10th Brazilian Congress on Computational Intelligence (CBIC'2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil

AGRUPAMENTO DE DOCUMENTOS APLICADO À COMPUTAÇÃO FORENSE: UMA ABORDAGEM PARA APERFEIÇOAR ANÁLISES PERICIAIS DE COMPUTADORES

Luís Filipe da Cruz Nassif[1,2] e **Eduardo Raul Hruschka**[1,3]
[1]Universidade de Brasília (UnB), [2]Departamento de Polícia Federal (DPF), [3]Universidade de São Paulo (ICMC/USP)
nassif.lfcn@dpf.gov.br, erh@icmc.usp.br

Resumo – Em análises periciais de computadores, usualmente são examinados centenas de milhares de arquivos. Grande parte dos dados desses arquivos é constituída por texto não estruturado, cuja análise por parte dos peritos é difícil de ser realizada. Nesse contexto, o uso de métodos automatizados de análise baseados na mineração de textos é de grande interesse. Particularmente, algoritmos de agrupamento podem facilitar a descoberta de conhecimentos novos e úteis nos textos sob análise. Este trabalho apresenta uma abordagem para aplicar agrupamento de documentos em análises periciais de computadores apreendidos em operações policiais. Para ilustrar tal abordagem, foi realizado um estudo comparativo de seis algoritmos de agrupamento de dados (*K-means, K-medoids, Single Link, Complete Link, Average Link* e CSPA) aplicados a cinco bases de dados textuais provenientes de investigações reais. Foram realizados experimentos utilizando-se diferentes combinações de parâmetros, totalizando dezesseis instanciações diferentes dos algoritmos. Adicionalmente, dois índices de validade relativos (Silhueta e sua versão simplificada) foram utilizados para estimar automaticamente o número de grupos. Estudos relacionados encontrados na literatura se mostram significativamente mais limitados do que o estudo aqui apresentado, especialmente ao se considerar a combinação de características do estudo experimental realizado, que envolve uma significativa variedade de algoritmos e a estimativa automática do número de grupos. Nesse contexto, o presente estudo poderá servir como ponto de partida para aqueles interessados em desenvolver pesquisas neste domínio de aplicação específico. Além disso, os experimentos realizados mostram que os algoritmos hierárquicos *Average Link* e *Complete Link* proporcionaram os melhores resultados. Os algoritmos particionais *K-means* e *K-medoids*, quando adequadamente inicializados, apresentaram resultados similares àqueles obtidos pelos algoritmos hierárquicos. Este estudo também apresenta e discute diversos resultados práticos mais específicos que podem ser úteis para pesquisadores e praticantes de análises forenses computacionais.

Palavras-chave – Agrupamento de dados, mineração de textos, Computação Forense.

1 Introdução

Estima-se que o volume de dados no universo digital aumentou de 161 hexabytes em 2006 para 988 hexabytes em 2010 [1] – aproximadamente 18 vezes a quantidade de informação presente em todos os livros já escritos – e continua crescendo de forma exponencial. Essa grande quantidade de dados tem impacto direto na área da *Computação Forense*, que trata da análise ou perícia de vestígios digitais visando à produção de provas durante investigações e processos judiciais, tanto cíveis quanto criminais, contribuindo, dessa forma, para a melhoria da Segurança Pública e da Justiça. Normalmente as análises periciais de computadores envolvem examinar centenas de milhares de arquivos por disco rígido, excedendo a capacidade de análise e interpretação dos vestígios por parte do perito e tornando imprescindível o uso de métodos automatizados de análise dos repositórios de dados sob exame. Nesse contexto, o uso de métodos de mineração de dados, com ênfase na descoberta de padrões de conteúdo nos arquivos, é promissor, pois pode contribuir para uma melhor organização dos dados por assunto e, consequentemente, para a descoberta de informações novas e úteis para as investigações. É importante ressaltar que a maior parte (cerca de 80%) dos dados de empresas e organizações, que frequentemente são objeto de exames periciais, se constitui de dados não estruturados [1], formados em grande parte por texto em linguagem natural. Assim, é muito promissora a utilização, em exames periciais, de métodos de mineração de dados textuais [2]. Em particular, métodos para agrupamento de dados textuais são de grande interesse para a perícia computacional.

Métodos para agrupamento (*clustering*) de dados têm como objetivo induzir grupos (*clusters*) de dados, de tal forma que objetos pertencentes ao mesmo grupo sejam mais semelhantes entre si do que objetos pertencentes a grupos distintos [3]. Algoritmos de agrupamento são normalmente utilizados em análises exploratórias de dados, nas quais se dispõe de pouco ou nenhum conhecimento sobre os dados [4]. Esse é precisamente o caso encontrado em exames periciais. Nesse tipo de aplicação, bases de dados formadas por objetos com rótulos de classes usualmente não estão disponíveis, pois não se conhece, *a priori*, as classes de documentos que poderiam ser encontradas. Mesmo assumindo que se disponha de uma base de dados rotulada, obtida a partir de perícias anteriores, há poucas chances de que as mesmas classes (possivelmente aprendidas anteriormente por um classificador, num processo de aprendizado supervisionado) continuem sendo válidas para novas amostras de dados, obtidas a partir de outros computadores e vinculados a processos de investigação diferentes. Mais precisamente, a probabilidade de que os dados seriam oriundos de diferentes populações seria alta, o que inviabilizaria a inferência estatística a partir de modelos existentes. Nesse contexto, algoritmos para aprendizado não supervisionado são promissores. Em particular, documentos previamente desconhecidos, mas com mesmo padrão de conteúdo, poderiam ser alocados no mesmo grupo, dessa forma, facilitando a análise dos documentos presentes em computadores apreendidos. Nesse sentido, o perito poderia se concentrar inicialmente em analisar documentos representativos de cada grupo e, a partir dessa análise preliminar, eventualmente decidir pelo exame detalhado dos demais documentos de cada grupo em questão. Em outras palavras, o agrupamento de documentos pode evitar o exame de todos os documentos individualmente e, no pior caso, o perito

ainda poderia optar por analisá-los em sua totalidade. Num cenário mais prático e realista, no qual especialistas de domínio (*e.g.*, peritos criminais) são escassos e dispõem de tempo limitado, é razoável assumir que, após encontrar um documento relevante, o perito poderia priorizar a análise dos outros documentos pertencentes ao grupo do documento encontrado, pois é provável que tais documentos também sejam relevantes para a investigação. A adoção da abordagem pericial baseada em agrupamento de documentos pode melhorar a eficiência do processo de análise de computadores apreendidos, conforme será discutido em maiores detalhes no presente artigo.

