmentos dos arquivos. Já Garfinkel (2007) utilizou a validação de objeto através de validadores específicos que tentam determinar se uma sequência de *bytes* é um arquivo válido. A remontagem do arquivo, entretanto, está fora do escopo deste trabalho.

4.4 Conjunto de atributos utilizados no modelo

Cada fragmento é caracterizado pelos valores de atributos que medem diferentes aspectos do fragmento. Existem muitos tipos diferentes de atributos, os quais foram apresentados no Capítulo 3. Para este modelo, os principais atributos utilizados nos experimentos realizados por Veenman (2007), Calhoun & Coles (2008), Axelsson (2010), Conti et al. (2010), Li et al. (2011) e Fitzgerald et al. (2012) foram utilizados. Na Tabela 4.1 são descritos, brevemente, os atributos do modelo.

Tabela 4.1: Descrição dos atributos extraídos de cada fragmento.

NOME DADO AO ATRIBUTO	DESCRIÇÃO DO ATRIBUTO
ascii-percentual	Percentual de caracteres ASCII imprimíveis.
ascii-freq	Frequência de bytes no fragmento com o valor v , onde $x00 \leq v < x20$.
ascii-low-freq	Frequência de bytes no fragmento com o valor v , onde $x20 \leq v \leq x7E$.
ascii-high-freq	Frequência de bytes no fragmento com o valor v , onde $x80 \leq v$.
base64-percentual	Percentual de caracteres com codificação Base64.
base85-percentual	Percentual de caracteres com codificação Base85.
byte-mean	Média dos valores dos bytes do fragmento.
byte-sd	Desvio padrão dos valores dos bytes do fragmento.
byte-cornext	Correlação dos valores dos bytes nas posições n e $n+1$.
chi-square	Calcula o grau de aleatoriedade dos bytes do fragmento.
npl-dist-mean	Calculo média da distância entre os valores de dois bytes consecutivos.
npl-long-seq	Cálculo da sequência mais longa de bytes repetidos.
monte-carlo-pi	Cálculo do valor Monte Carlo para Pi utilizando-se cada sequência sucessiva de 6 bytes como coordenadas X e Y.
serial-correlation	Calcula o grau de dependência de cada byte com o byte anterior.
pattern-[FF,F8]-[1,2,3]gram	Contagem do número de bytes xFF e xF8, inclusive sequências de 1, 2 e 3 gramas (ou seja, sequências de 1, 2 e 3 bytes iguais a xFF e xF8).
hamming-weight	Calcula o total de bits 1 dividido pelo total de bits do fragmento.
histo-[1,2,3]gram-modes	Soma dos 4 maiores valores do histograma de combinações de 1, 2 e 3 bytes.
histo-[1,2,3]gram-sd	Desvio padrão dos valores do histograma de combinações de 1, 2 e 3 bytes.
histo-[1,2,3]gram-cornext	Correlação da frequência dos valores m e $m+1$ do histograma de combinações de 1, 2 e 3 bytes.
lang-detect	Detecção da língua utilizada no texto (ex: português, inglês, espanhol, japonês, etc.)
program-lang	Detecção da linguagem de programação ou de marcação utilizada. (ex: Java, C, Python, HTML, etc).
entropy-[1,2,3]gram	Cálculo do grau de entropia de 1, 2 e 3 gramas (ou seja, combinações de 1, 2 e 3 bytes).
tag-score	Contagem do número de bytes com valores x28, x29, x3C, x3E, x5B, x5D, x7B, x7D.
zero-[1,2]gram-percentual	Percentual de bytes x00 no fragmento, inclusive sequências de 1 e 2 gramas.
zip-compression-score	Taxa de compressão do fragmento.

Ainda sobre a Tabela 4.1, alguns atributos foram agrupados em uma única linha, para facilitar suas descrições, tais como os atributos byte-cornext, pattern-[FF,F8]-[1,2,3]gram, histo-[1,2,3]gram-modes, histo-[1,2,3]gram-sd, histo-[1,2,3]gram-cornext, entropy-[1,2,3]gram e zero-[1,2]gram-percentual. No total, foram considerados 46 atributos extraíveis de cada fragmento de arquivo. A listagem completa dos atributos pode ser vista no Apêndice B.

5 MÉTODO DE TRABALHO

Neste capítulo é apresentado o método de trabalho para produzir os modelos de classificador binomial e multinomial. Dessa forma, o método é descrito na Seção 5.1. Já na Seção 5.2 são descritos os trabalhos correlatos, cujos experimentos também utilizaram técnicas de aprendizagem automática.

5.1 Descrição do método de trabalho

O método de trabalho consiste em 4 fases principais, quais sejam: (1) seleção dos arquivos, (2) extração dos fragmentos dos arquivos, (3) extração dos atributos dos fragmentos e (4) treinamento e validação do classificador. Na Figura 5.1 são apresentadas as fases do método, assim como o fluxo de execução de cada uma delas.

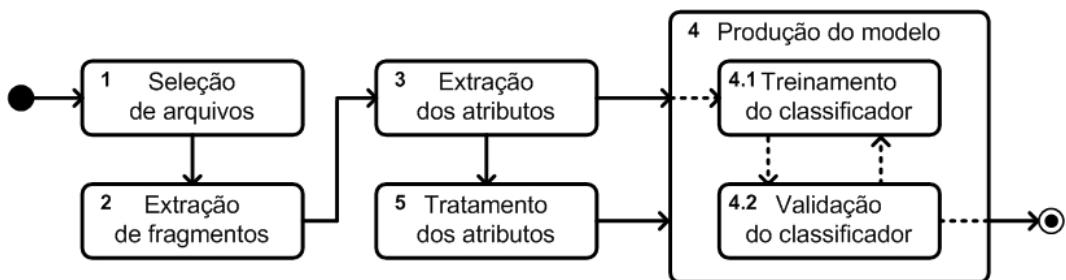


Figura 5.1: Visão geral das fases do método de trabalho.

Nas subseções seguintes serão abordados, em detalhe, cada fase do método de trabalho.

5.1.1 Seleção dos arquivos

No diagrama apresentado na Figura 5.2 são ilustradas as fases (1) e (2) do método de trabalho. Nela pode-se observar como o conjunto inicial de arquivos é filtrado, quando são retirados os arquivos idênticos – através do cálculo e comparação dos *hashes* MD5 (*Message-Digest Algorithm 5*) dos fragmentos –, menores que 10 *Kbytes* e maiores que 100 *Mbytes*.

Os tipos selecionados são aqueles mais bem representados dentro do conjunto total dos arquivos. Dessa forma, são selecionados os tipos que possuíam a maior quantidade

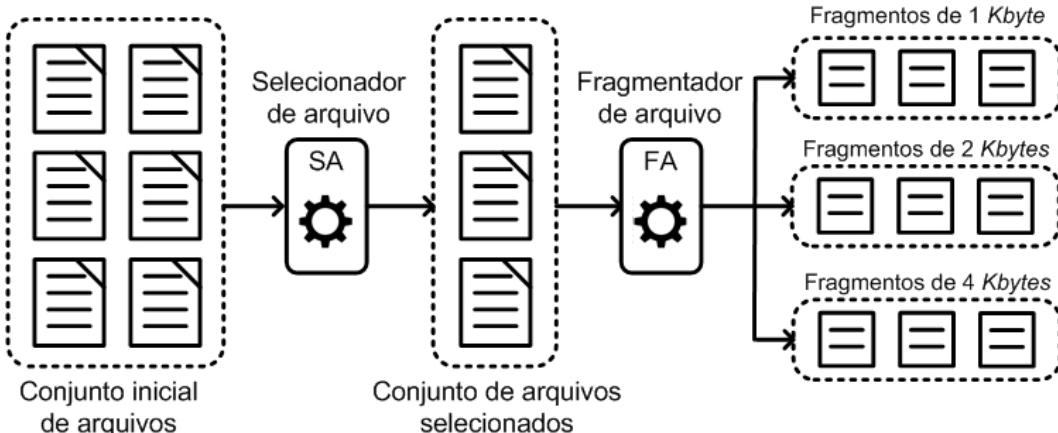


Figura 5.2: Visão geral das fases seleção e fragmentação de arquivo.

de arquivos e a maior quantidade de espaço ocupado em disco. O primeiro critério visa obter a maior variedade possível de arquivos para o experimento; já o segundo visa obter uma quantidade expressiva de fragmentos.

A medida que os arquivos são selecionados, são comparados calculados os valores de *hash* MD5 dos arquivos, a fim de evitar que a amostra possua arquivos idênticos. Portanto, os arquivos que possuam o mesmo valor de *hash* de um arquivo já selecionado são excluídos.

São excluídos, ainda, os arquivos menores que 10 *Kbytes*, os quais não gerariam fragmentos de 4 *Kbytes* úteis para o experimento, uma vez que os fragmentos inicial e final dos arquivos (que podem conter dados de assinaturas do tipo de arquivo) são descartados; e também são excluídos os arquivos maiores que 100 *Mbytes*, os quais produziriam uma grande quantidade não desejada de fragmentos do mesmo arquivo.

5.1.2 Tamanhos dos fragmentos

O tamanho do fragmento é importante porque muitos classificadores, particularmente os baseados em técnicas estatísticas, exigem um tamanho mínimo da amostra para fornecer resultados precisos (CONTI et al., 2010).

Para este método, foram adotados os tamanhos de fragmentos de 1.024, 2.048 e 4.096 *bytes*. Existem três razões para utilizar esses valores de tamanhos de fragmentos:

- As pesquisas de trabalhos relacionados indicam o uso desses tamanhos (VEENMAN, 2007; CALHOUN; COLES, 2008; CONTI et al., 2010; LI et al., 2011; XIE; ABDULLAH; SULAIMAN, 2013; XU et al., 2014).

- O tamanho de bloco maior ajudará a melhorar o desempenho de entrada/saída de disco ao usar arquivos grandes (FITZGERALD et al., 2012).
- Os tamanhos de 1, 2 e 4 *Kbytes* são tamanhos de blocos ou *clusters* tipicamente utilizados por sistemas operacionais atuais (LI et al., 2011; XIE; ABDULLAH; SULAIMAN, 2013).

Os autores Axelsson (2010) e Fitzgerald et al. (2012) utilizaram o tamanho de 512 *bytes*, que é o tamanho padrão dos setores de mídias de armazenamento. Entretanto, este tamanho de fragmento não foi considerado neste trabalho, por dois motivos: os sistemas de arquivos comumente usados hoje têm um tamanho de bloco maior que 1.024 *bytes*, em vez de 512 *bytes* (XU et al., 2014); e os classificadores, particularmente os baseados em técnicas estatísticas, exigem um tamanho mínimo da amostra para fornecer resultados precisos (CONTI et al., 2010).

5.1.3 Extração dos fragmentos

Os arquivos selecionados são, então, fragmentados em tamanhos de 1, 2 e 4 *Kbytes*. Como mostrado na Figura 5.3, depois de processar a fragmentação, o primeiro fragmento de arquivo é removido, pois ele comumente possui metadados de informações de cabeçalho, que normalmente afeta a precisão da classificação (VEENMAN, 2007; FITZGERALD et al., 2012; XIE; ABDULLAH; SULAIMAN, 2013). Igualmente, o último fragmento do arquivo é suprimido, pois ele também pode incluir metadados de informações. Os trabalhos realizados por Calhoun & Coles (2008), Conti et al. (2010), Li et al. (2011), Fitzgerald et al. (2012), Xie, Abdullah & Sulaiman (2013), Xu et al. (2014) também omitem esses fragmentos.

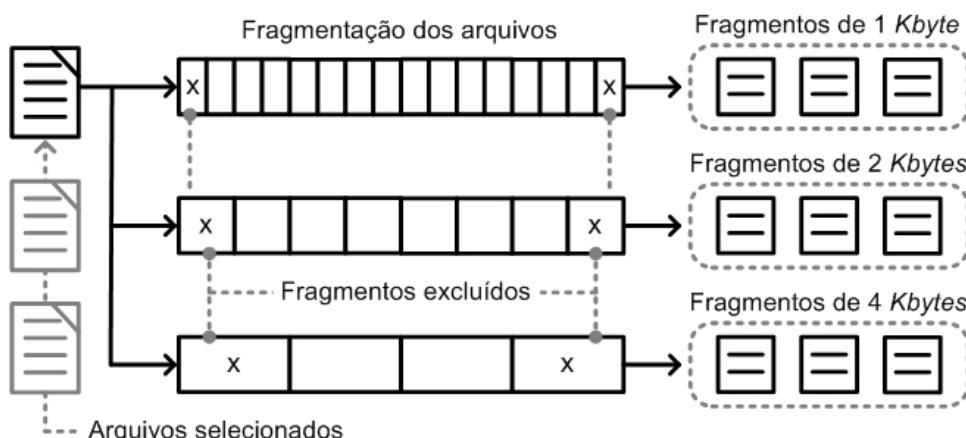


Figura 5.3: Visão geral da fragmentação de arquivos.

São dois os principais motivos para a remoção do primeiro e do último fragmentos. Primeiro, esses fragmentos são facilmente identificáveis usando palavras-chave, uma vez que normalmente há um cabeçalho e um rodapé que marcam o início e o fim do arquivo. Segundo, esses fragmentos geralmente contêm dados que não fazem parte do fluxo de dados principal do arquivo (LI et al., 2011).

O uso dessa abordagem permite gerar um grande conjunto de dados de fragmentos de arquivo com um número igual de fragmentos para cada tipo de arquivo e, ainda, com cada fragmento sendo derivado de uma variedade de arquivos com o mesmo tipo.

5.1.4 Extração dos atributos

Cada fragmento que fornece uma entrada para a aprendizagem automática é caracterizado por seus valores em um conjunto fixo, pré-definido de recursos ou atributos. Dessa forma, a fim de produzir o vetor de atributos dos fragmentos, é extraído um conjunto de atributos dos fragmentos de 1, 2 e 4 *Kbytes*. São selecionados atributos em nível de *bits*, *bytes* (considerando também as contagens de 1, 2 e 3-gramas), caracteres e palavras. Entretanto, não são extraídos os atributos baseados no contexto do fragmento, tal como a ordem dos fragmentos no arquivo.

No diagrama apresentado na Figura 5.4 é representado o *workflow* de execução da fase (3), quando são extraídos um conjunto de atributos dos fragmentos dos arquivos. Nela pode-se observar um componente de extração de atributos (EA), que é formado por um conjunto de extratores especializados em extraer determinados atributos dos fragmentos (EA1, EA2, ..., EAn).

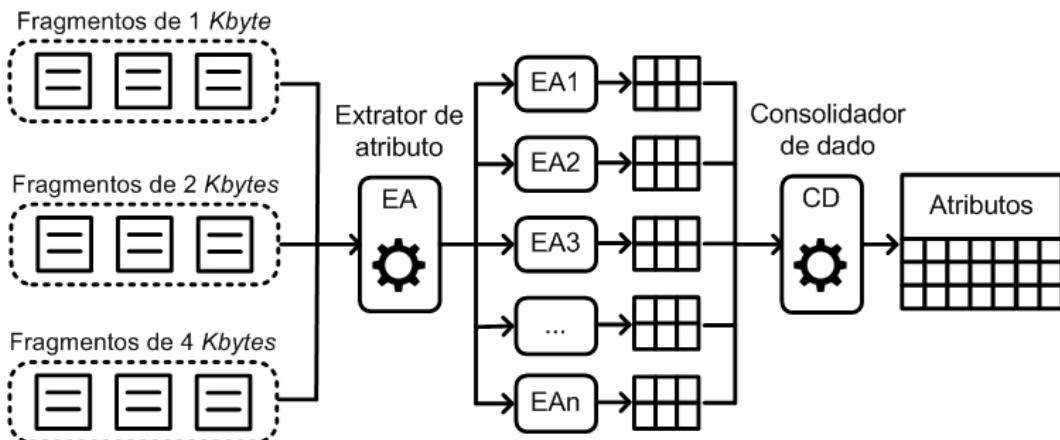


Figura 5.4: Visão geral da fase de extração dos atributos dos fragmentos.

Ao final dessa fase, os vetores de atributos são consolidados em um conjunto único

de dados. Assim, a partir desta fase, os fragmentos passam a ser representados pelos seus dados de atributo.

5.1.5 Treinamento dos classificadores

A fase (4) do método de trabalho consiste em transformar o conjunto de dados dos atributos em conjuntos binomiais e multinomiais, ambos para o aprendizado supervisionado do algoritmo de classificação baseado em árvores de decisão. Para treinar uma árvore de decisão, primeiro deve-se reunir fragmentos de arquivo com tipos conhecidos e, em seguida, deve-se alimentar os vetores de atributo dos fragmentos de dado e suas informações de tipo para o algoritmo. Por fim, o resultado é um conjunto reutilizável de modelos de classificadores binomiais e multinomiais. Na Figura 5.5 é ilustrada essa fase.

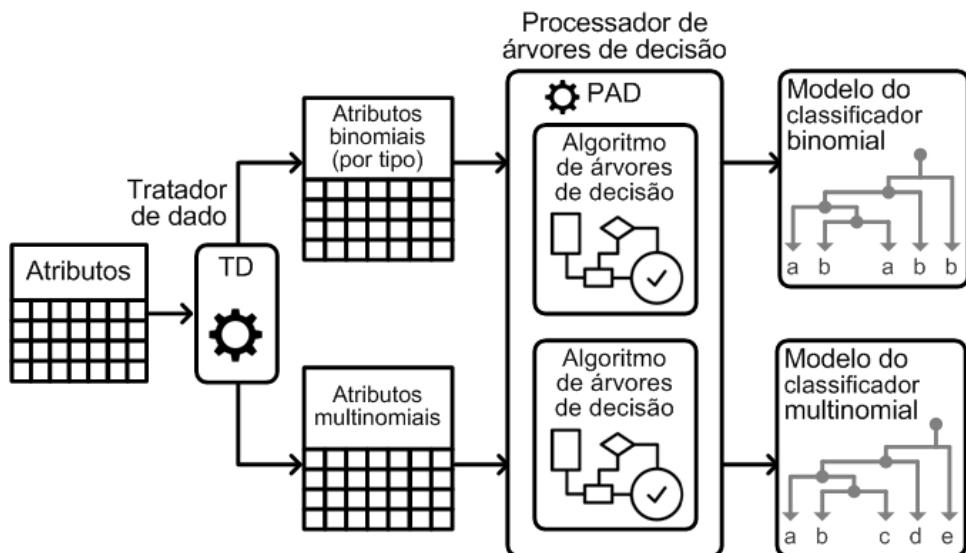


Figura 5.5: Visão geral da fase de treinamento e validação dos classificadores.

5.1.6 Validação dos classificadores

A validação dos modelos de classificador foi através da validação cruzada repetida sobre a base de dados estratificada. Especificamente foi usado o modelo *k-fold cross validation*, o qual foi indicado por Kahavi (1995) como a melhor escolha para se obter uma estimativa precisa e, ademais, foi o modelo utilizado por Konstantinos (2015).

Essa abordagem envolve dividir aleatoriamente o conjunto de observações em *k* grupos, ou *folds*, de tamanhos aproximadamente iguais. O primeiro *fold* é tratado como um conjunto de validação, e o método é treinado nos *k – 1* *folds* restantes (JAMES et al., 2013).

O erro quadrático médio (MSE) é então calculado sobre as iterações do k -fold cross validation. Este procedimento é repetido k vezes; a cada vez, um grupo diferente de observações é tratado como um conjunto de validação. Este processo resulta em k estimativas do erro de teste ($MSE_1, MSE_2, \dots, MSE_k$). A estimativa do cross validation (CV) é calculada pela média desses valores de MSE (JAMES et al., 2013), conforme a Equação 5.1:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2 \quad CV_{(k)} = \frac{1}{k} \sum_{i=1}^k MSE_i \quad (5.1)$$

A Figura 5.6, adaptada de James et al. (2013), apresenta um exemplo de validação cruzada do tipo 5 -fold, na qual a cada iteração um conjunto é destacado para o treinamento, enquanto o restante dos conjuntos são utilizados para o teste.

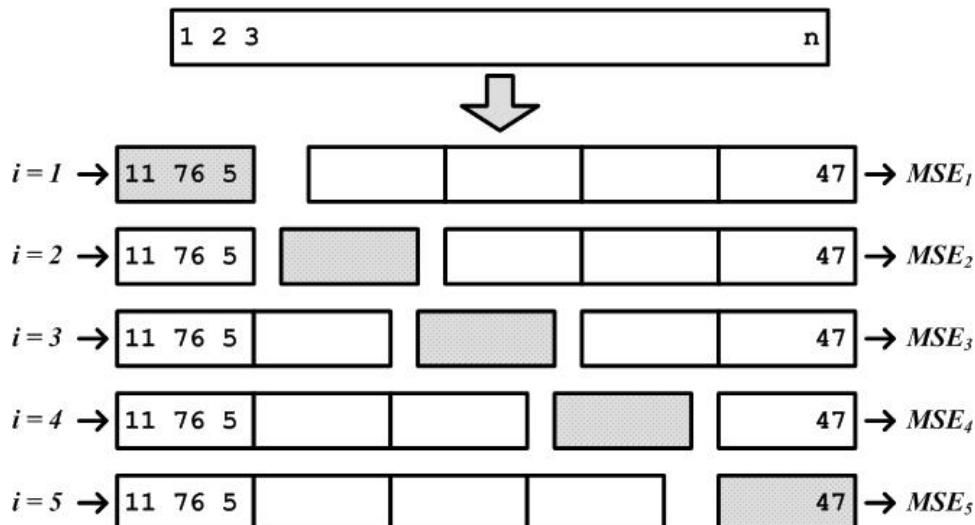


Figura 5.6: Exemplo de validação cruzada do tipo 5 -fold (adaptada de James et al. (2013)).