Este trabalho investiga uma abordagem baseada em agrupamento de dados para aperfeiçoar análises periciais de computadores. Considerando que métodos para agrupamento de dados têm sido estudados há décadas, e que a literatura sobre o assunto é extensa, optou-se por escolher um conjunto de (seis) algoritmos representativos para verificar a viabilidade da abordagem proposta. Particularmente, foram comparados os algoritmos particionais *K-means* [4] e *K-medoids* [5], os hierárquicos *Single Link*, *Complete Link* e *Average Link* [6], e o algoritmo de consenso entre partições CSPA [7], incluindo algumas variações de parâmetros, totalizando 16 instanciações diferentes de algoritmos. Dessa forma, pode-se também comparar os desempenhos relativos desses algoritmos ao serem aplicados para analisar conteúdos de arquivos em computadores apreendidos. Todos os algoritmos foram avaliados em bases de dados textuais obtidas a partir de cinco casos reais de perícia realizados na *Polícia Federal*. A fim de tornar a análise comparativa entre algoritmos mais realista do ponto de vista prático, dois índices de validade relativos (Silhueta [5] e sua versão simplificada [8]) foram usados para estimar o número de grupos automaticamente a partir dos dados. É importante notar que o número de grupos é um parâmetro crítico de vários algoritmos e é normalmente desconhecido *a priori*. Além disso, não foram encontrados na literatura estudos sobre algoritmos de agrupamento hierárquicos e de consenso entre partições aplicados à *Computação Forense*, nem sobre a estimativa automática do número de grupos. Essa característica do estudo realizado é, portanto, uma contribuição importante deste artigo, cujas demais seções estão organizadas como segue. A próxima seção aborda trabalhos relacionados, enquanto que a Seção 3 descreve resumidamente os algoritmos de agrupamento utilizados. A Seção 4 reporta a avaliação experimental realizada e, finalmente, a Seção 5 apresenta as principais conclusões derivadas do trabalho realizado.

2 Trabalhos Relacionados

Existem poucos trabalhos na literatura reportando o uso de técnicas de agrupamento de dados para *Computação Forense*. Essencialmente, a maioria dos trabalhos adota alguns algoritmos clássicos para agrupamento de dados - e.g., *Expectation-Maximization* (EM) para aprendizado não supervisionado de mistura de gaussianas, *K-means*, *Fuzzy C-means* (FCM) e redes neurais do tipo mapas auto-organizáveis (*Self Organizing Maps* - SOM). Esses algoritmos possuem propriedades bem conhecidas e são amplamente usados na prática. Mais especificamente, os algoritmos *K-means* e FCM podem ser vistos como casos particulares do EM. Algoritmos do tipo SOM, por sua vez, em geral apresentam bias indutivo semelhante ao *K-means*.

Em [9] redes neurais artificiais do tipo SOM foram usadas no agrupamento de arquivos para auxiliar a tomada de decisão dos peritos e tornar o processo de análise mais eficiente. Os arquivos foram agrupados com base nas suas datas de criação, extensões e horários de criação. Em [10] foi proposto o uso de algoritmos do tipo SOM para agrupar tematicamente os resultados de buscas por palavras chave obtidos durante exames periciais. A hipótese subjacente é que o agrupamento dos resultados aumentaria a eficiência do processo de recuperação da informação, pois não seria necessário a revisão de todos os documentos encontrados pelo usuário. Um ambiente integrado de mineração de *e-mails* para análises periciais, utilizando algoritmos de classificação e agrupamento, foi apresentado em [11]. Posteriormente, foi proposta a extração de características de estilos de escrita de um conjunto de *e-mails* anônimos para posterior agrupamento das mensagens por autor [12]. Foram avaliadas várias combinações de características, como léxicas, sintáticas, estruturais e específicas de domínio, e três algoritmos de agrupamento (*K-means*, *Bisecting K-means* e *EM*). Agrupamento de dados de e-mails para fins perciais também foi discutido em [13], no qual foi aplicado uma variante do algoritmo *K-Means* baseada em funções do tipo *Kernel*. Os resultados dessa aplicação foram analisados subjetivamente, e os autores concluíram que os resultados são interessantes e úteis do ponto de vista de uma investigação, pois proporcionam uma visualização geral dos dados, separados por assunto, sem a necessidade de se analisar individualmente o conteúdo de todos os arquivos. Mais recentemente, um método para inferir regras de associação a partir de dados forenses, supostamente fáceis de serem compreendidas por policiais e outros especialistas de domínio, foi descrito em [14]. O método é baseado no algoritmo de agrupamento probabilístico FCM. Os autores afirmam que foram obtidas regras de associação muito boas, mas foi difícil de gerar um significado semântico intuitivo para algumas das relações de pertinência, o que pode prejudicar a compreensão das regras por parte dos especialistas de domínio.

Todos os trabalhos reportados na literatura assumem que o usuário determina, *a priori*, o número de grupos (*clusters*) a serem obtidos. Evidentemente, tal parâmetro é de difícil escolha em situações práticas. Essa limitação prática pode ser contornada pelo uso de métodos que basicamente aplicam determinado algoritmo de agrupamento de dados (e.g., *K-Means*) múltiplas vezes, variando-se a quantidade de grupos, e, a partir do conjunto de partições obtidas, escolhem, de acordo com algum critério numérico, a melhor partição dentre aquelas obtidas [15]. O presente trabalho faz uso de tais métodos que permitem estimar o número de grupos a partir dos dados, facilitando o trabalho do perito, que, na maior parte das situações, dificilmente saberia estimar, *a priori*, o número de grupos presente em determinada base de dados. Além disso, surpreendentemente, a literatura não faz menção ao uso de algoritmos clássicos de agrupamento hierárquicos aplicados ao domínio de aplicação aqui abordado. O presente estudo considera tais algoritmos clássicos, bem como avanços recentes na área de agrupamento de dados – tais como o uso de partições de consenso.

3 Métodos de Agrupamento Utilizados

Antes da aplicação dos algoritmos e técnicas de agrupamento de dados nas bases textuais, foi necessário realizar um pré-processamento dos documentos. Nessa etapa, foram eliminadas palavras sem significado semântico útil (*stopwords*) e foi utilizado um algoritmo de radicalização (*stemming)* para palavras da Língua Portuguesa. Foi adotada então uma abordagem estatística tradicional para minerar textos. Nessa abordagem, os documentos são representados em um modelo de espaço vetorial [16], no qual cada documento é representado por um vetor que contém as frequências das ocorrências das palavras. Utilizou-se também de uma técnica de redução de dimensionalidade – *Term Variance* (TV) [17] – para aumentar a eficácia e eficiência dos algoritmos de agrupamento. A TV seleciona os 100 atributos (palavras) com maiores variâncias de frequência nos documentos. Para a aplicação dos algoritmos de agrupamento, foram usadas duas medidas de distância entre os documentos, a saber: uma baseada no cosseno entre vetores [16], usualmente utilizada em aplicações de mineração de textos, e outra baseada na distância de Levenshtein [18], utilizada para calcular as distâncias entre os nomes dos documentos. A seguir, serão brevemente abordados os seis algoritmos de agrupamentos de dados utilizados nos experimentos.

O algoritmo *K-means* [4] é um algoritmo particional amplamente utilizado na prática devido à sua simplicidade e eficiência computacional. Trata-se de um algoritmo de otimização que visa a minimizar a seguinte função objetivo:

$$SEQ = \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in \boldsymbol{C}_i} d(\boldsymbol{x}, \overline{\boldsymbol{x}}_i)^2 \quad (1)$$

onde *K* é o número de grupos, $\boldsymbol{x}$ representa um objeto (i.e., documento) e $\overline{\boldsymbol{x}}_i$ é o centroide do i-ésimo grupo $\boldsymbol{C}_i$, definido por:

$$\overline{\boldsymbol{x}}_i = \frac{1}{n_i} \sum_{\boldsymbol{x} \in \boldsymbol{C}_i} \boldsymbol{x} \quad (2)$$

onde n_i é o número de objetos do grupo $\boldsymbol{C}_i$. O algoritmo *K-means* pode ser descrito pelo seguinte procedimento iterativo:

1. Escolher aleatoriamente *K* protótipos iniciais (usualmente objetos da base de dados);
2. Associar cada objeto ao centroide mais próximo, formando *K* grupos;
3. Atualizar os centroides, segundo a Eq. 2;
4. Repetir os passos 2 e 3 até convergir (na prática normalmente se adota um número máximo de iterações);

O algoritmo particional *K-medoids* [5] é semelhante ao *K-means*. Entretanto, ao invés de computar centroides, esse algoritmo utiliza medoides, que são objetos representativos dos grupos. Isso permite utilizar o algoritmo quando não se podem calcular centroides, mas se dispõe de distâncias entre pares de objetos. No presente trabalho, a distância de Levenshtein entre os nomes dos documentos é computada, fornecendo uma matriz de proximidade. Nesse contexto, o algoritmo *K-medoids* se afigura interessante. O *K-medoids* pode ser descrito pelos seguintes passos:

1. Escolher aleatoriamente *K* medoides iniciais dentre os dados;
2. Associar cada objeto ao medoide mais próximo, formando *K* grupos;
3. Para cada grupo, escolher como medoide o objeto que minimize a soma dos erros quadráticos (Eq. 1, neste caso, utilizando medoides ao invés de centroides);
4. Repetir os passos 2 e 3 até convergir.

Os dois algoritmos acima apresentados são muito sensíveis à inicialização inicial e usualmente convergem para soluções que representam mínimos locais. Para tentar minimizar esses problemas, foi usada uma inicialização não aleatória na qual objetos distantes entre si são escolhidos [19]. Tal abordagem pode ser sumarizada da seguinte forma:

1. Selecionar aleatoriamente um protótipo inicial;
2. Para cada objeto restante $\boldsymbol{x}_i$, definir d_i como a distância para o protótipo mais próximo;
3. Definir como novo protótipo o objeto com maior valor de d_i;
4. Repetir os passos 2 e 3 até que sejam escolhidos *K* protótipos iniciais.