5.1.7 Tratamento dos atributos

As técnicas de transformação de dados são aplicadas sobre o conjunto de dados de treinamento com o objetivo de ajustar os dados de entrada para o algoritmo de aprendizagem automática. Em alguns casos, tal ajuste pode auxiliar o algoritmo a aprender mais rápido e a obter um melhor desempenho (XU et al., 2014). Para o modelo de classificação são considerados as técnicas de normalização, discretização e seleção de atributo.

A normalização dos dados dimensiona os valores de atributos para um intervalo

específico, geralmente entre 0 e 1. Cada valor de atributo X é ajustado com base nos valores mínimos (min) e máximos (max) do conjunto de dados, conforme Equação 5.2:

$$X = \frac{X - min}{max - min} \quad (5.2)$$

A discretização é uma técnica de transformação que divide o intervalo de um atributo contínuo em um conjunto de intervalos. As formas mais comuns para selecionar os intervalos são por (KALUZA, 2016):

- igual largura: o intervalo da variável contínua é dividido em k intervalos de igual largura;
- igual frequência: suponha que existem N instâncias, cada um dos k intervalos contém aproximadamente N/k instâncias;
- entropia mínima: a abordagem recursivamente divide os intervalos até que a entropia diminui mais do que o aumento da entropia, introduzida pela divisão de intervalo.

A redução de dados lida com atributos e instâncias abundantes. O número de atributos corresponde ao número de dimensões no conjunto de dados. Atributos com baixo poder de predição não só contribuem muito pouco para o modelo global, mas também podem reduzir sua performance. Por exemplo, um atributo com valores aleatórios pode introduzir alguns padrões aleatórios que serão captados pelo algoritmo de aprendizagem automática. Para lidar com este problema, há técnicas que removem tais atributos ou, em outras palavras, seleciona os atributos mais promissores. Este processo é conhecido como seleção de atributo (KALUZA, 2016).

Dentre os possíveis benefícios da seleção de atributo, Konstantinos (2015) cita a possibilidade de:

- reduzir o *overfitting*: dados menos redundantes significam menos oportunidades de tomar decisões com base no ruído.
- melhorar a precisão: menos dados enganosos significa que a precisão do modelo aumenta.
- reduzir o tempo de treinamento: menos dados significa que os algoritmos de aprendizado da máquina são mais rápidos.

- simplificar os modelos: modelos mais simples são mais fáceis de serem implantados ou analisados pelos pesquisadores.

5.2 Trabalhos correlatos

Os trabalhos de Veenman (2007), Calhoun & Coles (2008), Axelsson (2010), Conti et al. (2010), Li et al. (2011), Fitzgerald et al. (2012), Xie, Abdullah & Sulaiman (2013) e Xu et al. (2014) também exploraram a aplicação de técnicas de aprendizagem automática com o objetivo de classificar fragmentos de arquivo. Em geral, os experimentos realizados nesses trabalhos variam em número e tamanho dos fragmentos utilizados como base de treinamento e validação, número de tipos de arquivo utilizados na classificação, números de atributos extraídos dos fragmentos e, por fim, a técnica de aprendizagem automática utilizada.

No experimento realizado por Veenman (2007), foram utilizados um conjunto de 3.000 a 20.000 fragmentos por tipos de arquivo, dentre 11 tipos distintos. Foram utilizados fragmentos de 4 *Kbytes* de tamanho, dos quais foram extraídos os atributos de histograma de 1-grama, entropia de *Shannon* de 1-grama e a complexidade de *Kolmogorov*. O autor utiliza um discriminante linear para classificar os fragmentos e obtém uma precisão média do classificador de 45%.

Já Calhoun & Coles (2008) alcançaram uma precisão média de 88,30%, que fora obtida limitando-se a 4 tipos de arquivo (JPG, BMP, GIF e PDF), os quais são analisados aos pares (classificação binomial), através da aplicação de um classificador de regressão linear. O autor extraiu, dos fragmentos, atributos estatísticos (média, desvio padrão e correlação de valores), frequências de caracteres ASCII, entropia de *Shannon* de 1-grama e, ainda, combinou os valores desses atributos.

O trabalho realizado por Axelsson (2010) utilizou o maior número de tipos de arquivo (28 tipos distintos), entretanto, obteve a menor precisão média com o classificador *k-nearest-neighbors* (34%). O autor utilizou 31.541 fragmentos de 512 *bytes*, dos quais extraiu a medida NDC (*Normalised Compression Distance*), que é baseada no cálculo de complexidade de *Kolmogorov*.

O trabalho realizado por Conti et al. (2010) utiliza o classificador *k-nearest-neighbors* aplicado sobre os atributos de medidas estatísticas, entropia de *Shannon* de 2-gramas, peso de *Hamming*, Qui-quadrado e média aritmética, os quais foram extraídos de 14.000 fragmentos de 1 *Kbyte*. Os autores obtiveram uma precisão média de

96%, entretanto, em vez de utilizar tipos específicos de arquivo, os autores utilizaram grupos bastante abrangentes no classificador: randômico/compactado/criptografado, codificado em *Base64*, codificado em *Uuencoding (Unix-to-Unix encoding)*, código de máquina, texto e *Bitmap*.

Através da aplicação de SVM (*Support Vector Machine*), Li et al. (2011) obtiveram uma precisão média de 81,5%. Para seu experimento, os autores extraíram o histograma de 1-grama de fragmentos de 4 *Kbytes* de 3.600 arquivos. Utilizaram, no entanto, apenas 5 tipos de arquivo (JPEG, DLL, EXE, MP3 e PDF).

Dentre os trabalhos apresentados, os autores Fitzgerald et al. (2012) realizaram o experimento com o maior número de atributos extraídos dos fragmentos de 24 tipos de arquivo. Para cada fragmento foram extraídos: histograma 1-grama, histograma 2-gramas, medidas estatísticas, entropia de *Shannon* 1-grama, entropia de *Shannon* 2-gramas, peso de *Hamming*, média aritmética, taxa de compressão do arquivo, média da distância entre *bytes* consecutivos e maior sequência contígua de *bytes* repetidos. Apesar da grande quantidade de atributos considerados, a precisão média de 47,5%, obtida através da aplicação de técnicas de SVM, pode ser considerada baixa.

Também utilizando o SVM, Xie, Abdullah & Sulaiman (2013) obtiveram uma precisão média de 86,18%. Para seu experimento, os autores extraíram, além do histograma de 1-grama, o histograma circular de 1-grama de fragmentos de 4 *Kbytes* de 4.000 arquivos. Utilizaram 10 tipos de arquivo (DLL, EXE, MP3, JPEG, PDF, CSV, HTML, LOG, TXT e XML). Dentre os trabalhos apresentados, o autor Xu et al. (2014) realizou o experimento com 24 tipos de arquivo. De cada arquivo selecionado foram extraídos fragmentos de 1 *Kbyte*. O atributo de escala de cinza das imagens foi utilizado como entrada para a classificação dos fragmentos com as técnicas KNN, SVM, *Naive Bayes* e árvores de decisão. Como resultado o autor obteve o melhor resultado com o classificador KNN (39,70%).

Neste trabalho é explorado o uso de técnica de aprendizagem automática baseada em árvores de decisão. Os classificadores foram obtidos a partir do processamento de 45 atributos extraídos de 2.830.472 fragmentos de arquivo. Foram considerados os três tamanhos distintos de fragmentos (1, 2 e 4 *Kbytes*) e 21 tipos de arquivo.

6 EXPERIMENTOS E RESULTADOS

Neste capítulo é descrito o experimento para produzir e validar os modelos de classificador binomial e multinomial. Este capítulo é dividido em 2 seções. Na Seção 6.1 são descritos as fases do experimento (tais como a seleção dos arquivos, a extração dos fragmentos, a extração dos atributos e o treinamento e validação dos classificadores). Já na Seção 6.2 são apresentados e discutidos os resultados obtidos a partir do experimento.

6.1 Execução do experimento

Utilizando-se do método de trabalho apresentado no Capítulo 5 (vide Figura 5.1), foram realizados vários experimentos com dados de fragmentações de 1, 2 e 4 *Kbytes* de 21 tipos de arquivo.

O processo envolve a seleção de arquivos, a extração de fragmentos e a extração de atributos do fragmento, a produção dos modelos de classificadores e o tratamento dos atributos. Cada uma dessas fases foram automatizadas através de um protótipo desenvolvido em linguagem *Java*.

Ainda com o mesmo protótipo, os dados coletados foram convertidos para o formato ARFF da ferramenta *Weka* (*Waikato Environment for Knowledge Analysis*). Utilizando-se da API do *Weka*, foi aplicado o algoritmo de classificação baseado em árvores de decisão. Nas seções seguintes é descrita cada fase do processo.

6.1.1 Desenvolvimento do protótipo

Um protótipo em linguagem de programação *Java* (ORACLE, 2016) foi desenvolvido para automatizar a extração e processamento dos atributos dos fragmentos e as chamadas dos algoritmos de classificação da API *Weka* (WAIKATO, 2016c). O código do programa compilado para *bytecode* foi executado na máquina virtual *java* da plataforma *Java SE* da *Oracle*.

A implementação automatiza todas as fases do método de trabalho, ou seja, realiza a seleção dos arquivos, a extração dos fragmentos, a extração dos atributos e a consolidação final dos dados. A automatização das atividades se mostrou essencial

para a realização do experimento, dada a grande quantidade de informações e de passos para processar e analisar os dados.

Para cada atributo do modelo de classificação foi desenvolvida uma classe específica que realiza a sua extração. Ressalte-se que o protótipo pode ser facilmente adaptado para extrair novos atributos. Dessa forma, foram implementados algoritmos específicos para extrair os atributos descritos na Tabela 4.1 do Capítulo 4.

O *Weka* possui uma coleção de algoritmos de aprendizagem automática específicos para árvores de decisão. Ainda, o *Weka* contém ferramentas para pré-processamento de dados, classificação, regressão, agrupamento, regras de associação e visualização. É também adequado para o desenvolvimento de novos esquemas de aprendizagem automática (HALL et al., 2009).

Adicionalmente, o protótipo processa e realiza a conversão automática dos dados dos atributos extraídos para o formato ARFF (WAIKATO, 2016a) e, dessa forma, os mesmos dados de atributos podem ser processados com outros algoritmos de classificação de forma independente no ambiente *explorer* do *Weka*.

6.1.2 Sobre o formato de arquivo ARFF

O ARFF (*Attribute-Relation File Format*) é o formato padrão de armazenamento e processamento de arquivos de dados no *Weka*. Trata-se de uma extensão do formato de arquivos CSV (*Comma-Separated Values*) e que utiliza um cabeçalho com informações de metadados a respeitos dos dados armazenados no arquivo. Um arquivo ARFF de atributos extraídos dos fragmentos de arquivo pode, por exemplo, ser escrito da seguinte forma:

```
@relation 1K-21T-47A-ALL
@attribute ascii-freq numeric
@attribute ascii-low-freq numeric
@attribute ascii-high-freq numeric
@attribute base64-percentual numeric
@attribute base85-percentual numeric
@attribute byte-mean numeric
@attribute byte-sd numeric
@attribute chi-square numeric
@attribute monte-carlo-pi numeric
...
@attribute lang-detect {??,EN,SV,BN,VI,HI,DE,PL,HR,IT,RO,PT,ES,...}
@attribute program-lang {??,CSS,C++,XML,HTML,JAVA,C#,DOS,PHP,JS,RUBY}
```

```

...
@attribute class {AVI,BMP,DLL,DOCX,DOC,DWG,EML,EXE,HTM,JAR,JPG,...}
@data
0.037109,0.037109,0.960938,0.002331,0,0,0.016665,
0.158608,0.357556,0.851954,0.913711,1,0.999998,0,
0,0.052632,0,0.02296,0.962664,0.95461,?,0.20079,
0.955752,0.948583,0.218747,0.949853,0.942402,0.274673,
SV,??,0.014862,0.372434,0.059584,0.055126,0.062418,
0,0,0,0,0,0,0,0.956055,0.950147,0.94804,AVI

```

Pode-se observar que as diretivas começam com o símbolo @ e que há uma diretiva para o nome do conjunto de dados (no exemplo, @relation 1K-21T-47A-ALL). Existe ainda uma diretiva para definir o nome e o tipo de dado de cada atributo (por exemplo, @attribute ascii-freq numeric), e outra diretiva para indicar o início dos dados brutos (@data). Na seção de dados brutos, os valores que têm um símbolo de ponto de interrogação indicam um valor desconhecido ou em falta. O formato suporta valores numéricos e nominais.

6.1.3 Seleção do algoritmo de classificação

Foi realizado um experimento com um conjunto reduzido de 21.000 registros de dados de fragmentos de 1, 2 e 4 *Kbytes* (ou seja, cerca de 2% da base total utilizada no experimento principal). Sobre esses registros, foram executados os algoritmos de classificação baseados em árvores de decisão *Decision Stump*, J48, *REPTree*, *Random Forest* e *Random Tree* (WAIKATO, 2016b), a fim de avaliar o percentual de acertos, o tempo de execução e o tamanho físico do modelo do classificador. Todas as execuções dos algoritmos de classificação foram validadas com o k-fold cross validation (com o valor $k = 10$). O resultado é apresentado na Figura 6.1, onde a média dos valores de fragmentos de 1, 2 e 4 *Kbytes* dos parâmetros avaliados são postos em uma escala de 0 a 1.

Dessa forma, o algoritmo J48 foi selecionado para a realização do experimento principal, por apresentar taxa de acertos acima da média, bom tempo de execução na construção do modelo e bom tamanho físico do modelo de classificação – em detrimento, por exemplo, do algoritmo *Random Forest*, que apesar de apresentar a melhor taxa de acertos, teve o pior tempo de processamento e originou um modelo com tamanho físico extremamente grande. Comparado, ainda, com os algoritmos *REPTree* e *Random Tree*, que apesar de apresentarem ótimos tempos de execução e tamanho do modelo, o algoritmo J48 apresentou a melhor taxa de acertos, o que reforçou a escolha por esse algoritmo.

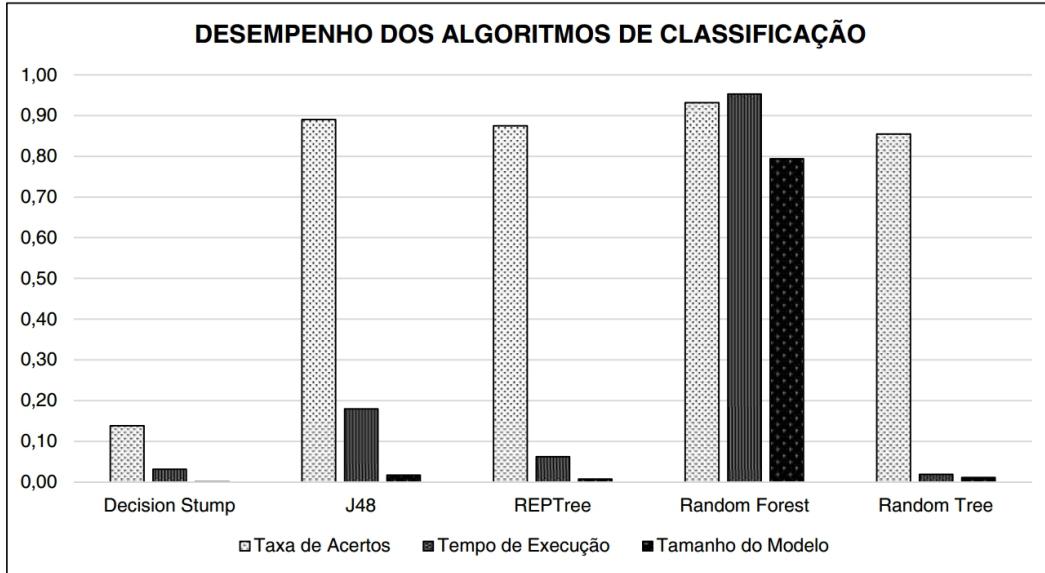


Figura 6.1: Avaliação de desempenho dos algoritmos de árvores de decisão.

6.1.4 Seleção dos arquivos

Foram selecionados cerca de 500 relatórios de evidências periciais de casos reais, cujas investigações ocorreram nos anos de 2014, 2015 e 2016. Dos relatórios, foram extraídos os arquivos de evidências vinculados a eventos de natureza criminal e, dentre esses arquivos, foram selecionados 21 tipos diferentes de arquivos. Na Tabela 6.1 são descritos os tipos selecionados para o experimento.

Após a filtragem dos arquivos para o experimento (como a exclusão de arquivos idênticos, menores que 10 *Kbytes* e maiores que 100 *Mbytes*), foram selecionados 12.153 arquivos, distribuídos em 21 tipos de arquivo, conforme pode ser observado na Tabela 6.2.

6.1.5 Extração dos fragmentos e dos atributos

A partir dos 12.153 arquivos selecionados foram extraídos os fragmentos de 3 tamanhos distintos: 1, 2 e 4 *Kbytes*. Os fragmentos que faziam parte do cabeçalho ou do final dos arquivos foram excluídos, a fim de eliminar qualquer vestígio da assinatura dos arquivos. O resultado final foi a seleção de 2.830.472 fragmentos, divididos da seguinte forma: 945.000 fragmentos de 1 *Kbyte*, 945.000 fragmentos de 2 *Kbytes* e 940.472 fragmentos de 4 *Kbytes*.

Os fragmentos foram selecionados de forma homogênea entre os tipos de arquivo. Para a maioria dos tipos, foram selecionados 45.000 fragmentos por tipo. A exceção

Tabela 6.1: Descrição dos tipos de arquivo selecionados para o experimento.

TIPO	DESCRIÇÃO DO TIPO
AVI	Arquivos de vídeos AVI (<i>Audio Video Interleave</i>).
BMP	Arquivos de imagens <i>Bitmap</i> .
DLL	Arquivo de bibliotecas dinâmicas DLL (<i>Dynamic-Link Library</i>).
DOC	Arquivos de documentos do programa <i>Microsoft Word</i> .
DOCX	Arquivos de documentos do programa <i>Microsoft Word</i> com formato XML.
DWG	Arquivos do programa <i>Autocad</i> .
EML	Arquivos de mensagens de <i>e-mails</i> com codificação <i>Base64</i> .
EXE	Arquivos de programas executáveis.
HTML	Páginas web no formato HTML (<i>Hiper Text Markup Language</i>).
JAR	Arquivos de programas ou bibliotecas em Java.
JPG	Arquivos de imagem JPEG (<i>Joint Photographic Experts Group</i>).
MP3	Arquivos de áudio MP3 (<i>MPEG-1/2 Audio Layer 3</i>).
MP4	Arquivos de vídeos MP4 (<i>MPEG-4 Part 14</i>).
PDF	Arquivos de documentos do programa <i>Adobe Acrobat Reader</i> .
RTF	Arquivos de documentos RTF (<i>Rich Text Format</i>).
SQLL	Arquivos de banco de dados <i>SQLite</i> .
TIF	Arquivos de imagens TIFF (<i>Tagged Image File Format</i>).
TXT	Arquivos de texto simples.
WMV	Arquivos de vídeos WMV (<i>Windows Media Player</i>).
XML	Arquivos de documentos XML (<i>Extensible Markup Language</i>).
ZIP	Arquivos de documentos compactado no formato <i>Winzip</i> .

Tabela 6.2: Quantitativos dos arquivos selecionados para o experimento.

TIPO	TAMANHO (KB)	% DO TOTAL	QUANTIDADE	% DE ARQUIVOS
AVI	195.233	4,76%	12	0,10%
BMP	195.311	4,76%	199	1,64%
DLL	195.313	4,76%	300	2,47%
DOC	195.312	4,76%	901	7,41%
DOCX	195.307	4,76%	716	5,89%
DWG	195.309	4,76%	269	2,21%
EML	195.310	4,76%	198	1,63%
EXE	195.306	4,76%	72	0,59%
HTM	195.312	4,76%	2.424	19,95%
JAR	195.312	4,76%	275	2,26%
JPG	195.193	4,76%	265	2,18%
MP3	195.306	4,76%	136	1,12%
MP4	195.211	4,76%	88	0,72%
PDF	195.214	4,76%	358	2,95%
RTF	195.298	4,76%	2.020	16,62%
SQLI	195.312	4,76%	401	3,30%
TIF	194.737	4,75%	33	0,27%
TXT	195.310	4,76%	1.320	10,86%
WMV	195.308	4,76%	29	0,24%
XML	195.312	4,76%	1.886	15,52%
ZIP	195.308	4,76%	251	2,07%

ocorreu para os tipos HTML, RTF e XML, para os quais foram selecionados 42.578, 43.783 e 44.111 fragmentos de 4 *Kbytes*, respectivamente, e cujos arquivos são, em média, pequenos e, dessa forma, geram menos fragmentos de 4 *Kbytes*.

A partir dos fragmentos selecionados, foram extraídos os principais atributos utilizados nos experimentos realizados por Veenman (2007), Calhoun & Coles (2008), Axelsson (2010), Conti et al. (2010), Li et al. (2011), Fitzgerald et al. (2012), Fitzgerald et al. (2012); os quais foram descritos nos Capítulos 3 e 4. Portanto, a partir dessa fase do experimento, cada fragmento passa a ser representado pelo seu conjunto de atributos.

6.1.6 Produção dos classificadores

O conjunto de atributos foram consolidados e, em seguida, convertidos para o formato ARFF, o qual é o formato de arquivo utilizado pelo projeto *Weka*. Por meio da API do *Weka*, foi aplicado o algoritmo de árvores de decisão J48.

O algoritmo J48, cuja implementação é baseada no algoritmo C4.5 (QUINLAN, 1993), percorre todos os atributos de modo a identificar aquele que apresenta o maior ganho de informação (ou seja, maior contribuição para o resultado na árvore). Após identificar esse atributo, ele é definido como um nó na árvore (ou a raiz, caso seja a primeira iteração) (BELL, 2014).