Outra categoria de algoritmos clássicos de agrupamento são os hierárquicos. Diferentemente dos particionais, os algoritmos hierárquicos fornecem um conjunto de partições aninhadas [5], usualmente representadas sob a forma de um dendrograma. Um algoritmo hierárquico aglomerativo pode ser descrito genericamente pelos seguintes passos [6]:

1. Seja $\widetilde{N}$ o número de grupos atual. Comece com $\widetilde{N} = N$ e calcule a respectiva a matriz de distâncias;
2. Seja $\boldsymbol{C}_i, 1 \leq i \leq \widetilde{N}$, um grupo qualquer. Faça $\boldsymbol{C}_i = \boldsymbol{C}_i \cup \boldsymbol{C}_j$ tal que $d(\boldsymbol{C}_i, \boldsymbol{C}_j) = d_{min}(\boldsymbol{C}_k, \boldsymbol{C}_l)\ \forall\ k, l \in \{1..\widetilde{N}\}, k \neq l$, e descarte o grupo $\boldsymbol{C}_j$;
3. Faça $\widetilde{N} = \widetilde{N} - 1$ e atualize $d(\boldsymbol{C}_i, \boldsymbol{C}_j)\ \forall\ j \in \{1..\widetilde{N}\}$ na matriz de distâncias;
4. Repita os passos 2 e 3 até obter $\widetilde{N} = 1$.

onde *N* representa o número de objetos da base. O conceito de distância entre dois grupos $d(\boldsymbol{C}_i, \boldsymbol{C}_j)$ difere de acordo com o algoritmo hierárquico. A seguir, são apresentadas as definições de $d(\boldsymbol{C}_i, \boldsymbol{C}_j)$ dos algoritmos usados neste trabalho [6]:

1. *Single Link*: $d(\boldsymbol{C}_i, \boldsymbol{C}_j) = d_{min}(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x} \in \boldsymbol{C}_i, \boldsymbol{y} \in \boldsymbol{C}_j$
2. *Complete Link*: $d(\boldsymbol{C}_i, \boldsymbol{C}_j) = d_{max}(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x} \in \boldsymbol{C}_i, \boldsymbol{y} \in \boldsymbol{C}_j$

3. *Average Link*: $d(\boldsymbol{C}_i, \boldsymbol{C}_j) = |\boldsymbol{C}_i|^{-1}|\boldsymbol{C}_j|^{-1} \sum_{\boldsymbol{x}\in\boldsymbol{C}_i, \boldsymbol{y}\in\boldsymbol{C}_j} d(\boldsymbol{x}, \boldsymbol{y})$

Neste trabalho, também foi utilizado o algoritmo de agrupamento de consenso entre partições *Cluster-based Similarity Partitioning Algorithm* (CSPA) [7]. Fundalmentalmente, o CSPA produz um agrupamento de consenso obtido a partir de um conjunto (*ensemble*) de agrupamentos diferentes. Conceitualmente, o CSPA define uma nova medida de similaridade entre os objetos, na qual a presença de dois objetos em um mesmo grupo, em cada partição do conjunto, é computada como um voto de similaridade, gerando uma matriz de co-associação [20]. Posteriormente, essa medida de similaridade é utilizada por um algoritmo de agrupamento que trabalhe a partir de uma matriz de proximidades – e.g., *K-medoids* – para produzir o agrupamento final de consenso. Neste estudo, os conjuntos de partições foram gerados de duas formas diferentes. Na primeira, o algoritmo *K-means* é executado 100 vezes, utilizando subconjuntos de atributos diferentes do modelo de espaço vetorial em cada execução. O outro tipo de conjunto utilizado é composto apenas por duas partições dos dados. Uma delas é obtida pelo algoritmo *K-medoids*, a partir das dissimilaridades entre as nomenclaturas dos arquivos – utilizando a distância de Levenshtein – e a outra partição é obtida pelo algoritmo *K-means*, a partir das dissimilaridades entre os conteúdos dos arquivos – utilizando a distância baseada no cosseno. Neste caso, cada uma das partições pode ter um peso diferente. Nos experimentos realizados no presente trabalho, os pesos foram variados entre 0 e 1 (em incrementos de 0,1 e mantendo-se sua soma igual a 1) a fim de se investigar valores que potencialmente poderiam "otimizar" o consenso entre as duas partições.

Em relação aos métodos para estimativa automática do número de grupos, *K*, foi utilizado o índice de validade relativo denominado Silhueta [5] e sua versão simplificada [8], os quais apresentaram resultados muito bons num estudo recente [15]. A Silhueta pode ser definida da seguinte forma: seja a a distância média de um objeto $\boldsymbol{x}$ a todos os outros objetos pertencentes ao seu grupo e seja b a distância média de $\boldsymbol{x}$ a todos os objetos do grupo vizinho mais próximo. Então, a Silhueta individual do objeto $\boldsymbol{x}$ é definida por:

$$s = (b - a)/\max(a, b) \tag{3}$$

Caso o objeto seja o único elemento do seu grupo (*singleton*), arbitrariamente atribui-se o valor "zero" para sua Silhueta [5]. A Silhueta (global) do agrupamento é dada pela média das Silhuetas de todos os objetos. Assim, o algoritmo de agrupamento é executado para vários valores de *K*, sendo escolhida a partição com maior valor de Silhueta. O custo computacional para computar a Silhueta pode ser elevado para algumas aplicações. Levando-se em conta essa eventual limitação, pode-se usar uma versão simplificada da Silhueta, em que a distância média aos objetos de determinado grupo é substituída pela distância ao centroide do grupo [8]. Também foi utilizada uma versão aqui denominada de recursiva da Silhueta, na qual os *singletons* são removidos. Nesse caso, após a remoção dos *singletons*, a Silhueta é aplicada novamente, até ser obtida uma partição sem *singletons*. Ao final do processo, para fins de avaliação, todos os *singletons* são incorporados à partição resultante. A Tabela 1 sumariza os métodos de agrupamento e respectivas variações de parâmetros utilizados nos experimentos.

Tabela 1 – Resumo dos algoritmos e respectivos parâmetros.