Neste trabalho, foi realizada a classificação binomial na forma TYPE vs. NOT-TYPE, na qual a cada execução o fragmento é classificado como de um tipo específico ou como diferente do tipo específico. Também foi realizada a classificação multinomial, na forma TYPE-ALL, na qual o classificador resultante será capaz de distinguir qualquer um dos 21 tipos. O número de execuções do algoritmo J48 foi 66, distribuídas da seguintes forma:

- Classificados binomial com o algoritmo J48:
 - 21 execuções sobre os dados de fragmentos de 1 *Kbyte* (por tipo);
 - 21 execuções sobre os dados de fragmentos de 2 *Kbytes* (por tipo);
 - 21 execuções sobre os dados de fragmentos de 4 *Kbytes* (por tipo).
- Classificados multinomial com o algoritmo J48:
 - 1 execução sobre os dados de fragmentos de 1 *Kbyte*;
 - 1 execução sobre os dados de fragmentos de 2 *Kbytes*;
 - 1 execução sobre os dados de fragmentos de 4 *Kbytes*.

Em seguida é listada, como exemplo, a árvore de decisão extraída após o processamento do algoritmo J48 para a classificação binomial dos fragmentos de 1 *Kbyte*

do tipo EML. Cada linha segue o formato [regra-de-decisão] : [classificação] ([total-de-instâncias] / [erro-de-classificação]).

```

base64-percentual <= 0.998047
| ascii-percentual <= 0.999023: NOT-EML (774069.0)
| ascii-percentual > 0.999023
| | histo-2gram-sd <= 0.071029
| | | ascii-freq <= 0.987305
| | | | histo-2gram-modes <= 0.042198
| | | | | ascii-freq <= 0.979492
| | | | | base64-percentual <= 0.843137
| | | | | tag-score-3C <= 0.055172: EML (22.0/2.0)
| | | | | tag-score-3C > 0.055172: NOT-EML (2.0)
... linhas removidas ...
Correctly Classified Instances 944817 99.9806 %
Incorrectly Classified Instances 183 0.0194 %
... linhas removidas ...
Confusion Matrix:
a b <- classified as
44880 120 | a = EML
63 899937 | b = NOT-EML

```

No final do resultado da validação do classificador é apresentado o percentual de classificações corretas e incorretas e a matriz de confusão.

Como apresentado no Capítulo 5, a validação dos modelos de classificadores foi através da validação cruzada, especificamente foi usado o modelo *k-fold cross validation* com o valor de $k = 10$, ou seja, o conjunto de dados dos atributos foi particionado em um conjunto de treinamento e um conjunto de testes, na proporção de 9 para 1. Para tanto, foi utilizado o algoritmo de validação do *Weka* nas suas configurações padrão (excetuando o valor de $k = 10$).

6.1.7 Tratamento dos atributos

Foram aplicados os filtros de normalização, discretização e seleção de atributo providos pela API do *Weka*. O processo de aplicação de filtros no *Weka* é dividido em duas partes: *attribute evaluator* (método pelo qual os subconjuntos de atributos são avaliados) e *search method* (método pelo qual o espaço de possíveis subconjuntos é pesquisado). Para os experimentos realizados neste trabalho, foram utilizados os algoritmos *CfsSubsetEval* (para avaliar os atributos) e *GreedyStepWise* (para pesquisar o conjunto de dados) (REMCO; FRANK et al., 2010):

- *CfsSubsetEval*: avalia os valores dos subconjuntos que se correlacionam altamente com o valor da classe e possuem baixa correlação uns com os outros;
- *GreedyStep Wise*: usa uma estratégia passo a passo para adiante (aditiva) ou para trás (subtrativa) para navegar subconjuntos de atributos.

A Figura 6.2 ilustra o *workflow* de aplicação dos filtros sobre os dados dos atributos. Após cada aplicação de um filtro, os dados de atributos foram processados pelo algoritmo de árvores de decisão para gerar os modelos binomiais e multinomiais.

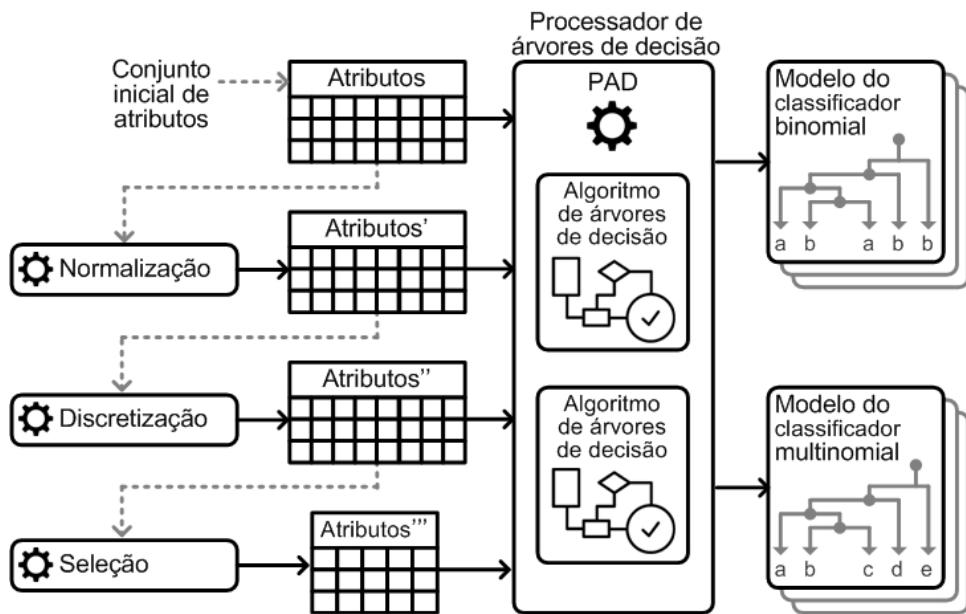


Figura 6.2: Visão geral do *workflow* de aplicação dos filtros de normalização, discretização e seleção de atributo.

Para os experimentos relacionados à aplicação de filtros sobre a base de dados dos atributos, foram realizadas mais 198 execuções do algoritmo J48 (para cada um dos 3 filtros aplicados, foram 66 execuções, sendo 63 para os classificadores binomiais e 3 para os classificadores multinomiais).

6.2 Resultados obtidos

Após a execução dos experimentos, todos os resultados em ambos os modelos são apresentados para comparação e análise. Nesta seção, são apresentados os resultados dos modelos de classificação binomial e multinomial, do *ranking* dos atributos e, por fim, da comparação com os resultados dos outros autores.

Ao final do processamento do algoritmo J48, foram obtidos 63 classificadores binomiais (específicos para cada um dos 21 tipos e para cada tamanho de fragmento de

1, 2 e 4 *Kbytes*) e 3 classificadores multinomiais (específicos para cada um dos tamanhos de 1, 2 e 4 *Kbytes*). As taxas de precisão foram consolidadas e são apresentadas nas seções seguintes.

6.2.1 Classificadores binomiais

Os classificadores binomiais apresentaram melhores percentuais de acerto quando comparados com os classificadores multinomiais, apresentando uma média de 98,78% de classificações corretas (contra uma média de 86,05% do classificador multinomial). Este resultado era esperado, visto que a classificação multinomial envolve uma complexidade maior em distinguir um fragmento dentro de uma variedade de 21 tipos distintos.

Resultado do modelo de classificação binomial (sem a aplicação dos filtros de normalização, discretização e seleção de atributo) é apresentado no gráfico da Figura 6.3. Nela pode-se observar os percentuais de acerto dos classificadores por tipos de arquivo e por tamanhos de fragmento.

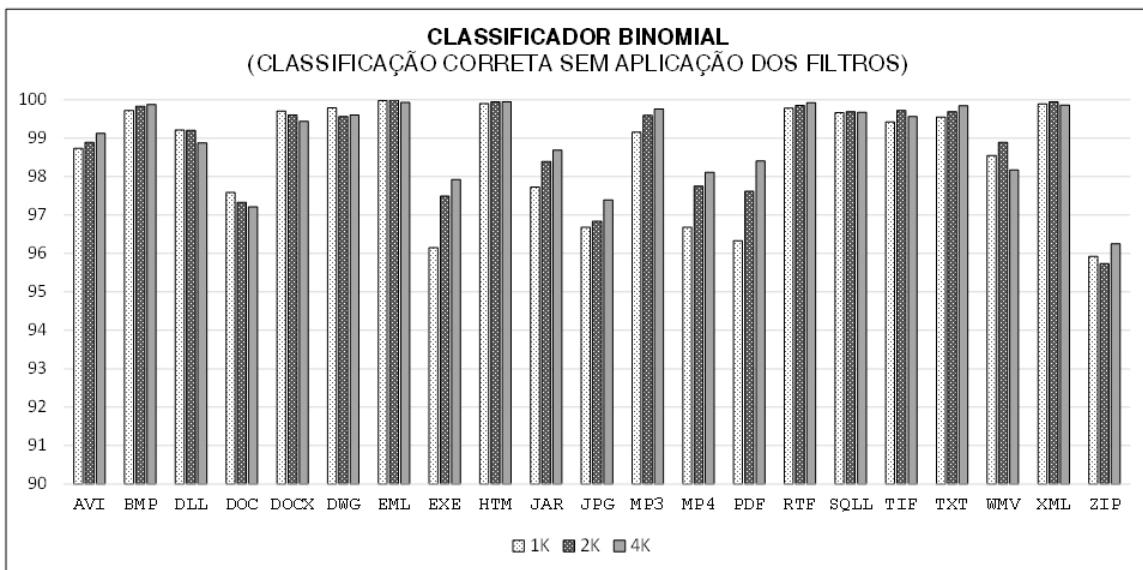


Figura 6.3: Gráfico com os resultados das classificações corretas do classificador binomial, sem aplicação de filtros, por tipos de arquivo e por tamanhos de fragmento.

Fitzgerald et al. (2012) comentam que é provável que o maior tamanho do fragmento de arquivo facilite a tarefa de classificação. Tal fenômeno pode ser observado nos tipos AVI, BMP, EXE, JAR, JPG, MP3, MP4, PDF e TXT; cujos fragmentos de 4 *Kbytes* tiveram melhores desempenhos que os fragmentos menores.

Os tipos DLL, DOC e DOCX tiveram um comportamento inverso, no qual a maior

quantidade de informação dos fragmentos torna a classificação mais difícil, levando o classificador a cometer mais erros de classificação.

Já os tipos EML, HTM, RTF e XML apresentaram desempenhos semelhantes para todos os tamanhos de fragmentos. Cabe notar que esses tipos apresentam altas taxas de acerto, mostrando que mesmo com fragmentos pequenos é possível fazer a classificação dos fragmentos com altas taxas de acertos.

Os tipos DWG, SQLITE, TIF, WMV e ZIP apresentaram comportamentos não esperados para os fragmentos de 2 *Kbytes*. Nesses tipos de arquivo, houve uma classificação semelhante para os fragmentos de 1 e 4 *Kbytes*; e para os fragmentos de 2 *Kbytes* houve variações superiores e inferiores.

Na Tabela 6.3 são apresentados os percentuais de classificações corretas para cada um dos tipos, agrupados por tamanhos dos fragmentos (a tabela ordenada por tipo de arquivo pode ser vista no Apêndice C). Podemos notar que fragmentos com conteúdos predominantemente textuais apresentam melhores desempenhos, tais como os tipos EML, HTM e XML; e, por outro lado, tipos compactados ou containers, como PDF, EXE, ZIP e DOC, apresentam piores performances.

Tabela 6.3: Comparativo dos percentuais de classificações corretas do classificador binomial.

FRAGMENTOS DE 1 KBYTE		FRAGMENTOS DE 2 KBYTES		FRAGMENTOS DE 4 KBYTES	
EML	99,98	EML	99,99	HTM	99,95
HTM	99,90	HTM	99,94	EML	99,93
XML	99,89	XML	99,94	RTF	99,92
DWG	99,79	RTF	99,85	BMP	99,88
RTF	99,78	BMP	99,83	XML	99,86
BMP	99,72	TIF	99,72	TXT	99,84
DOCX	99,70	SQLITE	99,69	MP3	99,76
SQLITE	99,66	TXT	99,69	SQLITE	99,67
TXT	99,55	DOCX	99,60	DWG	99,60
TIF	99,42	MP3	99,59	TIF	99,56
DLL	99,21	DWG	99,56	DOCX	99,44
MP3	99,15	DLL	99,20	AVI	99,12
AVI	98,73	AVI	98,89	DLL	98,87
WMV	98,55	WMV	98,89	JAR	98,69
JAR	97,72	JAR	98,39	PDF	98,40
DOC	97,59	MP4	97,75	WMV	98,17
JPG	96,68	PDF	97,62	MP4	98,11
MP4	96,68	EXE	97,49	EXE	97,92
PDF	96,33	DOC	97,33	JPG	97,39
EXE	96,15	JPG	96,83	DOC	97,21
ZIP	95,92	ZIP	95,73	ZIP	96,25

6.2.2 Classificadores multinomiais

Os classificadores multinomiais, apresentaram uma queda em seus percentuais de classificação (uma média de 86,05% de classificações corretas). No caso específico do classificador multinomial, como era esperado, há uma redução no percentual de acerto do modelo, em troca da flexibilidade de se ter um único modelo capaz de classificar todos os tipos.

O resultado do modelo de classificação multinomial (sem a aplicação dos filtros de normalização, discretização e seleção de atributo) é apresentado no gráfico da Figura 6.4. Nela pode-se observar os percentuais de acerto dos classificadores por tipos de arquivo e por tamanhos de fragmento.

Diferentemente do classificador binomial, pode-se observar que o classificador multinomial apresenta variações de desempenho mais acentuadas entre os tipos de arquivo e os tamanhos de fragmento. Nos classificadores binomiais a variação é entre 95% e 99%, já os classificadores multinomiais apresentaram uma variação de percentual de acerto entre 52% e 99%.

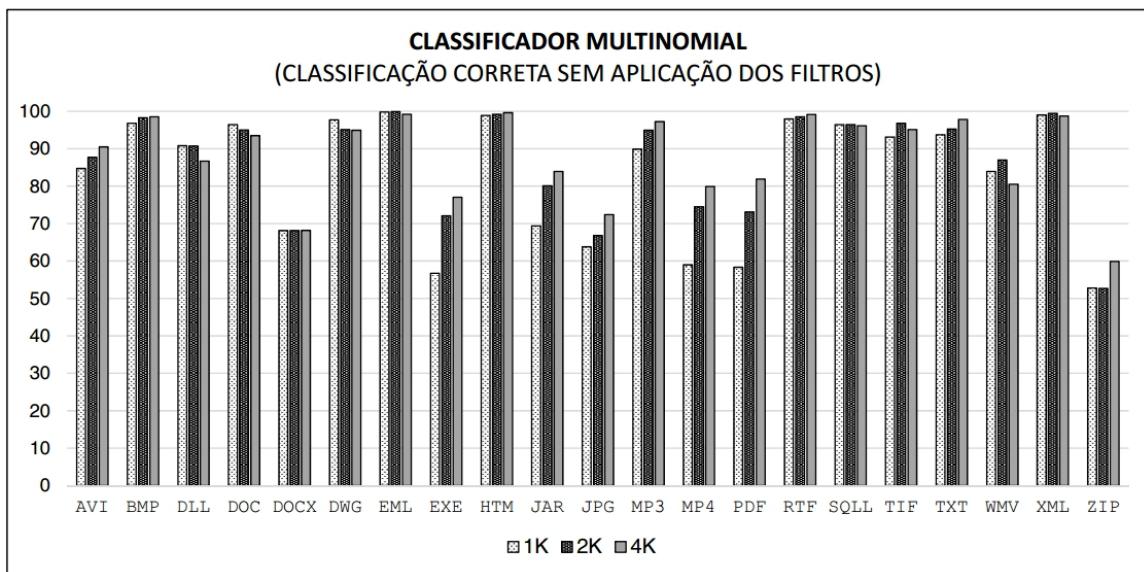


Figura 6.4: Gráfico com os resultados das classificações corretas do classificador multinomial, sem aplicação de filtros, por tipos de arquivo e por tamanhos de fragmento.

Para alguns tipos, fragmentos maiores tiveram desempenhos melhores. Tal fenômeno pode ser observado nos tipos AVI, BMP, EXE, HTM, JAR, JPG, MP3, MP4, PDF, RTF, TXT e ZIP; cujos fragmentos de 4 *Kbytes* tiveram melhores desempenhos que os fragmentos menores.

Os tipos DLL, DOC, DWG, EML e SQLITE tiveram um comportamento inverso, no qual a maior quantidade de informações dos fragmentos torna a classificação mais difícil, levando o classificador a cometer mais erros de classificação.

Já os tipos DOCX e XML apresentaram desempenhos semelhantes para todos os tamanhos de fragmentos.

Os tipos TIF e WMV apresentaram comportamentos não esperados para os fragmentos de 2 *Kbytes*. Nesses tipos de arquivo, houve uma classificação semelhante para os fragmentos de 1 e 4 *Kbytes*; e para os fragmentos de 2 *Kbytes* houve variações superiores e inferiores.

Na Tabela 6.4 são apresentados os percentuais de classificação correta para cada um dos tipos, agrupados por tamanhos dos fragmentos (a tabela ordenada por tipo de arquivo pode ser vista no Apêndice D). Podemos notar que fragmentos com conteúdos predominantemente textuais apresentam melhores desempenhos (assim como nos classificadores binomiais), tais como os tipos EML, HTM e XML; e, por outro lado, tipos compactados ou containers, como PDF, EXE, ZIP e DOC, continuam apresentando os piores desempenhos.

Tabela 6.4: Comparativo dos percentuais de classificação correta do classificador multinomial.

FRAGMENTOS DE 1 KBYTE		FRAGMENTOS DE 2 KBYTES		FRAGMENTOS DE 4 KBYTES	
EML	99,80	EML	99,90	HTM	99,60
XML	99,00	XML	99,40	EML	99,20
HTM	98,90	HTM	99,20	RTF	99,10
RTF	97,90	RTF	98,50	XML	98,70
DWG	97,70	BMP	98,20	BMP	98,50
BMP	96,80	TIF	96,80	TXT	97,80
DOCX	96,40	SQLITE	96,40	MP3	97,20
SQLITE	96,40	TXT	95,20	SQLITE	96,10
TXT	93,70	DWG	95,10	TIF	95,10
TIF	93,10	DOCX	95,00	DWG	94,90
DLL	90,80	MP3	94,90	DOCX	93,50
MP3	89,90	DLL	90,70	AVI	90,50
AVI	84,70	AVI	87,70	DLL	86,70
WMV	83,90	WMV	87,00	JAR	83,90
JAR	69,40	JAR	80,10	PDF	81,90
DOC	68,10	MP4	74,50	WMV	80,50
JPG	63,80	PDF	73,10	MP4	79,90
MP4	59,00	EXE	72,10	EXE	77,00
PDF	58,30	DOC	68,10	JPG	72,40
EXE	56,70	JPG	66,80	DOC	68,20
ZIP	52,80	ZIP	52,70	ZIP	59,90

A seguir, na Tabela 6.5, é apresentada a matriz de confusão do classificador

multinomial para fragmentos de 4 *Kbytes* (outras matrizes de confusão podem ser vistas no Apêndice D). A diagonal da tabela indica o percentual de classificações corretas sobre o total de fragmentos dos tipos.

Tabela 6.5: Matriz de confusão da classificação multinomial do algoritmo J48 (4 *Kbytes*).

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 4 KBYTES (NÃO NORMALIZADO E NÃO DISCRETIZADO)																					
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQL	TIF	TXT	WMV	XML	ZIP	
AVI	91,5	0,0	0,2	0,1	0,2	0,2	0,0	0,2	0,0	0,1	0,0	0,7	2,7	0,2	0,0	0,0	0,4	0,1	2,8	0,0	0,6	
BMP	0,0	98,7	0,2	0,1	0,1	0,1	0,0	0,1	0,0	0,1	0,0	0,0	0,1	0,1	0,0	0,1	0,1	0,1	0,0	0,0	0,0	
DLL	0,2	0,3	86,1	0,3	1,7	1,2	0,0	3,5	0,0	0,8	0,3	0,1	0,7	0,6	0,1	1,4	0,1	0,8	0,7	0,1	1,1	
DOCX	0,1	0,1	0,3	93,8	1,8	0,1	0,0	0,1	0,0	0,2	0,7	0,0	0,2	0,9	0,0	0,4	0,0	0,0	0,2	0,0	1,0	
DOC	0,3	0,1	1,5	1,7	69,6	0,3	0,0	0,7	0,0	0,9	15,0	0,1	0,7	2,3	0,1	0,5	0,3	0,1	1,0	0,0	4,8	
DWG	0,2	0,1	0,9	0,2	0,3	95,6	0,0	0,2	0,0	0,1	0,1	0,0	0,2	0,3	0,0	0,4	0,1	0,2	0,8	0,0	0,2	
EML	0,0	0,0	0,0	0,0	0,0	0,0	99,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,0	0,6	0,0	
EXE	0,2	0,1	4,1	0,1	0,8	0,3	0,0	79,9	0,0	1,8	0,0	0,1	2,6	0,5	0,1	0,3	0,1	0,4	0,3	0,0	8,3	
HTML	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	99,6	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,2	0,0	
JAR	0,2	0,1	0,8	0,3	0,9	0,2	0,0	2,5	0,0	83,1	0,2	0,0	0,9	1,9	0,0	0,2	0,7	0,0	1,0	0,1	7,0	
JPG	0,1	0,0	0,3	0,7	14,6	0,1	0,0	0,0	0,0	0,2	73,6	0,0	0,1	5,1	0,0	0,1	0,2	0,1	0,5	0,0	4,1	
MP3	0,7	0,0	0,1	0,0	0,1	0,1	0,0	0,1	0,0	0,0	97,1	0,9	0,1	0,0	0,0	0,0	0,1	0,6	0,0	0,1		
MP4	2,8	0,0	0,8	0,2	0,8	0,3	0,0	3,8	0,0	0,7	0,1	0,9	79,3	0,7	0,0	0,0	0,1	0,1	6,3	0,0	3,0	
PDF	0,2	0,1	0,7	1,0	2,9	0,4	0,0	1,1	0,0	2,2	5,8	0,1	0,7	79,7	0,0	0,1	0,4	0,0	0,8	0,0	3,5	
RTF	0,0	0,1	0,1	0,1	0,2	0,0	0,0	0,1	0,0	0,1	0,0	0,0	0,1	0,1	98,8	0,0	0,0	0,0	0,1	0,0	0,2	
SQL	0,0	0,1	0,9	0,5	0,5	0,5	0,0	0,2	0,0	0,1	0,0	0,0	0,1	0,1	0,0	96,6	0,0	0,0	0,2	0,0	0,1	
TIF	0,3	0,1	0,2	0,0	0,4	0,1	0,0	0,1	0,0	0,6	0,2	0,0	0,2	0,4	0,0	0,0	95,3	0,0	1,1	0,0	0,9	
TXT	0,0	0,0	0,2	0,0	0,1	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,2	0,0	0,0	0,1	0,0	98,7	0,1	0,1	0,1	
WMV	3,3	0,0	0,7	0,2	1,2	0,8	0,0	0,4	0,0	1,1	0,4	0,5	6,5	0,8	0,0	0,2	1,4	0,1	79,4	0,0	3,0	
XML	0,0	0,0	0,1	0,0	0,0	0,0	0,7	0,0	0,2	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	98,7	0,0		
ZIP	0,9	0,1	1,2	1,1	5,5	0,3	0,0	11,0	0,0	6,9	5,0	0,1	3,3	3,6	0,1	0,1	1,0	0,0	2,9	0,0	56,7	

Assim como constatado no trabalho de Fitzgerald et al. (2012), o classificador teve melhor desempenho em fragmentos de arquivos de entropia baixa (por exemplo, arquivos de texto simples, imagens não compactadas, etc) e pior em fragmentos de arquivo com entropia alta (por exemplo, arquivos compactados, executáveis binários, etc). Esse resultado está consoante com os trabalhos correlatos, ou seja, fragmentos de arquivo de entropia alta são mais difíceis de classificar.

Ainda na Tabela 6.5, foram destacados os maiores percentuais de erro de classificação dos fragmentos. Esses são casos que requerem maiores estudos para introdução de novos atributos que auxiliem nessa classificação. Por exemplo, no caso específico da classificação dos fragmentos dos tipos JPG, 14,7% dos fragmentos foram classificados como fragmentos do tipo DOC e 14,9% dos fragmentos do tipo DOC foram considerados como fragmentos JPG. Esse erro de classificação pode ser atribuído a presença de imagens JPG nos fragmentos do tipo DOC.

Da mesma forma, observa-se que o tipo ZIP tem a pior taxa de acerto (56,7%). A matriz mostra que 10,9% dos fragmentos ZIP foram classificados como fragmentos do tipo EXE. No sentido oposto, 8,5% dos fragmentos EXE foram classificados como fragmentos do tipo ZIP. Isso decorre do fato de que vários outros tipos de arquivo utilizam compressão semelhante àquelas aplicadas nos arquivos ZIP, tais como JAR, DOCX e executáveis comprimidos com *packers* (EXE).

Os casos de falsos-positivos e falsos-negativos têm impactos diretos sobre o processo posterior de recuperação do arquivo. No primeiro, o componente de recuperação tentaria recuperar um fragmento inválido; já no segundo, o componente de recuperação poderia ignorar incorretamente o fragmento válido. Esses cenários indicam a necessidade de classificadores específicos como os classificadores binários apresentados anteriormente. Note-se que vários classificadores binários podem ser aplicados sobre um mesmo fragmento. Se mais de um classificador indicar que aquele fragmento é do seu tipo, o componente de recuperação deverá testar as duas hipóteses. O importante, nesse caso, é que, pelo menos, um dos classificadores gere uma classificação positiva correta, apesar dos demais falsos-positivos.

6.2.3 *Ranking* dos atributos

Na Tabela 6.6 é apresentada os 15 atributos com as melhores pontuações no *ranking* gerado pelos algoritmos *CfsSubsetEval* e *GreedyStepWise* (a tabela completa pode ser vista no Apêndice B).

Tabela 6.6: Relação dos atributos com maiores valores de *ranking* no classificador multinomial, por tamanho de fragmento.

RANK	NOME DO ATRIBUTO		
	FRAG. 1 KBYTE	FRAG. 2 KBYTES	FRAG. 4 KBYTES
1	program-lang	ascii-percentual	entropy-2gram
2	ascii-percentual	histo-1gram-sd	histo-1gram-cornext-t2
3	histo-1gram-sd	byte-cornext1	byte-cornext1
4	monte-carlo-pi	npl-dist-mean	tag-score-3C
5	base85-percentual	histo-1gram-modes	base64-percentual
6	hamming-weight	tag-score-3E	lang-detect
7	byte-cornext1	monte-carlo-pi	histo-2gram-cornext-t2
8	tag-score-3E	histo-1gram-cornext-t2	monte-carlo-pi
9	byte-mean	program-lang	histo-1gram-sd
10	histo-2gram-cornext-t2	base85-percentual	base85-percentual
11	histo-1gram-cornext-t1	base64-percentual	histo-1gram-modes
12	entropy-2gram	hamming-weight	hamming-weight
13	npl-dist-mean	byte-mean	ascii-percentual
14	base64-percentual	entropy-2gram	tag-score-28
15	histo-1gram-modes	tag-score-28	npl-dist-mean

Pode-se notar que vários atributos permaneceram entre os atributos mais impor-

tantes para o classificador, tais como a frequência de caracteres ASCII, codificação *Base64* e *Base85*, histograma de *bytes*, peso de *Hamming*, cálculo de *Monte Carlo Pi* e entropia dos *bytes*.

O estudo detalhado dos atributos, entretanto, é deixado fora do escopo deste trabalho, visto que para a análise adequada dos atributos é necessário definir uma metodologia de trabalho específica para avaliar cada atributo e sua relação com o tipo de fragmento, tipo de classificador e tipo de arquivo. Isto é reforçado pela observação empírica das regras das árvores de decisão, cujos *rankings* de atributo variam bastante entre os classificadores.

6.2.4 Aplicação dos filtros

Foram aplicados os filtros de normalização, discretização e seleção de atributo. O gráfico da Figura 6.5 mostra o tempo de produção do modelo de classificador binomial, a cada aplicação de um filtro. Pode-se observar que a aplicação do filtro de normalização acrescenta um tempo maior para a produção do modelo. No sentido oposto, a aplicação dos filtros de discretização e redução de atributo reduz drasticamente o tempo para se gerar os modelos de classificador.

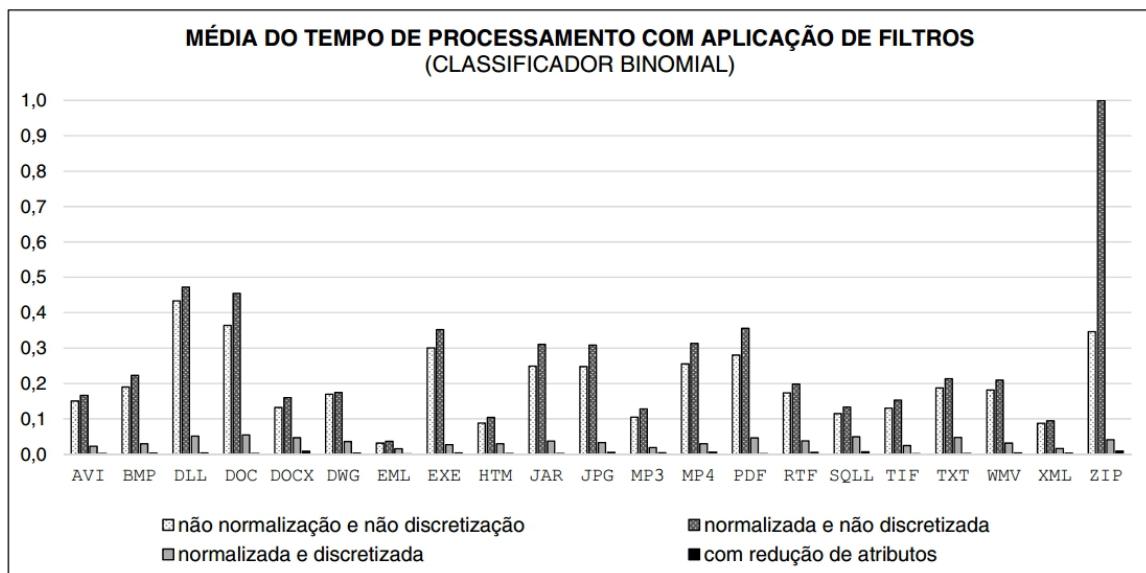


Figura 6.5: Gráfico com os custos de tempo para gerar o classificador binomial a cada aplicação de filtros.

Já o gráfico da Figura 6.6 apresenta o percentual de classificação correta para o classificador binomial a cada aplicação dos filtros de normalização, discretização e seleção de atributo. Ainda no gráfico, cada barra do histograma representa a média dos desempenhos dos fragmentos de 1, 2 e 4 *Kbytes*.

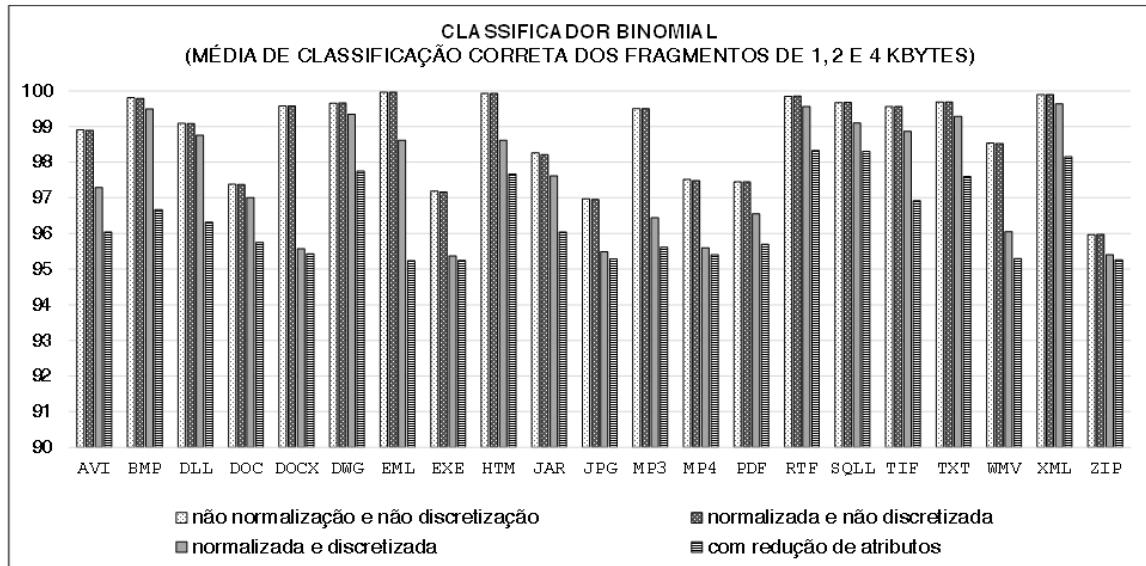


Figura 6.6: Gráfico com os resultados das classificações corretas do classificador binomial após a aplicação dos filtros.

Igualmente, o gráfico da Figura 6.7 apresenta o percentual de classificações corretas para o classificador multinomial a cada aplicação dos filtros de normalização, discretização e seleção de atributo. Da mesma forma, cada barra do histograma representa a média dos desempenhos dos fragmentos de 1, 2 e 4 *Kbytes*.

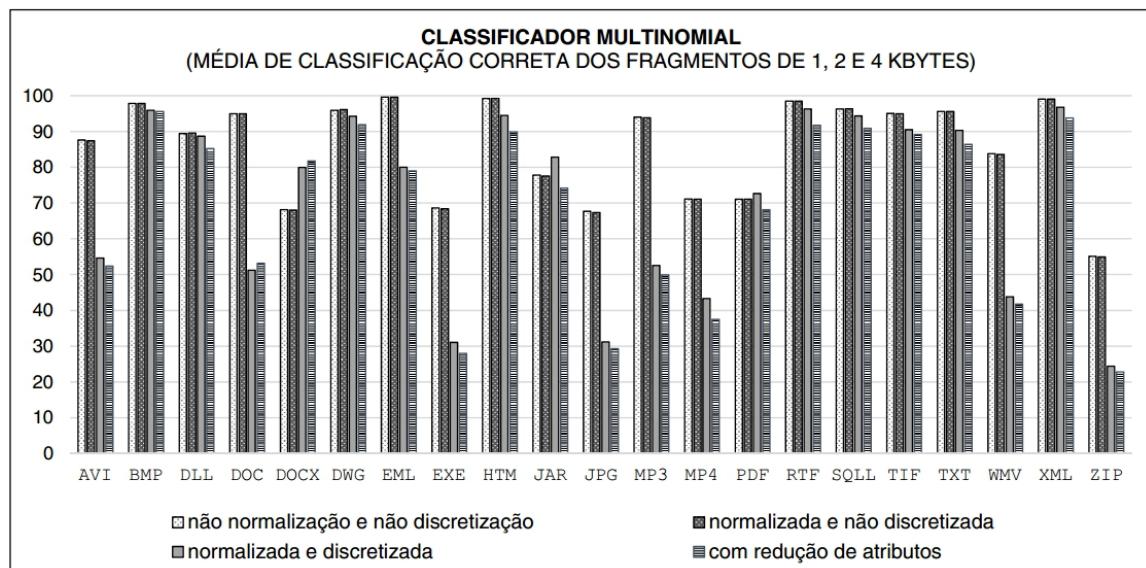


Figura 6.7: Gráfico com os resultados das classificações corretas do classificador multinomial após a aplicação dos filtros.

Os resultados apresentados nos gráficos das Figuras 6.6 e 6.7 mostram que o filtro de normalização tem pouco efeito sobre o classificador binomial e multinomial. Já os filtros de discretização e redução de atributo produzem uma queda drástica nas

performances dos classificadores.

6.2.5 Considerações finais

A Tabela 6.7 é um resumo comparativo dos trabalhos de classificação de fragmento utilizando alguma técnica de aprendizagem automática. Ainda a respeito da Tabela 6.7, algumas considerações:

Tabela 6.7: Comparativo dos métodos e resultados dos experimentos de outros autores.

#	Nome do autor	Método utilizado	Nº de tipos	Nº de atributos	Tamanho dos Fragmentos	Precisão média
A1	Veenman (2007)	Discriminante linear	11	3	4 Kbytes	45,00%
A2	Calhoun e Coles (2008)	Discriminante linear	4	16 ⁽¹⁾	1 Kbyte	88,30% ⁽²⁾
A3	Axelsson (2010)	KNN	28	1 ⁽³⁾	512 bytes	34,00%
A4	Conti et al. (2010)	KNN	6 ⁽⁴⁾	5	1 Kbyte	96,07%
A5	Li et al. (2011)	SVM	5	256 ⁽⁵⁾	4 Kbytes	81,50%
A6	Fitzgerald et al. (2012)	SVM	24	9 ⁽⁶⁾	512 bytes	48,30%
A7	Xie et al. (2013)	SVM	10	256 ⁽⁷⁾	4 Kbytes	86,18%
A8	Xu et al. (2014)	KNN ⁽⁸⁾	29	1 ⁽⁹⁾	1 Kbyte	39,70%
A9	Presente trabalho (2016)	Árvore de Decisão	21	39	1, 2 e 4 Kbytes	98,78% ⁽¹⁰⁾ 86,05% ⁽¹¹⁾

1. Os autores Calhoun & Coles (2008) geraram novos atributos a partir da combinação dos atributos básicos.
2. Calhoun & Coles (2008) obtiveram o percentual médio de 88,30% para os classificadores binomiais.
3. O autor Axelsson (2010) utiliza a distância de compressão normalizada entre os blocos dos fragmentos.
4. Os autores Conti et al. (2010) utilizam, em vez tipos específicos de arquivo, grupos mais abrangentes de tipos de arquivo.
5. Os autores Li et al. (2011) utilizaram um vetor de histograma de *bytes* e consideraram cada frequência como um atributo.
6. Os autores Fitzgerald et al. (2012) utilizaram, entre outros atributos, os histogramas de 1 e 2 *bytes*, nos quais cada frequência foi considerada um atributo (gerando 256 e mais 256*256 atributos).
7. Os autores Xie, Abdullah & Sulaiman (2013) utilizaram o histograma de 1 *byte* (gerando 256 atributos) com o cálculo circular entre os fragmentos (relação entre os fragmentos).

8. Os autores Xu et al. (2014) utilizaram os algoritmos *Naive Bayes*, SVM, Árvores de Decisão e KNN; obtendo o melhor resultado com último algoritmo.
9. Os autores Xu et al. (2014) utilizaram um vetor de escala de cinza.
10. O presente trabalho obteve o percentual médio dos classificadores binomiais de 98,78%.
11. O presente trabalho obteve o percentual médio dos classificadores multinomiais de 86,05%.

As considerações acima, sobre a Tabela 6.7, mostram a dificuldade em comparar os trabalhos relacionados. Este trabalho procurou aproximar-se de outras abordagens ao tentar selecionar o maior número de atributos utilizados por outros autores, o que nem sempre é possível, como nos casos dos trabalhos de Xie, Abdullah & Sulaiman (2013) e Xu et al. (2014) que utilizaram informações do contexto dos fragmentos. No mesmo sentido, utilizou-se de tamanhos de fragmento distintos de 1, 2 e 4 *Kbytes*.

Ademais, este trabalho buscou utilizar o maior número possível de tipos de arquivo (limitado apenas pela quantidade de amostras disponíveis nos relatórios periciais). A falta de padrões nos tipos utilizados nos experimentos deste trabalho e dos outros autores dificulta a comparação da média dos resultados, dado que irá depender dos resultados parciais de classificação de cada tipo de arquivo. Assim, a escolha de determinados tipos de arquivo pode elevar ou reduzir a média do percentual de acertos. Por exemplo, o trabalho de Conti et al. (2010) elevou o seu percentual de acertos na medida que utiliza grupos de tipos de arquivo. Já o trabalho de Axelsson (2010) utilizou os tipos de arquivo JPG, GIF, PPTX, PPS, GZ e PNG, os quais tiveram baixos percentuais de acerto.

7 CONCLUSÕES

Neste trabalho foi explorado o uso de técnicas de aprendizagem automática baseadas em árvores de decisão aplicadas no problema de classificação de fragmentos para posterior recuperação do arquivo. Em cenários nos quais inexistem informações sobre o sistema de arquivos e, ainda, inexistem o cabeçalho e o final do arquivo, a classificação e a recuperação de arquivos se tornam atividades desafiadoras.

Uma dificuldade adicional na classificação é a fragmentação dos arquivos. O modelo de classificação apresentado, baseado em árvores de decisão binomiais e multinomiais, se mostrou eficiente para classificar os fragmentos de arquivo. O modelo apresentado faz uso de 46 atributos extraídos dos fragmentos de arquivo. Para cada atributo, foi implementado um algoritmo que o extraí. Após a aplicação dos algoritmos, foi produzida uma base de dados de atributos de fragmentos de 1, 2 e 4 *Kbytes*.

Além do modelo de classificação de fragmento, foi apresentado um método de trabalho para a seleção dos arquivos, a extração dos fragmentos dos arquivos, a extração dos atributos dos fragmentos e, por fim, o treinamento e validação do classificador. Tal método foi utilizado em experimentos sobre arquivos de evidências de relatórios periciais.

Nos experimentos, foi empregado o algoritmo de classificação J48, considerado um classificador estável e de amplo uso na área de aprendizagem automática, sobre um conjunto de fragmentos extraídos de arquivos de 21 tipos distintos. Como resultado, obteve-se um conjunto de classificadores binomiais especializados para cada um dos 21 tipos de arquivo e um conjunto de classificadores multinomiais para os tamanhos de 1, 2 e 4 *Kbytes*, capazes de classificar entre os 21 tipos de arquivo.

O resultado do experimento mostrou que mesmo quando inexistentes a informação do cabeçalho, do final do arquivo e do sistema de arquivos, é possível classificar um fragmento com percentuais de acerto de 98,78% (com classificadores binomiais) e de 86,05% (com classificadores multinomiais). Logo, considera-se que a utilização de tal técnica pode trazer grandes ganhos para o exame pericial ao complementar as demais formas de recuperação de dados.

O modelo proposto, assim como os resultados dos experimentos, indicam várias possibilidades de trabalhos futuros, tais como: (1) a análise dos atributos utilizados no experimento, (2) o ajuste da proporção dos tipos de fragmentos do experimento, (3) o tratamento de fragmentos de tipos desconhecidos, (4) a execução dos experimentos com outros algoritmos de classificação e (5) o uso de um processo padrão para a comparação com os trabalho relacionados.

Pode-se analisar a necessidade de normalizar ou mesmo discretizar alguns atributos, assim como analisar o custo e o impacto de cada um. Para determinados modelos de classificação, pode-se eliminar alguns desses atributos sem que haja prejuízos significativos aos resultados do modelo.

Ainda, pode-se equilibrar a amostragem de registros para o modelo de classificação binária. O experimento atual foi realizado com uma proporção de 4,76% contra 95,24% dos fragmentos (na forma TYPE vs. NOT-TYPE), de tal forma que algumas árvores de classificação binária se tornaram “especialistas” em identificar fragmentos que não pertencem ao tipo. Experimentos com uma quantidade reduzida de registros mostrou que, após o ajuste para uma proporção de 50% contra 50%, os modelos de classificação têm uma redução média de 5% na taxa de acerto do algoritmo.