Sigla	Algoritmo	Atributos	Medida de Distância	Inicialização	Estimativa de *K*
Kms	*K-means*	conteúdo (todos)	cosseno	aleatória	Silhueta simplificada
Kms100	*K-means*	conteúdo (100 > TV)	cosseno	aleatória	Silhueta simplificada
Kms100*	*K-means*	conteúdo (100 > TV)	cosseno	[19]	Silhueta simplificada
KmsT100*	*K-means*	conteúdo (100 > TV)	cosseno	[19]	Silhueta
KmsS	*K-means*	conteúdo (todos)	cosseno	aleatória	Silhueta recursiva
Kms100S	*K-means*	conteúdo (100 > TV)	cosseno	aleatória	Silhueta recursiva
Kmd100	*K-medoids*	conteúdo (100 > TV)	cosseno	aleatória	Silhueta
Kmd100*	*K-medoids*	conteúdo (100 > TV)	cosseno	[19]	Silhueta
KmdLev	*K-medoids*	nome	Levenshtein	aleatória	Silhueta
KmdLevS	*K-medoids*	nome	Levenshtein	aleatória	Silhueta recursiva
AL100	*AverageLink*	conteúdo (100 > TV)	cosseno	-	Silhueta
CL100	*CompleteLink*	conteúdo (100 > TV)	cosseno	-	Silhueta
SL100	*SingleLink*	conteúdo (100 > TV)	cosseno	-	Silhueta
NC	CSPA	nome e conteúdo (todos)	co-associação	aleatória	Silhueta simplificada
NC100	CSPA	nome e conteúdo (100 > TV)	co-associação	aleatória	Silhueta simplificada
E100	CSPA	conteúdo (100 aleatórios)	co-associação	aleatória	Silhueta simplificada

4 Avaliação Experimental

4.1 *Bases de Dados*

Nos experimentos realizados, foram utilizadas cinco bases de dados textuais obtidas a partir de cinco investigações reais. Cada base de dados foi obtida a partir de um disco rígido diferente, sendo selecionados todos os documentos não duplicados com extensões "doc", "docx" e "odt". Posteriormente, o texto plano dos documentos foi extraído para que pudessem ser aplicados os métodos de mineração de textos. Conforme detalhado na próxima seção, a medida de avaliação utilizada pressupõe a existência de uma partição de referência para cada base de dados. Entretanto, como partições de referência dificilmente estão disponíveis em bases provenientes de investigações reais, elas foram definidas manualmente por um especialista de domínio,

através da inspeção do conteúdo dos documentos. As bases de dados contêm quantidades variadas de documentos (*N*), grupos (*K*), atributos (*D*), *singletons* (*S*) – grupos unitários – e número de documentos por grupo (#), conforme reportado na Tabela 2.

Tabela 2 – Propriedades das bases de dados utilizadas.

Base	*N*	*K*	*D*	*S*	# Maior Grupo
A	37	23	1744	12	3
B	111	49	7894	28	12
C	68	40	2699	24	8
D	74	38	5095	26	17
E	131	51	4861	31	44

4.2 Método de Avaliação

Do ponto de vista científico, o uso de partições de referência para avaliar algoritmos de agrupamento de dados é considerado o procedimento mais principiado. Nesse caso, as partições de referência são normalmente obtidas a partir de dados gerados sinteticamente, de acordo com alguma distribuição de probabilidades. Do ponto de vista prático, tais partições de referência são usualmente empregadas para se escolher um determinado algoritmo de agrupamento que seja mais apropriado para determinada aplicação, ou para calibrar seus parâmetros. Nesse caso, uma partição de referência é construída por um especialista de domínio e reflete as expectativas que ele tem sobre os grupos que deveriam ser encontrados numa determinada amostra da base de dados. Nesse sentido, o método de avaliação experimental utilizado neste trabalho se baseia no índice de validação externo *Ajusted Rand Index* (ARI) [21], o qual mede a correspondência entre a partição obtida por um algoritmo de agrupamento particional *P* e uma partição de referência *R*. Mais formalmente, o ARI∈[0,1] é definido como:

$$ARI = \left(a - \frac{(a+c)(a+b)}{a+b+c+d}\right) / \left(\frac{(a+c)+(a+b)}{2} - \frac{(a+c)(a+b)}{a+b+c+d}\right) \quad (4)$$

onde *a* é o número de pares de objetos pertencentes ao mesmo grupo em *P* e em *R*, *b* é o número de pares de objetos pertencentes ao mesmo grupo em *P* e a grupos diferentes em *R*, *c* é o número de pares de objetos pertencentes a grupos diferentes em *P* e ao mesmo grupo em *R* e *d* é o número de pares de objetos pertencentes a grupos diferentes em *P* e em *R*.

4.3 Resultados Obtidos

De um modo geral, o algoritmo hierárquico *Average Link* (AL100) apresentou os melhores resultados, tanto em relação à média de ARI, quanto em relação ao seu desvio padrão, o que indica uma maior consistência e estabilidade, conforme pode ser observado na Tabela 3 – note que valores de ARI próximos de 1 indicam que as partições encontradas são bastante aderentes às partições de referência. Nessa tabela, foram reportados os resultados dos algoritmos de consenso entre nome e conteúdo dos arquivos (NC e NC100) correspondentes às partições cujo peso de conteúdo resultou no maior valor de ARI. O algoritmo *Complete Link* (CL100), também hierárquico, apresentou desempenho muito próximo ao AL100. Entretanto, o algoritmo hierárquico *Single Link* (SL100) obteve um desempenho relativamente inferior aos outros dois algoritmos hierárquicos, principalmente nas bases A e B. Isso pode ter sido provocado pela sensibilidade do *Single Link* à presença de *outliers*, que podem ocasionar um encadeamento de objetos durante o agrupamento, mesclando grupos diferentes [3].