Nesse experimento não é dado um tratamento para os fragmentos de tipos desconhecidos no classificador multinomial e, portanto, inexiste uma classe “outros”. Então, se um fragmento de um tipo desconhecido (ex.: GIF) for apresentado, o classificador necessariamente vai indicar um dos 21 tipos esperados, gerando um falso-positivo para aquele tipo.

O modelo de classificação pode ser adaptado para utilizar outras técnicas de aprendizagem automática, tais como o SVM - *Support Vector Machines*, KNN - *k-Nearest Neighbors* e *Nave Bayes*; assim como outros algoritmos baseados em árvores de decisão, tais como LMT (*Logistic Model Trees*), *Random Forest*, *Random Tree*, *REP Tree*, *Decision Stump*, *Hoeffding Tree*. Após a execução dos experimentos, seguindo o mesmo método de trabalho, os resultados podem ser facilmente comparados com os resultados obtidos neste trabalho.

Por fim, foi mostrada a dificuldade de comparação entre os modelos de classificadores, especialmente pela falta de padrões nos experimentos realizados. O processo CRISP-DM - *CRoss Industry Standard Process for Data Mining* (CHAPMAN et al.,

1999), cujo modelo de processo descreve as abordagens comumente usadas em mineração de dados, pode ser aplicado futuramente para a comparação entre os resultados obtidos por outros autores.

REFERÊNCIAS BIBLIOGRÁFICAS

- AHMED, I. et al. Fast content-based file type identification. In: SPRINGER. *IFIP International Conference on Digital Forensics*. [S.l.], 2011. p. 65–75.
- AXELSSON, S. The normalised compression distance as a file fragment classifier. *Digital Investigation*, Elsevier, v. 7, p. S24–S31, 2010.
- BARBETTA, P. A. *Estatística aplicada às ciências sociais*. [S.l.]: Ed. UFSC, 2008.
- BELL, J. *Machine Learning: Hands-on for developers and technical professionals*. [S.l.]: John Wiley & Sons, 2014.
- CALHOUN, W. C.; COLES, D. Predicting the types of file fragments. *Digital Investigation*, Elsevier, v. 5, p. S14–S20, 2008.
- CARRIER, B. *File system forensic analysis*. [S.l.]: Addison-Wesley Professional, 2005.
- CHAPMAN, P. et al. The crisp-dm process model. *The CRIP-DM Consortium*, v. 310, 1999.
- COHEN, M. I. Advanced carving techniques. *Digital Investigation*, Elsevier, v. 4, n. 3, p. 119–128, 2007.
- CONTI, G. et al. Automated mapping of large binary objects using primitive fragment type classification. *Digital Investigation*, Elsevier, v. 7, p. S3–S12, 2010.
- FITZGERALD, S. et al. Using nlp techniques for file fragment classification. *Digital Investigation*, Elsevier, v. 9, p. S44–S49, 2012.
- FOREMOST. *Ferramenta Carving Tool*. 2016. Disponível em: <<http://foremost.sourceforge.net/>>.
- GARFINKEL, S. L. Carving contiguous and fragmented files with fast object validation. *Digital Investigation*, Elsevier, v. 4, p. 2–12, 2007.
- GOLLAPUDI, S. *Practical Machine Learning*. [S.l.]: Packt Publishing Ltd, 2016.
- HALL, M. et al. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, ACM, v. 11, n. 1, p. 10–18, 2009.
- JAMES, G. et al. *An introduction to statistical learning*. [S.l.]: Springer, 2013. v. 6.
- KAHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *International joint conference on artificial intelligence, 1995*. [S.l.: s.n.], 1995. p. 1137–1143.
- KALUZA, B. *Machine learning in Java*. [S.l.]: Packt Publ., 2016.

KONSTANTINOS, K. *File type identification - a computational intelligence approach to digital forensics*. Tese (Doutorado) — Technological Educational Institute of Crete, 2015.

LI, Q. et al. A novel support vector machine approach to high entropy data fragment classification. *SAISMC*, p. 236–247, 2011.

NAKATANI, S. *Language Detection Library for Java*. 2010. Disponível em: <<https://github.com/shuyo/language-detection>>.

ORACLE. *Plataforma Java Standard Edition*. 2016. Disponível em: <<http://www.oracle.com/technetwork/java/index.html>>.

PAL, A.; SENCAR, H. T.; MEMON, N. Detecting file fragmentation point using sequential hypothesis testing. *Digital Investigation*, Elsevier, v. 5, p. S2–S13, 2008.

PAL, A.; SHANMUGASUNDARAM, K.; MEMON, N. D. Automated reassembly of fragmented images. In: *ICME*. [S.l.: s.n.], 2003. v. 3, p. 625–628.

QUINLAN, J. R. *C4.5: programs for machine learning*. [S.l.]: Elsevier, 2014.

QUINLAN, R. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

REMCO, R. B.; FRANK, E. et al. *Weka Manual for Version 3-6-2*. [S.l.]: January, 2010.

ROUSSEV, V.; GARFINKEL, S. L. File fragment classification-the case for specialized approaches. In: IEEE. *Systematic Approaches to Digital Forensic Engineering, 2009. SADFE'09. Fourth International IEEE Workshop on*. [S.l.], 2009. p. 3–14.

ROUSSEV, V.; QUATES, C. File fragment encoding classification - an empirical approach. *Digital Investigation*, Elsevier, v. 10, p. S69–S77, 2013.

SHANNON, M. Forensic relative strength scoring: Ascii and entropy scoring. *International Journal of Digital Evidence*, v. 2, n. 4, p. 1–19, 2004.

VEENMAN, C. J. Statistical disk cluster classification for file carving. In: IEEE. *Third international symposium on information assurance and security*. [S.l.], 2007. p. 393–398.

WAIKATO. *Weka: Attribute-Relation File Format*. 2016. Disponível em: <<http://weka.wikispaces.com/ARFF>>.

WAIKATO. *Weka: Decision Tree Algorithms*. 2016. Disponível em: <<http://weka.sourceforge.net/doc.stable/weka/classifiers/trees/package-summary.html>>.

WAIKATO. *Weka: Waikato Environment for Knowledge Analysis Tool*. 2016. Disponível em: <<http://www.cs.waikato.ac.nz/~ml/>>.

WALKER, J. Ent: A pseudorandom number sequence test program. *Software and documentation available at/www.fourmilab.ch/random/S*, 2008.

XIE, H.; ABDULLAH, A.; SULAIMAN, R. Byte frequency analysis descriptor with spatial information for file fragment classification. In: *Proceeding of the International Conference on Artificial Intelligence in Computer Science and ICT (AICS 2013)*. [S.l.: s.n.], 2013.

XU, T. et al. A file fragment classification method based on grayscale image. *Journal of Computers*, v. 9, n. 8, p. 1863–1870, 2014.

APÊNDICES

A RESUMO DE TRABALHOS RELACIONADOS

Tabela A.1: Comparativo dos atributos de fragmentos de arquivo utilizados pelos autores.

Nome do atributo	A1	A2	A3	A4	A5	A6	A7	A8	A9
ascii-percentual	✗	✓	✗	✗	✗	✗	✗	✗	✓
ascii-freq	✗	✓	✗	✗	✗	✗	✗	✗	✓
ascii-low-freq	✗	✓	✗	✗	✗	✗	✗	✗	✓
ascii-high-freq	✗	✓	✗	✗	✗	✗	✗	✗	✓
base64-percentual	✗	✗	✗	✗	✗	✗	✗	✗	✓
base85-percentual	✗	✗	✗	✗	✗	✗	✗	✗	✓
byte-mean	✗	✓	✗	✓	✗	✓	✗	✗	✓
byte-sd	✗	✓	✗	✗	✗	✗	✗	✗	✓
byte-cornext	✗	✓	✗	✗	✗	✗	✗	✗	✓
chi-square	✗	✗	✗	✓	✗	✗	✗	✗	✓
npl-dist-mean	✗	✗	✗	✗	✗	✓	✗	✗	✓
npl-long-seq	✗	✗	✗	✗	✗	✓	✗	✗	✓
monte-carlo-pi	✗	✗	✗	✗	✗	✗	✗	✗	✓
serial-correlation	✗	✗	✗	✗	✗	✗	✗	✗	✓
pattern-FF-[1,2,3]gram	✗	✗	✗	✗	✗	✗	✗	✗	✓
pattern-F8-[1,2,3]gram	✗	✗	✗	✗	✗	✗	✗	✗	✓
hamming-weight	✗	✗	✗	✓	✗	✓	✗	✗	✓
histo-1gram-modes	✗	✗	✗	✗	✗	✗	✗	✗	✓
histo-1gram-sd	✗	✗	✗	✗	✗	✗	✗	✗	✓
histo-1gram-cornext	✗	✗	✗	✗	✗	✗	✗	✗	✓
histo-2gram-modes	✗	✗	✗	✗	✗	✗	✗	✗	✓
histo-2gram-sd	✗	✗	✗	✗	✗	✗	✗	✗	✓
histo-2gram-cornext	✗	✗	✗	✗	✗	✗	✗	✗	✓
histo-3gram-modes	✗	✗	✗	✗	✗	✗	✗	✗	✓
histo-3gram-sd	✗	✗	✗	✗	✗	✗	✗	✗	✓
histo-3gram-cornext	✗	✗	✗	✗	✗	✗	✗	✗	✓
lang-detect	✗	✗	✗	✗	✗	✗	✗	✗	✓
program-lang	✗	✗	✗	✗	✗	✗	✗	✗	✓
entropy-1gram	✓	✓	✗	✓	✗	✗	✗	✗	✓
entropy-2gram	✗	✗	✗	✓	✗	✓	✗	✗	✓
entropy-3gram	✗	✗	✗	✗	✗	✗	✗	✗	✓
tag-score	✗	✗	✗	✗	✗	✗	✗	✗	✓
zero-1gram-percentual	✗	✗	✗	✗	✗	✗	✗	✗	✓
zero-2gram-percentual	✗	✗	✗	✗	✗	✗	✗	✗	✓
zip-compression-score	✗	✗	✗	✗	✗	✓	✗	✗	✓
complexidade kolmogorov	✓	✗	✗	✗	✗	✓	✗	✗	✗
histograma-1gram-vector	✓	✗	✗	✗	✓	✓	✓	✗	✗
histograma-2gram-vector	✗	✗	✗	✗	✗	✓	✗	✗	✗
longest-common-subsequence	✗	✓	✗	✗	✗	✗	✗	✗	✗
normalised-compression-distance	✗	✗	✓	✗	✗	✗	✗	✗	✗
byte-frequency-circular	✗	✗	✗	✗	✗	✗	✓	✗	✗
grayscale-vector	✗	✗	✗	✗	✗	✗	✗	✓	✗

Tabela A.2: Comparativo dos tipos de arquivo utilizados pelos autores.

Nome do tipo de arquivo	A1	A2	A3	A4	A5	A6	A7	A8	A9
AVI (Audio Video Interleave)	✓	✗	✗	✗	✗	✗	✗	✗	✓
BMP (<i>Bitmap picture</i>)	✓	✓	✓	✓	✗	✓	✗	✓	✓
CSV (<i>Comma-separated values</i>)	✗	✗	✓	✗	✗	✓	✓	✓	✗
DBX (<i>Outlook Express E-mail Folder</i>)	✓	✗	✗	✗	✗	✗	✗	✗	✗
DLL (<i>Dynamic-Link Library</i>)	✗	✗	✗	✗	✓	✗	✓	✗	✓
DOC (<i>Microsoft Word</i>)	✓	✗	✓	✗	✗	✓	✗	✓	✓
DOCX (<i>Microsoft Word XML Format</i>)	✗	✗	✓	✗	✗	✓	✗	✓	✓
DWG (<i>Programa Autocad</i>)	✗	✗	✗	✗	✗	✗	✗	✗	✓
EML (<i>mensagens de e-mails Base64</i>)	✗	✗	✗	✗	✗	✗	✗	✗	✓
EPS (<i>Encapsulated Postscript</i>)	✗	✗	✓	✗	✗	✗	✗	✓	✗
EXE (<i>Microsoft Windows Executable</i>)	✓	✗	✗	✓	✓	✗	✓	✗	✓
GIF (<i>Graphics interchange format</i>)	✓	✓	✓	✓	✗	✓	✗	✓	✗
GZ (<i>GNU zip compression</i>)	✗	✗	✓	✓	✗	✓	✗	✓	✗
HTML (<i>Hipertext markup language</i>)	✓	✗	✓	✗	✗	✓	✓	✓	✓
JAR (<i>Java archive</i>)	✗	✗	✓	✗	✗	✓	✗	✗	✓
JAVA (<i>Java source code</i>)	✗	✗	✓	✗	✗	✗	✗	✓	✗
JPEG (<i>Joint Photographic Experts Group</i>)	✓	✓	✓	✓	✓	✓	✓	✓	✓
JS (<i>Javascript source code</i>)	✗	✗	✓	✗	✗	✗	✗	✗	✗
LOG (<i>Logs file</i>)	✗	✗	✗	✗	✗	✗	✓	✓	✗
MP3 (<i>MPEG-1/2 Audio Layer 3</i>)	✗	✗	✗	✓	✓	✗	✓	✗	✓
MP4 (<i>MPEG-4 Part 14</i>)	✗	✗	✗	✗	✗	✗	✗	✗	✓
MPEG (<i>MPEG 1 System Stream</i>)	✗	✗	✗	✓	✗	✗	✗	✗	✗
PDF (<i>Portable document format</i>)	✓	✓	✓	✗	✓	✓	✓	✓	✓
PNG (<i>Portable network graphics</i>)	✗	✗	✓	✓	✗	✓	✗	✓	✗
PPS (<i>Microsoft Powerpoint Show</i>)	✗	✗	✓	✗	✗	✓	✗	✓	✗
PPT (<i>Microsoft Powerpoint</i>)	✓	✗	✓	✗	✗	✓	✗	✓	✗
PPTX (<i>Microsoft Powerpoint XML format</i>)	✗	✗	✓	✗	✗	✓	✗	✓	✗
PS (<i>PostScript</i>)	✗	✗	✓	✗	✗	✓	✗	✓	✗
PUB (<i>SSH public key files</i>)	✗	✗	✓	✗	✗	✗	✗	✓	✗
RTF (<i>Rich Text Format</i>)	✗	✗	✗	✗	✗	✓	✗	✓	✓
SQL (<i>SQL database scripts and dumps</i>)	✗	✗	✓	✗	✗	✓	✗	✓	✗
SQLITE (<i>SQLite database</i>)	✗	✗	✗	✗	✗	✗	✗	✗	✓
SWF (<i>Shockwave flash</i>)	✗	✗	✓	✗	✗	✓	✗	✓	✗
TIFF (<i>Tagged Image File Format</i>)	✗	✗	✗	✗	✗	✗	✗	✗	✓
TTF (<i>True type font</i>)	✗	✗	✓	✗	✗	✗	✗	✓	✗
TXT (<i>Text files DOS/Windows</i>)	✗	✗	✓	✓	✗	✓	✓	✓	✓
WMV (<i>Windows Media Player</i>)	✗	✗	✗	✗	✗	✗	✗	✗	✓
WP (<i>WordPerfect document</i>)	✗	✗	✗	✗	✗	✗	✗	✓	✗
XBM (<i>X Bitmap</i>)	✗	✗	✓	✗	✗	✗	✗	✓	✗
XLS (<i>Microsoft Excel</i>)	✗	✗	✓	✗	✗	✓	✗	✓	✗
XLSX (<i>Microsoft Excel XML format</i>)	✗	✗	✓	✗	✗	✓	✗	✓	✗
XML (<i>Extensible markup language</i>)	✗	✗	✓	✗	✗	✓	✓	✓	✓
ZIP (<i>Data compression and archiving</i>)	✓	✗	✓	✓	✗	✓	✗	✗	✓

B ATRIBUTOS DOS CLASSIFICADORES

Este apêndice apresenta os índices do *ranking* produzidos pelo algoritmo de classificação J48 sobre os atributos de 1, 2 e 4 *Kbytes*.

Tabela B.1: Resultado do *ranking* dos atributos para o algoritmo de classificação J48. Resultado ordenado por posição do atributo no *ranking*.

RANK	NOME DO ATRIBUTO		
	FRAG. 1 KBYTE	FRAG. 2 KBYTES	FRAG. 4 KBYTES
1	program-lang	ascii-percentual	entropy-2gram
2	ascii-percentual	histo-1gram-sd	histo-1gram-cornext-t2
3	histo-1gram-sd	byte-cornext1	byte-cornext1
4	monte-carlo-pi	npl-dist-mean	tag-score-3C
5	base85-percentual	histo-1gram-modes	base64-percentual
6	hamming-weight	tag-score-3E	lang-detect
7	byte-cornext1	monte-carlo-pi	histo-2gram-cornext-t2
8	tag-score-3E	histo-1gram-cornext-t2	monte-carlo-pi
9	byte-mean	program-lang	histo-1gram-sd
10	histo-2gram-cornext-t2	base85-percentual	base85-percentual
11	histo-1gram-cornext-t1	base64-percentual	histo-1gram-modes
12	entropy-2gram	hamming-weight	hamming-weight
13	npl-dist-mean	byte-mean	ascii-percentual
14	base64-percentual	entropy-2gram	tag-score-28
15	histo-1gram-modes	tag-score-28	npl-dist-mean
16	tag-score-28	histo-1gram-cornext-t1	tag-score-7D
17	tag-score-7B	tag-score-7B	tag-score-5D
18	lang-detect	tag-score-5B	byte-mean
19	pattern-f8-2gram	lang-detect	tag-score-7B
20	tag-score-5B	pattern-f8-2gram	histo-1gram-cornext-t1
21	tag-score-29	histo-3gram-cornext-t2	tag-score-3E
22	byte-cornext2	tag-score-29	pattern-f8-2gram
23	tag-score-5D	tag-score-7D	entropy-3gram
24	tag-score-7D	entropy-1gram	program-lang
25	histo-1gram-cornext-t2	tag-score-5D	tag-score-29
26	zip-compression-score	pattern-ff-2gram	tag-score-5B
27	pattern-ff-1gram	tag-score-3C	histo-3gram-cornext-t2
28	pattern-f8-1gram	histo-2gram-cornext-t2	pattern-ff-2gram
29	histo-2gram-modes	byte-cornext2	ascii-low-freq
30	ascii-low-freq	entropy-3gram	zero-1gram
31	tag-score-3C	pattern-f8-1gram	pattern-f8-1gram
32	zero-1gram	zero-1gram	byte-cornext2
33	histo-3gram-cornext-t2	ascii-low-freq	pattern-ff-1gram
34	entropy-3gram	pattern-ff-1gram	npl-long-seq
35	ascii-high-freq	histo-2gram-modes	entropy-1gram
36	pattern-ff-2gram	zero-2gram	histo-2gram-modes
37	zero-2gram	npl-long-seq	zero-2gram
38	npl-long-seq	ascii-freq	histo-3gram-modes
39	histo-3gram-modes	histo-3gram-modes	histo-3gram-sd
40	histo-2gram-sd	histo-2gram-sd	ascii-freq
41	ascii-freq	chi-square	histo-2gram-sd
42	chi-square	histo-3gram-sd	chi-square
43	histo-3gram-sd	serial-correlation	serial-correlation
44	serial-correlation	zip-compression-score	zip-compression-score
45	byte-sd	ascii-high-freq	byte-sd
46	entropy-1gram	byte-sd	ascii-high-freq

Tabela B.2: Resultado do *ranking* dos atributos para o algoritmo de classificação J48. Resultado ordenado por nome do atributo.

SELEÇÃO DE ATRIBUTOS			
NOME DO ATRIBUTO	FRAG. 1 KBYTES	FRAG. 2 KBYTES	FRAG. 4 KBYTES
ascii-percentual	0,4404560934	0,4413043743	0,4261134787
ascii-freq	0,3809702796	0,3903079349	0,3805870271
ascii-low-freq	0,4150475588	0,4064209843	0,4097304001
ascii-high-freq	0,4044696864	0,2983728051	0,0000000000
base64-percentual	0,4365616661	0,4381553351	0,4298716819
base85-percentual	0,4402741635	0,4384458341	0,4277043880
byte-mean	0,4387224466	0,4369631278	0,4217188883
byte-sd	0,3245627284	0,0000000000	0,2912604213
byte-cornext1	0,4396443850	0,4411057487	0,4303248928
byte-cornext2	0,4283245302	0,4157796903	0,4035460798
chi-square	0,3771273863	0,3741622368	0,3673691006
monte-carlo-pi	0,4403071764	0,4394807726	0,4288793420
serial-correlation	0,3632855424	0,3595822537	0,3603525198
pattern-ff-1gram	0,4217896039	0,4044239523	0,4019910536
pattern-ff-2gram	0,4024504223	0,4205935930	0,4098393360
pattern-f8-1gram	0,4192548614	0,4102851450	0,4049316842
pattern-f8-2gram	0,4331007680	0,4296856366	0,4178106089
hamming-weight	0,4402373925	0,4371029090	0,4267192171
histo-1gram-modes	0,4363962559	0,4402564130	0,4272292646
histo-1gram-sd	0,4403336917	0,4411533712	0,4278871985
histo-1gram-cornext-t1	0,4380768388	0,4343582332	0,4194602477
histo-1gram-cornext-t2	0,4239448721	0,4391633288	0,4305434670
histo-2gram-modes	0,4166066761	0,4010159331	0,3956524445
histo-2gram-sd	0,3837332267	0,3811189903	0,3743747687
histo-2gram-cornext-t2	0,4382032148	0,4159458512	0,4291935940
histo-3gram-modes	0,3895258066	0,3872909427	0,3867963236
histo-3gram-sd	0,3704267818	0,3670073158	0,3810621108
histo-3gram-cornext-t2	0,4086514589	0,4280033970	0,4116654343
lang-detect	0,4338616843	0,4302896903	0,4296309320
program-lang	0,4405001594	0,4390175230	0,4157120568
npl-dist-mean	0,4369050012	0,4404347411	0,4245476654
npl-long-seq	0,3947735293	0,3924211066	0,3989544645
entropy-1gram	0,0000000000	0,4240251409	0,3958397184
entropy-2gram	0,4371277456	0,4358025864	0,4306198391
entropy-3gram	0,4055693678	0,4132440463	0,4162643260
tag-score-28	0,4361042735	0,4344585328	0,4252576905
tag-score-29	0,4297095588	0,4262837085	0,4147667436
tag-score-3C	0,4140197942	0,4183120081	0,4301057766
tag-score-3E	0,4391817673	0,4401864431	0,4192950917
tag-score-5B	0,4313956221	0,4313812036	0,4133572118
tag-score-5D	0,4280185572	0,4226932633	0,4223924929
tag-score-7B	0,4347749227	0,4330609076	0,4208455678
tag-score-7D	0,4261454108	0,4245690019	0,4238559568
zero-1gram	0,4114835106	0,4075705670	0,4079949170
zero-2gram	0,3988081195	0,3969073226	0,3915840662
zip-compression-score	0,4219108215	0,3594493181	0,3470812315

C CLASSIFICADORES BINOMIAIS

Este apêndice apresenta os percentuais de classificação correta obtidos para os modelos de classificador binomial.

Tabela C.1: Percentuais de classificação correta do classificador binomial (resultados sem a aplicação dos filtros, resultado com a normalização dos atributos).

CLASSIFICADOR BINOMIAL (NÃO NORMALIZADO E NÃO DISCRETIZADO)				CLASSIFICADOR BINOMIAL (NORMALIZADO E NÃO DISCRETIZADO)			
TIPO	TAMANHO DO FRAGMENTO			TIPO	TAMANHO DO FRAGMENTO		
	1 KBYTE	2 KBYTES	4 KBYTES		1 KBYTE	2 KBYTES	4 KBYTES
AVI	98,73	98,89	99,12	AVI	98,70	98,85	99,12
BMP	99,72	99,83	99,88	BMP	99,72	99,80	99,86
DLL	99,21	99,20	98,87	DLL	99,22	99,19	98,86
DOC	97,59	97,33	97,21	DOC	97,60	97,31	97,21
DOCX	99,70	99,60	99,44	DOCX	99,68	99,60	99,45
DWG	99,79	99,56	99,60	DWG	99,82	99,56	99,61
EML	99,98	99,99	99,93	EML	99,98	99,99	99,93
EXE	96,15	97,49	97,92	EXE	96,16	97,45	97,88
HTM	99,90	99,94	99,95	HTM	99,90	99,94	99,95
JAR	97,72	98,39	98,69	JAR	97,67	98,34	98,61
JPG	96,68	96,83	97,39	JPG	96,65	96,85	97,36
MP3	99,15	99,59	99,76	MP3	99,17	99,59	99,75
MP4	96,68	97,75	98,11	MP4	96,63	97,74	98,06
PDF	96,33	97,62	98,40	PDF	96,32	97,62	98,40
RTF	99,78	99,85	99,92	RTF	99,79	99,85	99,92
SQLITE	99,66	99,69	99,67	SQLITE	99,65	99,68	99,70
TIF	99,42	99,72	99,56	TIF	99,41	99,71	99,56
TXT	99,55	99,69	99,84	TXT	99,56	99,68	99,83
WMV	98,55	98,89	98,17	WMV	98,54	98,85	98,17
XML	99,89	99,94	99,86	XML	99,89	99,94	99,86
ZIP	95,92	95,73	96,25	ZIP	95,93	95,73	96,27
media →	98,58	98,83	98,93	media →	98,57	98,82	98,92

Tabela C.2: Percentuais de classificação correta do classificador binomial (resultados com a discretização dos atributos, resultado com a redução dos atributos).

CLASSIFICADOR BINOMIAL (NORMALIZADO E DISCRETIZADO)				CLASSIFICADOR BINOMIAL (COM REDUÇÃO DE ATRIBUTOS)			
TIPO	TAMANHO DO FRAGMENTO			TIPO	TAMANHO DO FRAGMENTO		
	1 KBYTE	2 KBYTES	4 KBYTES		1 KBYTE	2 KBYTES	4 KBYTES
AVI	97,21	97,36	97,31	AVI	96,92	95,62	95,58
BMP	99,25	99,52	99,71	BMP	96,41	96,96	96,63
DLL	98,90	98,89	98,50	DLL	96,31	96,97	95,67
DOC	97,38	96,84	96,81	DOC	95,85	95,75	95,65
DOCX	95,45	95,59	95,69	DOCX	95,30	95,46	95,54
DWG	99,63	99,24	99,17	DWG	98,05	98,03	97,17
EML	97,87	98,68	99,29	EML	95,24	95,24	95,22
EXE	95,24	95,24	95,65	EXE	95,24	95,24	95,27
HTM	97,79	98,70	99,35	HTM	96,67	97,85	98,48
JAR	97,48	97,66	97,71	JAR	95,83	95,69	96,59
JPG	95,49	95,51	95,46	JPG	95,29	95,30	95,26
MP3	95,86	96,34	97,12	MP3	95,63	95,40	95,81
MP4	95,37	95,56	95,87	MP4	95,33	95,43	95,46
PDF	96,18	96,46	97,01	PDF	95,54	95,50	96,05
RTF	99,53	99,49	99,68	RTF	98,93	97,29	98,76
SQLITE	98,88	99,29	99,13	SQLITE	98,08	98,43	98,39
TIF	99,00	99,51	98,09	TIF	96,70	97,61	96,44
TXT	98,97	99,25	99,62	TXT	97,31	97,43	98,06
WMV	96,44	96,13	95,58	WMV	95,41	95,24	95,22
XML	99,57	99,67	99,69	XML	97,80	98,25	98,41
ZIP	95,35	95,38	95,49	ZIP	95,27	95,24	95,27
media →	97,47	97,63	97,71	media →	96,34	96,38	96,43

D CLASSIFICADORES MULTINOMIAIS

Este apêndice apresenta os percentuais de classificação correta obtidos para os modelos de classificadores multinomiais. Também são apresentadas as matrizes de confusão após cada aplicação dos filtros. Nessas matrizes são destacados os valores inferiores a 50.

Tabela D.1: Percentuais de classificação correta do classificador multinomial (resultados sem a aplicação dos filtros, resultado com a normalização dos atributos).

CLASSIFICADOR MULTINOMIAL (NÃO NORMALIZADO E NÃO DISCRETIZADO)				CLASSIFICADOR MULTINOMIAL (NORMALIZADO E NÃO DISCRETIZADO)			
TIPO	TAMANHO DO FRAGMENTO			TIPO	TAMANHO DO FRAGMENTO		
	1 KBYTE	2 KBYTES	4 KBYTES		1 KBYTE	2 KBYTES	4 KBYTES
AVI	84,70	87,70	90,50	AVI	84,30	87,40	90,60
BMP	96,80	98,20	98,50	BMP	96,80	98,20	98,50
DLL	90,80	90,70	86,70	DLL	90,80	90,90	86,90
DOC	96,40	95,00	93,50	DOC	96,30	95,00	93,50
DOCX	68,10	68,10	68,20	DOCX	68,30	68,10	67,60
DWG	97,70	95,10	94,90	DWG	97,70	95,10	95,50
EML	99,80	99,90	99,20	EML	99,80	99,90	99,20
EXE	56,70	72,10	77,00	EXE	56,60	71,90	76,60
HTM	98,90	99,20	99,60	HTM	99,00	99,20	99,50
JAR	69,40	80,10	83,90	JAR	69,40	79,70	83,50
JPG	63,80	66,80	72,40	JPG	63,60	66,50	71,90
MP3	89,90	94,90	97,20	MP3	89,60	94,80	97,00
MP4	59,00	74,50	79,90	MP4	58,80	74,60	79,90
PDF	58,30	73,10	81,90	PDF	58,50	72,80	81,90
RTF	97,90	98,50	99,10	RTF	97,90	98,50	99,10
SQLITE	96,40	96,40	96,10	SQLITE	96,30	96,30	96,40
TIF	93,10	96,80	95,10	TIF	93,00	96,80	95,00
TXT	93,70	95,20	97,80	TXT	93,80	95,20	97,80
WMV	83,90	87,00	80,50	WMV	83,40	87,00	80,40
XML	99,00	99,40	98,70	XML	99,00	99,40	98,70
ZIP	52,80	52,70	59,90	ZIP	52,80	52,50	59,40
media →	83,20	86,73	88,12	media →	83,13	86,66	88,04

Tabela D.2: Percentuais de classificação correta do classificador multinomial (resultados com a discretização dos atributos, resultado com a redução dos atributos).

CLASSIFICADOR MULTINOMIAL (NORMALIZADO E DISCRETIZADO)				CLASSIFICADOR MULTINOMIAL (COM REDUÇÃO DE ATRIBUTOS)			
TIPO	TAMANHO DO FRAGMENTO			TIPO	TAMANHO DO FRAGMENTO		
	1 KBYTE	2 KBYTES	4 KBYTES		1 KBYTE	2 KBYTES	4 KBYTES
AVI	49,90	60,20	53,80	AVI	55,90	50,10	51,00
BMP	93,30	96,60	97,80	BMP	93,50	96,10	97,10
DLL	89,30	90,60	86,10	DLL	85,50	86,50	83,60
DOC	41,00	54,40	58,20	DOC	38,20	59,20	62,20
DOCX	88,40	73,00	78,40	DOCX	85,00	80,90	79,30
DWG	95,90	93,70	93,20	DWG	93,00	91,10	91,50
EML	70,60	80,60	88,90	EML	69,20	79,20	88,60
EXE	26,40	30,50	36,20	EXE	23,60	26,60	33,70
HTM	90,90	95,00	97,50	HTM	83,60	89,50	96,50
JAR	83,50	83,60	81,40	JAR	69,00	79,30	74,20
JPG	29,20	31,60	32,80	JPG	27,30	29,50	31,00
MP3	45,10	50,60	61,90	MP3	41,80	49,10	58,90
MP4	36,30	43,40	50,20	MP4	28,70	37,10	46,50
PDF	69,70	69,90	78,40	PDF	62,70	65,50	76,10
RTF	96,10	95,70	97,00	RTF	86,70	92,80	95,80
SQLITE	93,80	94,90	94,20	SQLITE	89,50	91,60	91,50
TIF	92,50	96,80	82,10	TIF	92,70	96,20	78,60
TXT	85,90	89,60	95,40	TXT	82,90	86,20	90,20
WMV	47,20	44,30	39,90	WMV	43,80	43,50	37,90
XML	96,00	97,20	97,20	XML	88,90	95,80	96,50
ZIP	23,90	24,10	25,10	ZIP	21,90	23,20	23,30
media →	68,80	71,25	72,65	media →	64,92	69,00	70,67

Tabela D.3: Matriz de confusão do classificador multinomial para fragmentos de 1 *Kbyte* com dados não normalizados e não discretizados.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 1 KBYTE (NÃO NORMALIZADO E NÃO DISCRETIZADO)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQL	TIF	TXT	WMV	XML	ZIP
AVI	87,2	0,0	0,0	0,0	0,8	0,0	0,0	0,5	0,0	0,4	0,3	0,9	2,9	0,3	0,1	0,0	0,6	0,1	3,7	0,0	2,2
BMP	0,0	97,6	0,3	0,1	0,3	0,1	0,0	0,0	0,0	0,1	0,1	0,0	0,0	0,2	0,3	0,2	0,5	0,0	0,0	0,0	0,0
DLL	0,0	0,4	90,6	0,2	1,3	0,7	0,0	2,4	0,0	0,6	0,1	0,1	0,3	0,3	0,1	1,2	0,1	1,1	0,2	0,1	0,3
DOCX	0,0	0,1	0,1	97,0	0,5	0,0	0,0	0,0	0,0	0,1	0,3	0,0	0,0	0,9	0,0	0,7	0,0	0,0	0,0	0,0	0,3
DOC	0,8	0,8	1,1	0,6	69,2	0,2	0,0	2,5	0,0	2,1	5,6	0,4	2,4	5,4	0,3	0,4	0,4	2,7	1,1	0,0	3,9
DWG	0,0	0,1	0,5	0,1	0,2	98,1	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,1	0,0	0,2	0,0	0,2	0,0	0,0	0,1
EML	0,0	0,0	0,0	0,0	0,0	0,0	99,8	0,0	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
EXE	0,6	0,0	3,2	0,0	3,1	0,1	0,0	58,0	0,0	7,1	1,2	0,4	12,4	1,3	0,1	0,1	0,3	0,1	0,6	0,0	11,4
HTML	0,0	0,0	0,0	0,0	0,1	0,0	0,1	0,0	99,0	0,1	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,4
JAR	0,6	0,1	0,6	0,1	2,6	0,1	0,0	7,5	0,1	69,4	1,6	0,4	3,6	3,3	0,1	0,3	1,2	0,1	0,9	0,2	7,2
JPG	0,4	0,1	0,1	0,2	6,0	0,1	0,0	1,2	0,0	1,7	64,7	0,2	1,2	18,8	0,1	0,0	0,6	0,3	1,5	0,0	2,7
MP3	0,9	0,0	0,1	0,0	0,6	0,0	0,0	0,5	0,0	0,4	0,2	90,6	3,8	0,3	0,1	0,0	0,0	0,1	1,4	0,0	1,0
MP4	3,2	0,0	0,4	0,0	3,0	0,0	0,0	13,3	0,0	3,6	1,4	4,5	58,0	2,3	0,2	0,0	0,7	0,3	2,0	0,0	6,9
PDF	0,4	0,1	0,3	0,9	6,1	0,2	0,0	1,8	0,0	3,6	20,0	0,4	2,4	56,4	0,1	0,1	0,6	0,1	0,9	0,0	5,5
RTF	0,1	0,6	0,2	0,1	0,4	0,0	0,0	0,1	0,1	0,1	0,2	0,1	0,2	0,2	97,1	0,1	0,0	0,1	0,1	0,1	0,2
SQL	0,0	0,2	1,0	0,9	0,4	0,3	0,0	0,1	0,0	0,3	0,0	0,0	0,0	0,1	0,1	96,2	0,0	0,2	0,1	0,0	0,0
TIF	0,8	0,5	0,2	0,0	0,5	0,0	0,0	0,3	0,0	1,5	0,7	0,0	0,8	0,6	0,0	0,0	92,3	0,0	0,8	0,0	0,9
TXT	0,1	0,1	0,3	0,0	0,3	0,2	0,0	0,1	0,1	0,2	0,2	0,1	0,5	0,2	0,1	0,2	0,0	96,6	0,2	0,1	0,2
WMV	4,6	0,0	0,3	0,1	1,3	0,2	0,0	0,9	0,0	1,0	1,6	1,4	2,2	0,9	0,0	0,1	0,8	0,5	81,9	0,0	2,1
XML	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,5	0,2	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,2	0,0	98,9	0,0
ZIP	3,2	0,0	0,4	0,3	4,9	0,1	0,0	12,9	0,0	7,5	3,1	1,3	7,5	5,2	0,1	0,0	0,8	0,1	2,1	0,0	50,4

Tabela D.4: Matriz de confusão do classificador multinomial para fragmentos de 2 *Kbytes* com dados não normalizados e não discretizados.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 2 KBYTES (NÃO NORMALIZADO E NÃO DISCRETIZADO)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQL	TIF	TXT	WMV	XML	ZIP
AVI	88,8	0,0	0,2	0,1	0,5	0,3	0,0	0,2	0,0	0,4	0,1	0,6	3,5	0,4	0,0	0,0	0,5	0,2	2,8	0,0	1,6
BMP	0,0	98,2	0,3	0,1	0,2	0,1	0,0	0,0	0,0	0,1	0,0	0,0	0,1	0,1	0,1	0,1	0,3	0,0	0,0	0,0	0,1
DLL	0,2	0,4	89,6	0,2	1,2	0,8	0,0	1,8	0,0	0,9	0,1	0,0	0,4	0,4	0,1	1,4	0,1	1,5	0,2	0,0	0,8
DOCX	0,0	0,1	0,1	96,0	0,5	0,1	0,0	0,0	0,0	0,2	0,4	0,0	0,1	0,9	0,0	0,4	0,0	0,0	0,1	0,0	0,9
DOC	0,6	0,2	1,1	0,8	68,9	0,3	0,0	1,2	0,0	1,2	11,9	0,1	1,4	3,4	0,2	0,5	0,4	1,9	1,1	0,0	4,9
DWG	0,3	0,1	0,7	0,1	0,3	95,3	0,0	0,0	0,0	0,3	0,4	0,1	0,4	0,4	0,0	0,3	0,1	0,4	0,6	0,0	0,3
EML	0,0	0,0	0,0	0,0	0,0	99,9	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
EXE	0,2	0,0	1,9	0,0	1,3	0,1	0,0	78,3	0,0	2,8	0,1	0,1	4,2	0,5	0,1	0,1	0,1	0,1	0,4	0,0	10,0
HTML	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	99,4	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
JAR	0,4	0,1	0,8	0,3	1,5	0,3	0,0	3,9	0,1	79,3	0,5	0,1	1,5	2,3	0,1	0,3	0,4	0,0	0,6	0,1	7,3
JPG	0,1	0,0	0,1	0,5	12,6	0,3	0,0	0,1	0,0	0,5	68,0	0,1	0,4	10,9	0,0	0,0	0,2	0,1	0,4	0,0	5,6
MP3	0,7	0,0	0,0	0,0	0,2	0,1	0,0	0,1	0,0	0,1	0,1	95,0	2,3	0,1	0,0	0,0	0,0	0,1	0,7	0,0	0,4
MP4	3,7	0,0	0,4	0,1	1,6	0,4	0,0	7,1	0,0	1,5	0,4	2,4	72,6	1,1	0,1	0,0	0,2	0,3	3,2	0,0	4,7
PDF	0,5	0,2	0,5	1,0	3,8	0,5	0,0	1,1	0,0	2,7	12,0	0,2	1,3	70,5	0,1	0,1	0,3	0,0	0,4	0,0	4,7
RTF	0,0	0,2	0,2	0,1	0,3	0,1	0,0	0,1	0,0	0,1	0,1	0,0	0,3	0,1	97,9	0,1	0,0	0,0	0,1	0,0	0,4
SQL	0,0	0,2	1,3	0,6	0,4	0,3	0,0	0,0	0,1	0,3	0,0	0,0	0,0	0,1	0,1	96,4	0,0	0,1	0,1	0,0	0,0
TIF	0,5	0,2	0,1	0,0	0,5	0,1	0,0	0,1	0,0	0,5	0,2	0,0	0,2	0,3	0,0	0,0	96,5	0,0	0,3	0,0	0,4
TXT	0,0	0,0	0,2	0,0	0,1	0,2	0,0	0,0	0,1	0,1	0,1	0,3	0,1	0,1	0,1	0,0	98,1	0,1	0,1	0,2	0,2
WMV	3,1	0,0	0,2	0,1	1,2	0,6	0,0	0,5	0,0	0,7	0,4	0,7	3,5	0,5	0,0	0,1	0,3	0,2	86,0	0,0	1,7
XML	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,3	0,1	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,1	0,0	99,3	0,0
ZIP	2,1	0,1	0,9	1,1	6,0	0,4	0,0	13,9	0,0	7,2	6,8	0,5	5,1	4,4	0,2	0,1	0,4	0,1	1,7	0,0	49,1

Tabela D.5: Matriz de confusão do classificador multinomial para fragmentos de 4 *Kbytes* com dados não normalizados e não discretizados.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 4 KBYTES (NÃO NORMALIZADO E NÃO DISCRETIZADO)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQLL	TIF	TXT	WMV	XML	ZIP
AVI	91,5	0,0	0,2	0,1	0,2	0,2	0,0	0,2	0,0	0,1	0,0	0,7	2,7	0,2	0,0	0,0	0,4	0,1	2,8	0,0	0,6
BMP	0,0	98,7	0,2	0,1	0,1	0,1	0,0	0,1	0,0	0,1	0,0	0,0	0,1	0,1	0,0	0,1	0,1	0,1	0,0	0,0	0,0
DLL	0,2	0,3	86,1	0,3	1,7	1,2	0,0	3,5	0,0	0,8	0,3	0,1	0,7	0,6	0,1	1,4	0,1	0,8	0,7	0,1	1,1
DOCX	0,1	0,1	0,3	93,8	1,8	0,1	0,0	0,1	0,0	0,2	0,7	0,0	0,2	0,9	0,0	0,4	0,0	0,0	0,2	0,0	1,0
DOC	0,3	0,1	1,5	1,7	69,6	0,3	0,0	0,7	0,0	0,9	15,0	0,1	0,7	2,3	0,1	0,5	0,3	0,1	1,0	0,0	4,8
DWG	0,2	0,1	0,9	0,2	0,3	95,6	0,0	0,2	0,0	0,1	0,1	0,0	0,2	0,3	0,0	0,4	0,1	0,2	0,8	0,0	0,2
EML	0,0	0,0	0,0	0,0	0,0	0,0	99,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,0	0,6	0,0
EXE	0,2	0,1	4,1	0,1	0,8	0,3	0,0	79,9	0,0	1,8	0,0	0,1	2,6	0,5	0,1	0,3	0,1	0,4	0,3	0,0	8,3
HTML	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	99,6	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,2	0,0
JAR	0,2	0,1	0,8	0,3	0,9	0,2	0,0	2,5	0,0	83,1	0,2	0,0	0,9	1,9	0,0	0,2	0,7	0,0	1,0	0,1	7,0
JPG	0,1	0,0	0,3	0,7	14,6	0,1	0,0	0,0	0,0	0,2	73,6	0,0	0,1	5,1	0,0	0,1	0,2	0,1	0,5	0,0	4,1
MP3	0,7	0,0	0,1	0,0	0,1	0,1	0,0	0,1	0,0	0,0	0,0	97,1	0,9	0,1	0,0	0,0	0,0	0,1	0,6	0,0	0,1
MP4	2,8	0,0	0,8	0,2	0,8	0,3	0,0	3,8	0,0	0,7	0,1	0,9	79,3	0,7	0,0	0,0	0,1	0,1	6,3	0,0	3,0
PDF	0,2	0,1	0,7	1,0	2,9	0,4	0,0	1,1	0,0	2,2	5,8	0,1	0,7	79,7	0,0	0,1	0,4	0,0	0,8	0,0	3,5
RTF	0,0	0,1	0,1	0,1	0,2	0,0	0,0	0,1	0,0	0,1	0,0	0,0	0,1	0,1	98,8	0,0	0,0	0,0	0,1	0,0	0,2
SQLL	0,0	0,1	0,9	0,5	0,5	0,5	0,0	0,2	0,0	0,1	0,0	0,0	0,1	0,1	0,0	96,6	0,0	0,0	0,2	0,0	0,1
TIF	0,3	0,1	0,2	0,0	0,4	0,1	0,0	0,1	0,0	0,6	0,2	0,0	0,2	0,4	0,0	0,0	95,3	0,0	1,1	0,0	0,9
TXT	0,0	0,0	0,2	0,0	0,1	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,2	0,0	0,0	0,1	0,0	98,7	0,1	0,1	0,1
WMV	3,3	0,0	0,7	0,2	1,2	0,8	0,0	0,4	0,0	1,1	0,4	0,5	6,5	0,8	0,0	0,2	1,4	0,1	79,4	0,0	3,0
XML	0,0	0,0	0,1	0,0	0,0	0,0	0,7	0,0	0,2	0,1	0,0	0,0	0,0	0,0	0,0	0,1	0,0	98,7	0,0	0,0	0,0
ZIP	0,9	0,1	1,2	1,1	5,5	0,3	0,0	11,0	0,0	6,9	5,0	0,1	3,3	3,6	0,1	0,1	1,0	0,0	2,9	0,0	56,7