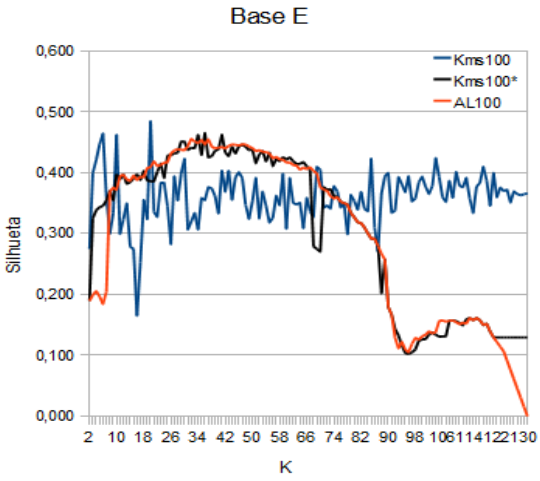
Tabela 3 – ARI e desvio padrão dos algoritmos

Algoritmo	Base A	Base B	Base C	Base D	Base E	Média	σ
AL100	0,94	0,83	0,89	0,99	0,90	0,91	0,06
CL100	0,94	0,76	0,89	0,98	0,90	0,89	0,08
KmsT100*	0,81	0,76	0,89	0,97	0,94	0,88	0,09
Kmd100*	0,81	0,76	0,89	0,96	0,93	0,87	0,08
SL100	0,54	0,63	0,90	0,98	0,88	0,79	0,19
NC100	0,66	0,64	0,78	0,74	0,72	0,71	0,06
Kms	0,61	0,60	0,69	0,79	0,84	0,71	0,11
NC	0,61	0,60	0,69	0,79	0,84	0,71	0,11
Kms100*	0,53	0,63	0,63	0,68	0,93	0,68	0,15
Kmd100	0,81	0,58	0,72	0,25	0,79	0,63	0,23
Kms100	0,64	0,64	0,78	0,29	0,72	0,62	0,19
KmsS	0,47	0,11	0,75	0,80	0,82	0,59	0,30
Kms100S	0,60	0,54	0,74	0,20	0,69	0,55	0,21
E100	0,61	0,10	0,29	0,76	0,08	0,37	0,31
KmdLevS	0,62	0,23	0,37	0,55	0,05	0,36	0,23
KmdLev	0,46	0,16	0,32	0,74	0,08	0,35	0,26

Merecem destaque os resultados dos algoritmos KmsT100* e Kmd100*. Os desempenhos desses algoritmos foram comparáveis aos melhores algoritmos hierárquicos – AL100 e CL100. Além disso, ressalta-se que os resultados dos algoritmos KmsT100* e Kmd100*, com inicialização em objetos distantes [19], foram melhores que suas respectivas versões com

inicialização em objetos aleatórios (Kms100 e Kmd100). Isso sugere que a estratégia de inicialização em objetos distantes minimiza o problema dos mínimos locais do *K-means* e *K-medoids* nas bases utilizadas. Para ilustrar melhor esse comportamento, a Figura 1 apresenta curvas dos valores de Silhueta em função de *K* para três algoritmos (Kms100, Kms100* e AL100) na base E. Observa-se que o algoritmo Kms100, com inicialização aleatória dos centroides, apresenta maior quantidade de máximos locais, gerando instabilidade de resultados. Opostamente, o algoritmo Kms100*, com a inicialização em objetos distantes, apresenta menor quantidade de máximos locais, sendo mais estável. Essa tendência se repete para as outras bases de dados. Surpreendentemente, o algoritmo Kms100* apresenta uma curva similar à curva do algoritmo AL100, principalmente para maiores valores de *K*. Esse fato pode ser justificado, em parte, porque ambos os algoritmos privilegiam a separação de *outliers*. Assim, o algoritmo Kms100* – que é computacionalmente mais eficiente – pode ser utilizado para alcançar resultados similares aos que seriam obtidos pelo algoritmo AL100.

Figura 1 – Silhuetas de Kms100, AL100 e Kms100*.



Também se pode observar que o algoritmo Kms100* obteve desempenho um pouco melhor do que a sua versão com inicialização em objetos aleatórios (Kms100). Entretanto, o algoritmo Kms100*, o qual utiliza a Silhueta simplificada para estimar *K*, obteve desempenho bastante inferior relativamente aos algoritmos KmsT100*, Kmd100* e aos algoritmos hierárquicos, que utilizam a versão tradicional da Silhueta. Essa observação sugere que a versão simplificada da Silhueta está fornecendo uma estimativa de *K* relativamente pior. Isso é justificável, pois, fundamentalmente, há perda de informação por parte da Silhueta simplificada, que é uma aproximação da Silhueta, utilizando distâncias aos centroides ao invés de distâncias médias aos objetos dos grupos. Em quatro das cinco bases de dados, a Silhueta simplificada forneceu valores de *K* menores do que ambas a Silhueta e a partição de referência. Em uma análise mais detalhada, a distância média de um determinado objeto a todos os objetos de um determinado grupo tende a ser maior do que a distância desse mesmo objeto ao centroide do grupo. E essa diferença tende a ser maior quando é considerado o grupo do próprio objeto e menor quando considerados outros grupos. Por isso, na Silhueta simplificada, o valor de a na Eq. 3 tende a diminuir mais do que o valor de b, fazendo com que a diferença $(b - a)$ seja maior, o que significa valores de Silhueta simplificada maiores que os de Silhueta. Entretanto, quanto menores os grupos – ou quanto maior o valor de *K* – a diferença entre a distância média aos objetos de um grupo e a distância ao centroide do grupo tende a diminuir. Logo, na Silhueta simplificada, a diferença $(b - a)$ tende a diminuir com o aumento de *K*, aproximando-se do valor de Silhueta e se igualando no caso extremo $K = N$, conforme ilustrado na Figura 2. Em todas as bases, o valor de Silhueta simplificada é maior que o de Silhueta para menores valores de *K* e diminui progressivamente com o aumento de *K*, até se igualar ao valor de Silhueta. Assim, a Silhueta simplificada favorece partições com menor número de grupos, o que prejudicou os algoritmos que a utilizaram nas bases avaliadas.

Figura 2 – Silhueta e Silhueta simplificada de Kms100*

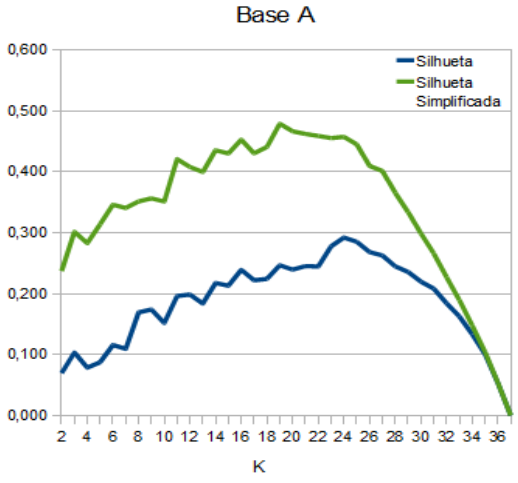
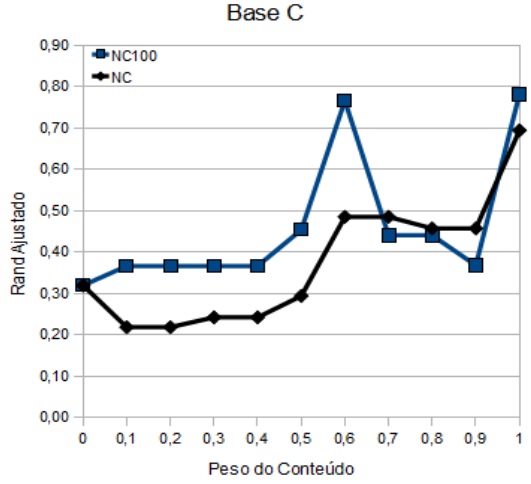


Figura 3 – ARI dos algoritmos NC e NC100



A utilização apenas da nomenclatura como medida de similaridade não trouxe bons resultados no geral – e.g., ver resultados para os algoritmos KmdLev e KmdLevS na Tabela 3. Esses resultados são esperados, pois o nome dos arquivos provê menos informações do que o seu próprio conteúdo. Entretanto, uma exceção a esse comportamento geral pode ser observada por meio dos resultados relativamente bons desses algoritmos na base D, sendo melhores até que alguns algoritmos baseados no conteúdo dos arquivos. Esse fato reforça a hipótese de que o nome dos arquivos, apesar de menos informativo que seu conteúdo, pode enriquecer o agrupamento se utilizado conjuntamente com este. Ainda nesse sentido, a Figura 3 apresenta, para a base C, valores de ARI dos algoritmos de consenso entre nome e conteúdo dos arquivos em função do peso do conteúdo no cálculo da matriz de co-associação. Nessa figura, o peso do conteúdo foi variado em intervalos de 0,1 e as linhas foram traçadas continuamente para facilitar a visualização da tendência da curva. Observa-se a ocorrência de um pico de ARI do algoritmo NC100 – de consenso entre nome e conteúdo dos arquivos com seleção dos 100 atributos de maior TV. Apesar desse resultado não ter ocorrido com todas as bases, aparentemente a adição de informações sobre nome ao conteúdo dos arquivos pode enriquecer o cálculo da similaridade entre eles, o que incentiva uma maior investigação futura desse aspecto.

O resultado relativamente inferior do algoritmo E100 merece explicação. Esse algoritmo obtém uma partição de consenso a partir de um conjunto de partições geradas por várias execuções do *K-means*, selecionando 100 atributos aleatórios em cada execução. Entretanto, devido à alta esparsidade dos vetores, comum em bases textuais, e devido à escolha aleatória dos atributos, muitos documentos são representados por vetores com a maioria dos atributos nulos, sendo então agrupados em um único grupo, cujo centroide é um vetor nulo. Isso fornece uma falsa informação de similaridade para o cálculo da matriz de co-associação, o que acaba prejudicando o agrupamento final de consenso. Logo, a escolha de atributos aleatórios para gerar conjuntos de partições em algoritmos de agrupamento de consenso não é uma boa estratégia para bases textuais.

Outro aspecto que deve ser abordado é o desempenho dos algoritmos com a aplicação recursiva da Silhueta. De acordo com a Tabela 3, os algoritmos KmsS e Kms100S apresentaram resultados relativamente inferiores às respectivas versões sem aplicação recursiva da Silhueta (Kms e Kms100). Já o algoritmo KmdLevS, baseado na similaridade entre os nomes dos documentos, apresentou resultado próximo à sua versão sem aplicação recursiva da Silhueta, KmdLev. Assim, os resultados sugerem que a remoção de *outliers* pela execução recursiva da Silhueta não foi adequada para as bases textuais utilizadas. Ainda de acordo com a Tabela 3, em três das cinco bases utilizadas, os algoritmos com seleção dos 100 atributos com maior TV obtiveram melhores resultados que os algoritmos com todos os atributos. Esse resultado sugere que a seleção de atributos com maior TV mantém a eficácia dos algoritmos comparável às suas versões com o uso de todos os atributos. Ao mesmo tempo, a eficiência computacional dos algoritmos é melhorada com a diminuição do tempo de execução, que normalmente é dependente do número de atributos processados.