Tabela D.6: Matriz de confusão do classificador multinomial para fragmentos de 1 *Kbyte* com dados normalizados e não discretizados.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 1 KBYTE (NORMALIZADO E NÃO DISCRETIZADO)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQLL	TIF	TXT	WMV	XML	ZIP
AVI	86,6	0,0	0,1	0,0	0,7	0,0	0,0	0,5	0,0	0,6	0,4	1,0	2,8	0,3	0,0	0,0	0,8	0,1	4,0	0,0	2,2
BMP	0,0	97,5	0,4	0,1	0,4	0,1	0,0	0,0	0,0	0,1	0,1	0,0	0,0	0,2	0,3	0,2	0,6	0,0	0,0	0,0	0,1
DLL	0,1	0,4	90,6	0,2	1,3	0,6	0,0	2,5	0,0	0,5	0,1	0,1	0,3	0,2	0,1	1,2	0,1	1,1	0,2	0,1	0,3
DOCX	0,0	0,1	0,2	96,8	0,5	0,1	0,0	0,0	0,0	0,1	0,2	0,0	0,0	0,9	0,0	0,8	0,0	0,0	0,1	0,0	0,3
DOC	0,8	0,9	1,1	0,6	69,1	0,2	0,0	2,5	0,0	2,1	5,7	0,4	2,5	5,5	0,3	0,4	0,4	2,6	1,1	0,0	3,8
DWG	0,0	0,1	0,5	0,0	0,2	98,2	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,1	0,0	0,2	0,0	0,2	0,2	0,0	0,1
EML	0,0	0,0	0,0	0,0	0,0	0,0	99,8	0,0	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
EXE	0,5	0,0	3,2	0,0	3,1	0,1	0,0	58,3	0,0	6,9	1,1	0,5	12,3	1,2	0,1	0,1	0,2	0,1	0,6	0,0	11,6
HTML	0,0	0,0	0,0	0,1	0,0	0,1	0,0	99,0	0,1	0,1	0,0	0,0	0,0	0,1	0,0	0,0	0,1	0,0	0,4	0,0	0,0
JAR	0,7	0,1	0,6	0,1	2,5	0,1	0,0	7,4	0,1	69,1	1,6	0,4	3,9	3,0	0,1	0,3	1,2	0,1	0,8	0,2	7,6
JPG	0,4	0,1	0,1	0,3	6,0	0,1	0,0	1,2	0,0	1,5	64,7	0,2	1,3	18,8	0,1	0,0	0,6	0,3	1,7	0,0	2,5
MP3	0,9	0,0	0,1	0,0	0,4	0,0	0,0	0,5	0,0	0,4	0,3	90,8	3,5	0,3	0,1	0,0	0,0	0,1	1,5	0,0	0,9
MP4	3,2	0,0	0,4	0,1	2,9	0,0	0,0	13,6	0,0	3,8	1,4	4,3	57,8	2,4	0,2	0,0	0,6	0,3	2,0	0,0	6,8
PDF	0,4	0,1	0,2	0,9	6,2	0,2	0,0	1,9	0,0	3,4	19,9	0,4	2,7	56,3	0,1	0,1	0,6	0,1	0,8	0,0	5,5
RTF	0,1	0,6	0,2	0,1	0,5	0,0	0,0	0,1	0,1	0,1	0,2	0,1	0,2	0,2	97,1	0,1	0,0	0,1	0,1	0,1	0,3
SQLL	0,0	0,2	1,1	1,0	0,4	0,3	0,0	0,1	0,0	0,2	0,0	0,0	0,0	0,1	0,1	96,1	0,0	0,2	0,1	0,0	0,0
TIF	0,8	0,4	0,2	0,0	0,6	0,0	0,0	0,4	0,0	1,6	0,7	0,0	0,7	0,6	0,0	0,0	92,2	0,0	0,8	0,0	0,9
TXT	0,1	0,1	0,3	0,0	0,3	0,1	0,0	0,1	0,1	0,2	0,2	0,2	0,5	0,1	0,1	0,2	0,0	96,6	0,3	0,1	0,2
WMV	4,8	0,0	0,2	0,1	1,2	0,2	0,0	0,9	0,0	1,1	1,7	1,5	2,3	1,0	0,1	0,1	0,9	0,5	81,5	0,0	2,0
XML	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,4	0,1	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,2	0,0	0,0	99,0	0,0
ZIP	3,2	0,0	0,4	0,3	4,8	0,1	0,0	13,1	0,0	7,5	3,4	1,3	7,2	5,1	0,2	0,0	0,8	0,2	2,0	0,0	50,5

Tabela D.7: Matriz de confusão do classificador multinomial para fragmentos de 2 *Kbytes* com dados normalizados e não discretizados.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 2 KBYTES (NORMALIZADO E NÃO DISCRETIZADO)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQL	TIF	TXT	WMV	XML	ZIP
AVI	88,4	0,0	0,2	0,1	0,6	0,3	0,0	0,2	0,0	0,4	0,2	0,6	3,4	0,4	0,0	0,0	0,5	0,2	2,9	0,0	1,6
BMP	0,0	98,3	0,2	0,1	0,2	0,1	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,1	0,1	0,1	0,3	0,0	0,0	0,0	0,1
DLL	0,2	0,3	89,7	0,2	1,1	0,8	0,0	1,8	0,0	0,8	0,1	0,0	0,4	0,4	0,1	1,4	0,1	1,5	0,2	0,0	0,8
DOCX	0,0	0,1	0,1	95,9	0,5	0,1	0,0	0,0	0,0	0,2	0,5	0,0	0,1	0,9	0,0	0,4	0,0	0,0	0,1	0,0	1,0
DOC	0,5	0,2	1,1	0,8	68,6	0,3	0,0	1,4	0,0	1,3	12,3	0,1	1,4	3,3	0,2	0,5	0,4	1,9	1,0	0,0	4,8
DWG	0,3	0,1	0,7	0,1	0,4	95,2	0,0	0,1	0,0	0,3	0,3	0,1	0,3	0,4	0,0	0,3	0,1	0,4	0,6	0,0	0,3
EML	0,0	0,0	0,0	0,0	0,0	0,0	99,9	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
EXE	0,2	0,0	1,8	0,0	1,2	0,1	0,0	78,7	0,0	2,6	0,1	0,1	4,1	0,6	0,1	0,1	0,0	0,1	0,3	0,0	9,8
HTML	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	99,4	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
JAR	0,5	0,1	0,9	0,2	1,5	0,4	0,0	3,9	0,1	78,6	0,5	0,1	1,6	2,3	0,1	0,3	0,5	0,0	0,6	0,1	7,7
JPG	0,2	0,0	0,1	0,6	12,5	0,3	0,0	0,1	0,0	0,5	68,0	0,1	0,4	10,8	0,0	0,0	0,2	0,0	0,4	0,0	5,8
MP3	0,6	0,0	0,0	0,0	0,1	0,1	0,0	0,1	0,0	0,1	0,1	94,9	2,4	0,2	0,0	0,0	0,0	0,0	0,8	0,0	0,4
MP4	3,6	0,0	0,5	0,1	1,6	0,4	0,0	7,3	0,0	1,6	0,5	2,5	72,2	1,2	0,1	0,0	0,2	0,2	3,3	0,0	4,5
PDF	0,6	0,2	0,5	1,0	3,8	0,5	0,0	1,0	0,0	2,7	12,1	0,3	1,3	70,5	0,1	0,1	0,3	0,0	0,4	0,0	4,7
RTF	0,1	0,2	0,2	0,1	0,3	0,1	0,0	0,1	0,0	0,1	0,1	0,0	0,2	0,1	97,8	0,1	0,0	0,0	0,1	0,0	0,4
SQL	0,0	0,2	1,2	0,6	0,4	0,3	0,0	0,0	0,1	0,3	0,0	0,0	0,0	0,1	0,1	96,5	0,0	0,1	0,1	0,0	0,0
TIF	0,6	0,2	0,1	0,0	0,5	0,1	0,0	0,1	0,0	0,6	0,3	0,0	0,2	0,3	0,0	0,0	96,5	0,0	0,3	0,0	0,4
TXT	0,0	0,0	0,2	0,0	0,2	0,2	0,0	0,0	0,1	0,1	0,1	0,1	0,3	0,1	0,1	0,0	98,1	0,1	0,1	0,2	0,2
WMV	3,2	0,0	0,2	0,1	1,2	0,5	0,0	0,6	0,0	0,7	0,4	0,7	3,5	0,4	0,1	0,1	0,3	0,2	86,1	0,0	1,6
XML	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,3	0,1	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,1	0,0	99,3	0,0
ZIP	2,2	0,1	0,9	1,1	6,1	0,4	0,0	14,0	0,0	7,3	6,6	0,4	5,1	4,6	0,2	0,1	0,4	0,1	1,6	0,0	48,8

Tabela D.8: Matriz de confusão do classificador multinomial para fragmentos de 4 *Kbytes* com dados normalizados e não discretizados.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 4 KBYTES (NORMALIZADO E NÃO DISCRETIZADO)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQL	TIF	TXT	WMV	XML	ZIP
AVI	91,5	0,0	0,2	0,1	0,3	0,2	0,0	0,2	0,0	0,2	0,0	0,7	2,6	0,2	0,0	0,0	0,4	0,1	2,7	0,0	0,6
BMP	0,0	98,7	0,2	0,1	0,2	0,1	0,0	0,0	0,0	0,1	0,0	0,0	0,1	0,1	0,1	0,1	0,1	0,0	0,1	0,0	0,1
DLL	0,3	0,3	86,2	0,3	1,7	1,1	0,0	3,5	0,0	0,7	0,3	0,1	0,7	0,6	0,1	1,2	0,1	0,9	0,7	0,1	1,1
DOCX	0,1	0,1	0,2	93,9	1,8	0,1	0,0	0,1	0,0	0,2	0,7	0,0	0,2	0,9	0,0	0,4	0,0	0,0	0,2	0,0	1,0
DOC	0,3	0,1	1,5	1,6	69,0	0,3	0,0	0,7	0,0	0,9	15,4	0,1	0,7	2,4	0,2	0,5	0,3	0,1	1,1	0,0	4,9
DWG	0,2	0,1	0,8	0,2	0,3	96,0	0,0	0,2	0,0	0,1	0,1	0,1	0,2	0,3	0,0	0,4	0,1	0,2	0,6	0,0	0,2
EML	0,0	0,0	0,0	0,0	0,0	0,0	99,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,6	0,0	0,0
EXE	0,3	0,0	4,0	0,1	0,9	0,3	0,0	79,7	0,0	1,8	0,0	0,1	2,6	0,5	0,0	0,3	0,1	0,4	0,3	0,0	8,5
HTML	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,0	99,5	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,3	0,0	0,0
JAR	0,2	0,1	0,7	0,2	1,1	0,2	0,0	2,6	0,0	82,3	0,2	0,0	0,9	1,8	0,0	0,2	0,8	0,0	1,1	0,1	7,3
JPG	0,0	0,0	0,4	0,7	14,6	0,1	0,0	0,1	0,0	0,1	73,7	0,0	0,1	5,1	0,0	0,0	0,3	0,1	0,5	0,0	4,0
MP3	0,6	0,0	0,0	0,0	0,1	0,0	0,0	0,1	0,0	0,1	0,1	97,1	0,8	0,1	0,0	0,0	0,0	0,1	0,6	0,0	0,1
MP4	2,7	0,0	0,7	0,1	0,8	0,2	0,0	3,8	0,0	0,8	0,1	1,0	79,4	0,6	0,0	0,0	0,2	0,1	6,3	0,0	3,1
PDF	0,2	0,1	0,8	1,0	2,9	0,3	0,0	1,1	0,0	2,1	6,0	0,1	0,7	79,6	0,0	0,1	0,4	0,0	0,8	0,0	3,7
RTF	0,0	0,1	0,1	0,0	0,2	0,0	0,0	0,1	0,0	0,1	0,0	0,0	0,1	0,1	98,8	0,0	0,0	0,0	0,1	0,0	0,2
SQL	0,0	0,1	0,9	0,6	0,5	0,3	0,0	0,2	0,0	0,1	0,1	0,0	0,0	0,1	0,0	96,7	0,0	0,1	0,2	0,0	0,1
TIF	0,4	0,1	0,1	0,0	0,5	0,1	0,0	0,1	0,0	0,7	0,3	0,0	0,2	0,4	0,0	0,0	95,3	0,0	1,0	0,0	0,9
TXT	0,0	0,0	0,2	0,0	0,2	0,1	0,0	0,0	0,1	0,0	0,0	0,0	0,1	0,0	0,0	0,1	0,0	98,7	0,1	0,0	0,0
WMV	3,3	0,1	0,7	0,2	1,3	0,7	0,0	0,4	0,0	1,3	0,5	0,6	6,6	0,9	0,0	0,1	1,3	0,1	78,8	0,0	3,1
XML	0,0	0,0	0,1	0,0	0,0	0,0	0,7	0,0	0,2	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,0	98,7	0,0
ZIP	0,8	0,1	1,2	1,2	5,7	0,3	0,0	11,1	0,0	6,8	4,9	0,2	3,3	3,5	0,1	0,1	1,0	0,0	2,9	0,0	56,7

Tabela D.9: Matriz de confusão do classificador multinomial para fragmentos de 1 *Kbyte* com dados normalizados e discretizados.

		CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 1 KBYTE (NORMALIZADO E DISCRETIZADO)																				
classif. como →		AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQL	TIF	TXT	WMV	XML	ZIP
AVI	77,4	0,0	0,0	0,2	0,0	0,0	0,0	4,6	0,0	0,1	0,7	5,0	2,1	0,2	0,0	0,0	0,2	0,0	9,0	0,0	0,4	
BMP	0,0	92,5	0,4	0,2	0,6	0,2	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,2	0,5	0,4	4,7	0,0	0,1	0,0	0,1	
DLL	0,2	0,9	88,1	0,3	1,5	1,3	0,0	1,3	0,1	0,6	0,3	0,3	0,2	0,4	0,4	1,7	0,1	1,1	0,5	0,2	0,4	
DOCX	5,9	0,3	0,3	15,2	0,6	0,2	0,0	20,4	0,0	1,1	20,3	5,5	6,0	1,7	0,0	0,1	0,1	0,1	5,0	0,0	17,4	
DOC	2,7	1,1	1,5	2,5	51,1	0,3	0,0	7,0	0,1	0,4	10,3	3,3	3,6	1,2	0,5	0,5	0,1	2,6	2,5	0,1	8,4	
DWG	0,1	0,1	1,1	0,1	0,2	96,7	0,0	0,1	0,0	0,0	0,0	0,0	0,0	0,2	0,1	0,4	0,0	0,2	0,5	0,0	0,1	
EML	0,0	0,0	0,0	0,0	0,0	0,0	94,9	0,0	3,9	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,0	0,9	0,0	
EXE	7,5	0,0	3,4	2,2	0,3	0,2	0,0	46,8	0,0	0,8	13,2	6,0	2,8	0,2	0,0	0,1	0,2	0,1	4,4	0,0	11,7	
HTML	0,0	0,0	0,1	0,0	0,2	0,0	37,8	0,0	59,6	0,2	0,0	0,0	0,0	0,0	0,6	0,1	0,0	0,1	0,0	1,1	0,0	
JAR	2,1	0,1	0,6	1,9	0,3	0,1	0,0	14,3	0,1	54,0	7,9	4,1	1,4	0,6	0,2	0,3	0,3	0,1	4,5	0,5	6,6	
JPG	5,2	0,0	0,1	3,0	0,4	0,1	0,0	14,0	0,0	0,8	43,2	5,2	5,5	4,9	0,0	0,0	0,2	0,3	7,6	0,0	9,4	
MP3	12,5	0,0	0,0	0,8	0,2	0,0	0,0	8,0	0,0	1,0	1,0	50,4	5,9	0,9	0,0	0,0	0,1	0,1	13,9	0,0	5,1	
MP4	12,8	0,0	0,1	2,1	0,2	0,0	0,0	21,1	0,0	0,4	8,6	8,6	24,2	0,2	0,0	0,0	0,1	0,0	4,0	0,0	17,4	
PDF	2,6	0,1	0,3	3,6	0,5	0,5	0,0	9,8	0,0	0,8	22,3	6,1	4,5	30,7	0,0	0,1	0,1	0,1	6,3	0,0	11,4	
RTF	0,3	0,8	0,3	0,1	0,5	0,1	0,1	0,2	0,4	0,2	0,2	0,1	0,2	0,1	94,9	0,2	0,0	0,5	0,1	0,4	0,3	
SQL	0,7	0,4	1,5	0,1	0,5	0,4	0,0	0,3	0,2	0,3	0,5	0,4	0,7	0,2	0,2	82,6	0,0	9,2	0,5	0,2	0,9	
TIF	2,3	2,5	0,1	0,4	0,1	0,0	0,0	3,2	0,0	0,5	0,7	1,2	0,8	0,1	0,0	0,0	84,7	0,0	2,0	0,0	1,5	
TXT	0,4	0,1	0,4	0,1	0,1	0,2	0,1	0,5	0,2	0,1	0,2	0,3	0,1	0,1	0,5	1,0	0,0	94,2	0,4	0,3	0,4	
WMV	12,0	0,0	0,1	0,6	0,2	0,2	0,0	6,9	0,0	1,4	2,0	9,6	2,1	1,4	0,0	0,1	0,4	0,4	61,1	0,0	1,6	
XML	0,0	0,0	0,2	0,0	0,1	0,0	1,5	0,0	0,9	0,5	0,0	0,0	0,0	0,0	0,7	0,3	0,0	0,5	0,0	95,4	0,0	
ZIP	10,3	0,0	0,1	3,7	0,2	0,3	0,0	18,2	0,0	1,1	16,5	5,6	6,5	0,8	0,0	0,0	0,2	0,0	7,0	0,0	29,2	

Tabela D.10: Matriz de confusão do classificador multinomial para fragmentos de 2 *Kbytes* com dados normalizados e discretizados.

		CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 2 KBYTES (NORMALIZADO E DISCRETIZADO)																				
classif. como →		AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQL	TIF	TXT	WMV	XML	ZIP
AVI	68,9	0,1	0,1	0,2	0,2	0,1	0,0	9,3	0,0	0,3	0,8	6,6	2,8	1,4	0,0	0,0	0,0	0,1	7,9	0,0	1,1	
BMP	0,1	95,2	0,3	0,1	0,3	0,2	0,0	0,0	0,0	0,2	0,0	0,1	0,0	0,1	0,4	0,2	2,5	0,0	0,1	0,1	0,1	
DLL	0,3	0,7	86,8	0,2	1,6	1,3	0,0	1,5	0,1	0,6	0,6	0,2	0,3	0,3	0,6	1,9	0,1	1,6	0,4	0,2	0,7	
DOCX	3,9	0,2	0,2	14,2	2,2	0,3	0,0	21,5	0,0	0,9	22,2	6,2	5,1	1,7	0,0	0,0	0,0	0,0	6,4	0,0	14,8	
DOC	3,4	0,3	1,5	1,4	42,8	0,2	0,0	14,3	0,0	0,7	12,1	3,4	2,4	0,8	0,3	0,6	0,1	1,8	7,4	0,0	6,4	
DWG	0,4	0,2	1,2	0,2	0,3	90,8	0,0	0,2	0,0	0,5	0,5	1,1	0,5	1,1	0,1	0,4	0,0	0,4	1,9	0,0	0,3	
EML	0,0	0,0	0,0	0,0	0,0	0,0	95,7	0,0	3,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,0	1,2	0,0		
EXE	4,1	0,0	1,8	1,1	1,6	0,1	0,0	59,1	0,0	0,4	12,9	3,3	1,1	0,1	0,0	0,1	0,0	0,1	2,4	0,0	11,8	
HTML	0,0	0,0	0,1	0,0	0,1	0,0	21,2	0,0	77,5	0,1	0,0	0,0	0,0	0,0	0,2	0,1	0,0	0,1	0,0	0,6	0,0	
JAR	0,9	0,2	0,6	1,5	1,2	0,4	0,0	8,7	0,1	60,7	8,8	3,4	1,7	0,8	0,2	0,3	0,1	0,0	4,0	0,2	6,2	
JPG	2,2	0,0	0,0	1,9	1,4	0,2	0,0	12,3	0,0	0,8	47,8	5,1	5,2	3,4	0,0	0,1	0,0	0,0	6,5	0,0	13,0	
MP3	3,0	0,0	0,0	0,3	0,3	0,1	0,0	5,2	0,0	1,5	0,4	62,6	3,2	1,8	0,0	0,0	0,0	0,0	19,5	0,0	1,9	
MP4	9,1	0,0	0,2	0,7	0,6	0,1	0,0	20,7	0,0	0,4	6,0	7,9	29,7	1,9	0,0	0,0	0,0	0,0	4,9	0,0	17,7	
PDF	2,1	0,1	0,4	1,6	0,8	1,1	0,1	7,2	0,0	1,8	15,4	9,6	6,7	38,6	0,0	0,2	0,1	0,0	7,8	0,1	6,4	
RTF	0,2	0,4	0,3	0,0	0,3	0,1	0,1	0,3	0,1	0,2	0,1	0,1	0,1	0,1	94,0	0,2	0,0	2,8	0,2	0,2	0,3	
SQL	0,1	0,2	1,6	0,1	0,6	0,4	0,1	0,1	0,1	0,3	0,4	0,3	0,5	0,2	0,1	90,3	0,0	3,6	0,3	0,1	0,4	
TIF	1,2	0,8	0,0	0,0	0,5	0,0	0,0	2,2	0,0	0,1	0,4	0,3	0,2	0,0	0,0	0,0	92,4	0,0	1,1	0,0	0,7	
TXT	0,1	0,0	0,3	0,0	0,1	0,3	0,0	0,2	0,1	0,1	0,1	0,2	0,1	0,0	1,8	0,4	0,0	95,4	0,3	0,1	0,3	
WMV	8,0	0,0	0,1	0,3	1,9	0,8	0,0	7,3	0,0	1,2	2,8	9,2	3,0	1,7	0,0	0,1	0,1	0,2	60,8	0,0	2,5	
XML	0,0	0,0	0,2	0,0	0,0	0,0	1,6	0,0	0,5	0,2	0,0	0,0	0,0	0,0	0,3	0,1	0,0	0,2	0,0	97,0	0,0	
ZIP	6,4	0,1	0,2	2,3	1,8	0,4	0,0	23,5	0,0	1,6	19,9	4,2	5,8	1,0	0,0	0,1	0,0	0,0	5,7	0,0	26,9	

Tabela D.11: Matriz de confusão do classificador multinomial para fragmentos de 4 *Kbytes* com dados normalizados e discretizados.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 4 KBYTES (NORMALIZADO E DISCRETIZADO)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQL	TIF	TXT	WMV	XML	ZIP
AVI	73,9	0,0	0,2	0,2	0,1	0,1	0,0	6,4	0,0	0,2	0,6	5,2	3,9	0,6	0,0	0,0	0,9	0,1	6,2	0,0	1,4
BMP	0,0	97,5	0,2	0,2	0,1	0,1	0,0	0,0	0,0	0,1	0,0	0,1	0,0	0,1	0,1	0,2	1,1	0,0	0,0	0,0	0,0
DLL	0,5	0,5	81,1	0,5	2,1	1,7	0,0	3,3	0,0	0,7	0,8	0,7	0,6	0,7	0,4	2,0	0,1	0,9	0,8	0,4	2,1
DOCX	6,3	0,1	0,2	18,7	2,3	0,1	0,0	13,3	0,0	1,7	13,0	4,0	3,2	1,4	0,0	0,2	1,2	0,0	5,2	0,0	29,0
DOC	3,7	0,2	1,8	2,8	42,3	0,4	0,0	7,2	0,0	0,9	11,7	2,0	1,9	0,8	0,2	0,8	1,2	0,1	4,6	0,1	17,5
DWG	0,4	0,1	1,5	0,2	0,4	89,8	0,0	0,3	0,0	0,6	0,3	1,8	0,3	0,7	0,0	0,6	0,1	0,2	2,3	0,0	0,4
EML	0,0	0,0	0,0	0,0	0,0	0,0	97,6	0,0	1,4	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,0	0,8	0,0
EXE	3,3	0,0	4,8	0,5	0,6	0,5	0,0	55,0	0,0	1,0	6,7	2,2	1,0	0,8	0,1	0,3	1,5	0,4	1,0	0,1	20,2
HTML	0,0	0,0	0,1	0,0	0,0	0,0	10,9	0,0	88,2	0,0	0,0	0,0	0,0	0,0	0,1	0,1	0,0	0,0	0,0	0,5	0,0
JAR	1,1	0,1	0,5	0,8	0,4	0,4	0,0	7,6	0,0	63,4	6,2	1,4	1,1	0,9	0,1	0,2	1,1	0,0	2,4	0,2	12,0
JPG	4,5	0,0	0,1	2,5	1,6	0,1	0,0	10,2	0,0	1,7	36,1	2,8	3,7	1,8	0,0	0,1	2,4	0,1	7,1	0,0	25,1
MP3	4,0	0,0	0,1	0,4	0,2	0,5	0,0	1,7	0,0	1,1	0,2	73,2	1,4	0,9	0,0	0,1	0,9	0,1	14,2	0,0	1,1
MP4	12,9	0,0	0,2	0,5	0,3	0,1	0,0	13,6	0,0	0,7	2,9	4,8	36,3	1,9	0,0	0,1	0,5	0,0	4,5	0,0	20,8
PDF	3,0	0,1	0,5	1,6	0,6	1,0	0,1	4,8	0,0	1,3	7,2	7,8	7,2	48,6	0,0	0,2	0,9	0,0	4,8	0,0	10,4
RTF	0,1	0,1	0,2	0,0	0,3	0,1	0,0	0,1	0,0	0,1	0,0	0,0	0,1	0,0	96,9	0,1	0,0	1,2	0,1	0,1	0,2
SQL	0,3	0,2	1,6	0,4	0,9	0,5	0,1	0,6	0,1	0,3	0,9	0,4	0,4	0,3	0,1	89,7	0,1	1,4	0,6	0,2	0,7
TIF	6,7	0,5	0,0	0,1	0,1	0,1	0,0	6,8	0,0	0,7	7,7	0,6	0,3	0,1	0,0	0,0	67,3	0,0	6,9	0,0	2,0
TXT	0,1	0,0	0,2	0,0	0,1	0,2	0,0	0,1	0,0	0,1	0,0	0,1	0,0	0,0	1,5	0,2	0,0	96,8	0,2	0,1	0,2
WMV	11,9	0,0	0,2	0,5	0,3	0,5	0,0	5,5	0,0	1,2	5,2	9,8	6,6	1,3	0,0	0,2	1,5	0,0	42,5	0,0	12,8
XML	0,0	0,0	0,3	0,0	0,1	0,0	1,6	0,1	0,4	0,1	0,0	0,0	0,0	0,0	0,2	0,2	0,0	0,1	0,0	96,9	0,0
ZIP	4,8	0,1	0,3	2,2	1,0	0,3	0,0	15,5	0,0	1,9	10,5	1,4	4,4	1,0	0,0	0,1	1,1	0,0	3,0	0,0	52,3

Tabela D.12: Matriz de confusão do classificador multinomial para fragmentos de 1 *Kbyte* com dados normalizados, discretizados e com redução de atributos.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 1 KBYTE (NORMALIZADO, DISCRETIZADO E COM REDUÇÃO DE ATRIBUTOS)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQL	TIF	TXT	WMV	XML	ZIP
AVI	71,0	0,0	0,0	0,9	0,0	0,0	0,0	5,1	0,0	0,1	1,2	5,0	6,7	0,2	0,0	0,0	0,2	0,0	9,0	0,0	0,7
BMP	0,0	89,8	0,4	0,2	1,1	0,2	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,1	1,8	0,3	5,6	0,1	0,1	0,0	0,1
DLL	0,2	1,0	83,9	0,3	1,8	1,8	0,0	0,9	0,4	1,0	0,5	0,3	0,3	1,1	0,8	2,7	0,1	1,2	0,6	0,7	0,5
DOCX	3,7	0,2	0,5	12,8	0,6	0,5	0,0	18,0	0,0	1,6	21,4	6,1	8,9	1,9	0,1	0,1	0,0	0,1	5,0	0,0	18,5
DOC	1,7	0,6	1,9	2,3	48,9	0,4	0,0	6,1	0,1	0,9	10,4	3,4	5,1	1,4	1,3	1,0	0,0	2,6	2,5	0,3	8,9
DWG	0,1	0,1	1,2	0,2	0,3	94,4	0,0	0,0	0,0	0,2	0,0	0,0	0,0	0,9	0,0	0,6	0,0	1,1	0,6	0,0	0,1
EML	0,0	0,0	0,0	0,0	0,0	0,0	95,5	0,0	2,9	0,0	0,0	0,0	0,0	0,0	0,4	0,0	0,0	0,1	0,0	1,0	0,0
EXE	4,6	0,0	3,4	1,7	0,2	0,2	0,0	34,5	0,0	1,5	17,9	6,5	8,9	0,2	0,0	0,1	0,0	0,1	4,4	0,0	15,8
HTML	0,0	0,0	0,3	0,0	0,2	0,0	40,3	0,0	53,1	0,2	0,0	0,0	0,0	0,0	2,5	0,4	0,0	0,3	0,0	2,4	0,0
JAR	1,3	0,1	1,0	1,6	0,3	0,1	0,0	12,5	0,2	45,7	8,9	7,0	2,8	0,8	0,7	0,7	0,0	0,1	7,9	1,1	7,1
JPG	4,1	0,0	0,2	3,1	0,3	0,3	0,0	12,8	0,0	1,3	43,4	5,1	7,5	4,3	0,0	0,1	0,1	0,2	7,5	0,0	9,7
MP3	8,6	0,0	0,0	0,4	0,2	0,0	0,0	5,0	0,0	3,2	1,1	47,4	11,9	1,0	0,1	0,0	0,0	14,5	0,0	6,8	
MP4	8,8	0,0	0,1	2,0	0,2	0,0	0,0	15,5	0,0	0,7	10,1	8,6	29,9	0,3	0,0	0,1	0,0	0,0	4,0	0,0	19,7
PDF	1,9	0,2	0,9	3,3	0,5	1,8	0,0	8,7	0,0	2,1	22,4	5,8	6,0	26,1	0,0	0,3	0,0	0,2	8,0	0,0	11,7
RTF	0,2	0,8	0,4	0,1	1,3	0,1	0,1	0,2	2,7	0,6	0,2	0,2	0,3	0,1	87,3	0,3	0,0	0,7	0,1	4,0	0,3
SQL	0,5	0,3	2,6	0,2	1,0	0,8	0,0	0,2	0,8	0,7	0,7	0,4	0,9	0,4	0,9	76,1	0,0	11,1	0,4	0,9	1,0
TIF	1,9	2,7	0,0	0,3	0,0	0,0	0,0	2,8	0,0	0,9	1,0	1,1	1,2	0,0	0,0	0,0	84,1	0,0	1,8	0,0	1,8
TXT	0,3	0,0	0,5	0,1	0,1	0,4	0,1	0,5	0,4	0,2	0,2	0,4	0,4	0,2	0,7	1,3	0,0	92,8	0,5	0,5	0,5
WMV	10,1	0,0	0,2	0,6	0,2	0,3	0,0	7,8	0,0	2,9	2,3	10,2	3,8	1,7	0,0	0,1	0,3	0,5	57,3	0,0	1,7
XML	0,0	0,0	0,3	0,0	0,1	0,0	2,0	0,0	2,9	0,5	0,0	0,0	0,0	0,0	4,0	0,5	0,0	0,6	0,0	89,1	0,0
ZIP	8,1	0,0	0,1	3,4	0,2	0,3	0,0	15,7	0,0	1,9	17,5	5,8	9,7	0,9	0,0	0,0	0,1	0,0	6,7	0,0	29,4

Tabela D.13: Matriz de confusão do classificador multinomial para fragmentos de 2 *Kbytes* com dados normalizados, discretizados e com redução de atributos.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 2 KBYTES (NORMALIZADO, DISCRETIZADO E COM REDUÇÃO DE ATRIBUTOS)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQLI	TIF	TXT	WMV	XML	ZIP
AVI	69,1	0,1	0,2	0,2	0,1	0,1	0,0	9,3	0,0	0,4	0,9	6,7	2,9	1,7	0,0	0,0	0,0	0,1	7,1	0,0	1,2
BMP	0,1	93,9	0,3	0,1	0,4	0,2	0,1	0,0	0,1	0,2	0,0	0,0	0,1	0,1	1,0	0,3	2,9	0,0	0,1	0,1	0,1
DLL	0,4	0,6	83,0	0,3	1,9	2,1	0,1	1,3	0,2	1,0	0,8	0,2	0,4	0,9	0,7	3,2	0,1	1,6	0,4	0,4	0,5
DOCX	5,9	0,2	0,3	11,8	0,6	0,4	0,0	22,7	0,0	1,3	23,6	5,8	7,2	2,1	0,0	0,1	0,0	0,0	5,3	0,0	12,6
DOC	6,8	0,3	1,8	1,0	37,4	0,3	0,0	17,8	0,0	0,8	12,7	2,6	3,6	1,2	0,7	1,1	0,1	1,8	4,5	0,1	5,3
DWG	0,6	0,2	1,5	0,2	0,4	88,1	0,0	0,2	0,0	1,1	0,5	1,0	0,7	1,1	0,1	0,8	0,0	0,7	2,6	0,0	0,3
EML	0,0	0,0	0,1	0,0	0,0	0,0	94,9	0,0	3,3	0,1	0,0	0,0	0,0	0,0	0,2	0,0	0,0	0,1	0,0	1,4	0,0
EXE	6,8	0,0	1,9	0,4	0,2	0,2	0,0	52,7	0,0	0,4	17,6	1,7	3,5	0,2	0,0	0,1	0,1	0,1	1,7	0,0	12,6
HTML	0,0	0,0	0,1	0,0	0,1	0,0	21,7	0,0	75,6	0,1	0,0	0,0	0,0	0,0	1,2	0,1	0,0	0,1	0,0	0,9	0,0
JAR	1,9	0,2	0,8	1,3	0,6	0,4	0,0	9,8	0,1	56,0	9,8	4,0	2,2	1,1	0,4	0,7	0,1	0,0	5,1	0,4	5,0
JPG	3,6	0,0	0,1	0,9	0,5	0,4	0,0	13,9	0,0	0,7	47,4	4,8	7,3	3,9	0,0	0,0	0,0	0,0	6,0	0,0	10,4
MP3	5,8	0,0	0,0	0,4	0,2	0,0	0,0	6,9	0,0	1,5	0,4	61,6	4,0	1,0	0,1	0,0	0,0	0,0	16,2	0,0	1,8
MP4	10,7	0,0	0,2	0,6	0,3	0,1	0,0	19,4	0,0	0,5	6,6	8,0	31,2	2,0	0,0	0,1	0,0	0,0	4,3	0,0	16,1
PDF	3,0	0,1	0,9	1,0	0,3	1,7	0,1	7,7	0,0	2,2	15,4	11,4	7,8	34,2	0,0	0,2	0,1	0,1	8,4	0,1	5,3
RTF	0,2	0,5	0,3	0,1	0,6	0,1	0,0	0,3	3,7	0,3	0,1	0,1	0,2	0,1	88,6	0,2	0,0	3,8	0,1	0,5	0,2
SQLI	0,1	0,3	3,4	0,1	1,1	0,8	0,2	0,1	0,1	0,7	0,4	0,3	0,6	0,2	0,2	84,3	0,0	6,2	0,3	0,1	0,4
TIF	1,5	1,0	0,0	0,0	0,1	0,0	0,0	2,6	0,0	0,1	0,4	0,2	0,3	0,0	0,0	0,0	92,1	0,0	0,9	0,0	0,6
TXT	0,2	0,0	0,3	0,0	0,1	0,5	0,1	0,3	0,2	0,1	0,1	0,2	0,1	0,1	1,7	0,6	0,0	94,9	0,2	0,2	0,2
WMV	12,9	0,0	0,2	0,2	0,7	0,9	0,0	8,4	0,0	1,3	2,9	13,1	3,3	1,1	0,0	0,1	0,1	0,2	52,3	0,0	2,3
XML	0,0	0,0	0,3	0,0	0,0	0,0	2,8	0,0	0,9	0,2	0,0	0,0	0,0	0,0	0,5	0,1	0,0	0,4	0,0	94,6	0,0
ZIP	8,4	0,1	0,3	1,4	0,5	0,5	0,0	24,9	0,0	1,6	21,3	3,6	8,9	1,1	0,1	0,1	0,1	0,0	4,6	0,0	22,6

Tabela D.14: Matriz de confusão do classificador multinomial para fragmentos de 4 *Kbytes* com dados normalizados, discretizados e com redução de atributos.

classif. como →	CLASSIFICADOR MULTINOMIAL – FRAGMENTOS DE 4 KBYTES (NORMALIZADO, DISCRETIZADO E COM REDUÇÃO DE ATRIBUTOS)																				
	AVI	BMP	DLL	DOCX	DOC	DWG	EML	EXE	HTML	JAR	JPG	MP3	MP4	PDF	RTF	SQLI	TIF	TXT	WMV	XML	ZIP
AVI	74,4	0,0	0,2	0,3	0,1	0,1	0,0	6,2	0,0	0,4	0,5	4,2	4,0	0,6	0,0	0,0	1,0	0,1	6,4	0,0	1,5
BMP	0,0	97,0	0,2	0,2	0,2	0,2	0,0	0,0	0,0	0,1	0,0	0,1	0,0	0,1	0,1	0,2	1,3	0,0	0,0	0,0	0,1
DLL	0,5	0,6	78,7	0,4	2,4	2,2	0,1	3,0	0,0	1,0	0,9	0,8	0,6	1,0	0,5	2,7	0,2	0,9	0,7	0,5	2,4
DOCX	7,2	0,2	0,3	15,8	1,2	0,3	0,0	11,3	0,0	1,6	14,6	3,5	4,2	1,7	0,0	0,2	1,6	0,0	5,0	0,0	31,4
DOC	3,9	0,3	2,0	1,8	39,4	0,5	0,0	6,2	0,0	0,9	12,5	2,2	2,3	1,0	0,3	1,1	1,4	0,0	4,7	0,1	19,5
DWG	0,4	0,2	1,9	0,3	0,5	86,0	0,0	0,2	0,0	2,6	0,3	2,0	0,4	0,8	0,1	1,0	0,2	0,4	2,2	0,0	0,4
EML	0,0	0,0	0,0	0,0	0,0	0,0	96,8	0,0	1,8	0,0	0,0	0,0	0,0	0,0	0,1	0,1	0,0	0,0	0,0	1,1	0,0
EXE	4,2	0,1	5,2	0,5	0,6	0,5	0,0	42,6	0,0	1,0	7,3	3,7	1,3	0,7	0,1	0,5	2,1	0,5	1,2	0,1	27,9
HTML	0,0	0,0	0,0	0,0	0,0	0,0	11,0	0,0	88,1	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,0	0,0	0,6	0,0	0,0
JAR	1,5	0,2	0,7	1,0	0,4	0,3	0,0	6,4	0,0	56,3	6,9	4,3	1,1	1,2	0,1	0,6	1,5	0,0	4,6	0,3	12,6
JPG	5,1	0,0	0,1	0,7	0,5	0,2	0,0	9,1	0,0	1,4	36,0	2,5	5,1	2,0	0,0	0,1	3,0	0,1	6,8	0,0	27,1
MP3	5,2	0,0	0,0	0,6	0,2	0,5	0,0	1,5	0,0	2,9	0,1	70,0	1,7	0,6	0,0	0,1	1,1	0,1	14,4	0,0	0,9
MP4	14,3	0,0	0,2	0,6	0,4	0,1	0,0	11,4	0,0	0,5	3,3	4,3	35,3	2,0	0,0	0,1	0,7	0,0	4,7	0,0	22,1
PDF	3,7	0,1	0,8	0,9	0,5	1,2	0,1	4,0	0,0	1,8	7,3	8,4	7,4	46,5	0,0	0,2	1,1	0,0	4,9	0,1	11,0
RTF	0,2	0,2	0,3	0,0	0,3	0,1	0,0	0,1	0,5	0,2	0,0	0,0	0,1	0,0	94,1	0,1	0,0	3,2	0,1	0,2	0,2
SQLI	0,4	0,2	2,2	0,2	1,5	0,7	0,1	0,6	0,0	0,7	1,0	0,5	0,5	0,3	0,1	84,0	0,1	5,0	0,6	0,3	0,9
TIF	6,6	0,6	0,0	0,2	0,1	0,1	0,0	5,4	0,0	0,8	8,1	0,4	0,4	0,1	0,0	0,0	67,3	0,0	6,5	0,0	3,2
TXT	0,1	0,0	0,2	0,0	0,1	0,3	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,0	1,9	0,2	0,0	96,1	0,2	0,1	0,2
WMV	12,6	0,0	0,1	0,5	0,4	0,6	0,0	5,0	0,0	1,6	5,8	10,6	6,5	1,1	0,0	0,1	1,6	0,1	40,4	0,0	12,8
XML	0,0	0,0	0,4	0,0	0,1	0,0	1,8	0,0	0,6	0,1	0,0	0,0	0,0	0,0	0,4	0,4	0,0	0,1	0,0	96,0	0,0
ZIP	5,5	0,1	0,4	1,4	0,7	0,3	0,0	13,2	0,0	1,7	11,4	1,3	4,9	1,4	0,1	0,1	1,6	0,0	3,0	0,0	52,9