Finalmente, em todas as bases de dados utilizadas, o melhor algoritmo de agrupamento (geralmente o AL100), obteve grupos contendo documentos relevantes e grupos contendo documentos irrelevantes, do ponto de vista pericial. Dentre os relevantes, podem ser citados grupos de documentos com comprovantes de transferências bancárias internacionais, com recibos de pagamento, com boletos de operações cambiais, com relatórios diários de movimentação financeira e grupos contendo procurações. Dentre os grupos irrelevantes, podem-se citar grupos contendo documentos em branco, documentos corrompidos, modelos de etiquetas, modelos de exemplo de aplicativo, avisos de período de férias, dentre outros. Dessa forma, os algoritmos de agrupamento se mostram muito úteis em exames periciais, pois podem auxiliar o perito a descartar grupos irrelevantes e concentrar a análise em grupos relevantes, sem precisar inspecionar todos os documentos individualmente, o que melhora a eficiência do processo de descoberta de conhecimento e de recuperação da informação.

5 Conclusões

Nos experimentos realizados, de modo geral, os algoritmos hierárquicos *Average Link* e *Complete Link* apresentaram os melhores resultados. Apesar de seus custos computacionais usualmente elevados, eles são indicados para bases de dados com poucas centenas de documentos. Além disso, os algoritmos particionais *K-means* e *K-medoids* foram menos afetados pela convergência para mínimos locais quando apropriadamente inicializados, apresentando resultados tão bons quanto os melhores algoritmos hierárquicos. Dessa forma, e levando-se em conta que o algoritmo *K-means* é bastante escalável, este é indicado para grandes bases de dados. Em relação aos métodos para estimativa do número de grupos, a Silhueta apresentou melhor resultado que sua versão simplificada, na qual fundamentalmente há perda de informação. Sua versão simplificada também induziu uma menor quantidade de grupos, o que foi prejudicial nas bases utilizadas, que apresentam grande quantidade de grupos. Entretanto, a Silhueta simplificada, por ser mais eficiente computacionalmente, pode ser mais adequada para utilização em bases de dados grandes. Adicionalmente, alguns resultados sugerem que a utilização do nome dos documentos em conjunto com informações de conteúdo, mediante o uso de algoritmos de consenso entre partições, pode enriquecer o processo de agrupamento de arquivos. Finalmente, foi possível observar que os algoritmos de agrupamento podem separar grupos de documentos relevantes de grupos de documentos irrelevantes. Assim, eles podem contribuir para com a descoberta de informações novas e úteis em exames periciais, tornando o processo de análise pericial mais eficiente.

Os resultados obtidos neste trabalho mostram que o uso de agrupamento de dados para auxiliar a análise pericial é promissor, podendo acelerar de maneira significativa a atividade pericial. Com o objetivo de adicionalmente alavancar o uso de algoritmos de agrupamento de dados em tais aplicações, sugere-se como trabalho futuro investigar formas de se rotular, automaticamente, os grupos obtidos. A atribuição de rótulos aos grupos permitiria ao perito identificar com maior rapidez o conteúdo semântico de cada grupo, mesmo antes de analisar o conteúdo dos documentos propriamente ditos. Por fim, a

avaliação de algoritmos que induzem partições sobrepostas (e.g., *Fuzzy C-Means* e *EM*) e que atribuem aos documentos probabilidades de pertinência a cada grupo é interessante.

6 Referências

1. Gantz, J. F., Reinsel, D., Chute, C., Schlichting, W., McArthur, J., Minton, S., Xheneti, I., Toncheva, A., e Manfrediz, A., The expanding digital universe: A forecast of worldwide information growth through 2010, External Publication of IDC **Information and Data**, (2007), 1 – 21.
2. Ebecken, N. F. F., Lopes, M. C. S., e Costa, M. C. A., Mineração de textos, Em Rezende, S. O., Sistemas Inteligentes: Fundamentos e Aplicações, **Manole**, (2003), 337 – 372.
3. Everitt, B. S., Landau, S., e Leese, M., Cluster Analysis, **Arnold**, (2001).
4. Jain, A. K., Dubes, R. C., Algorithms for Clustering Data, **Prentice-Hall**., (1988).
5. Kaufman, L, e Rousseeuw, P., Finding Groups in Data: An introduction to cluster analysis**, Wiley-Interscience**, (1990)**.**
6. Xu, R., Wunsch II, D. C., Clustering, **Wiley / IEEE Press**, (2009).
7. Strehl, A. e Ghosh, J., Cluster Ensembles: A Knowledge Reuse Framework for Combining Multiple Partitions, **Journal of Machine Learning Research**, 3 (2002), 583 – 617.
8. Hruschka, E. R., Campello, R. J. G. B., e de Castro, L. N., Evolving clusters in gene-expression data**, Information Sciences**, 176 (2006), 1898 – 1927.
9. Fei, B.K.L., Eloff, J.H.P., Venter, H.S. e Oliver, M.S., Exploring Forensic Data with Self-Organizing Maps, **IFIP International Conference on Digital Forensics**, (2005), 113 – 123.
10. Beebe, N. L., Clark, J. G., Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results, **Digital Investigation**, Elsevier, 4:1 (2007), 49 – 54.
11. Hadjidj R., Debbabi, M., Lounis, H., Iqbal, F., Szporer, A., Benredjem, D., Towards an integrated e-mail forensic analysis framework, **Digital Investigation**, Elsevier, 5:3-4 (2009), 124 – 137.
12. Iqbal, F., Binsalleeh, H., Fung, B. C. M., Debbabi, M., Mining writeprints from anonymous e-mails for forensic investigation. **Digital Investigation**, Elsevier, 7:1-2 (2010), 56 – 64.
13. Decherchi, S., Tacconi, S., Redi, J., Leoncini, A., Sangiacomo, F., Zunino, R., Text Clustering for Digital Forensics Analysis, **Computational Intelligence in Security for Information Systems**, (2009), 29 – 36.
14. Stoffel, K., Cotofrei, P., Han, D., Fuzzy Methods for Forensic Data Analysis, **IEEE International Conference of Soft Computing and Pattern Recognition**, (2010), 23 – 28.
15. Vendramin, L., Campello, R. J. G. B., Hruschka, E. R., Relative clustering validity criteria: A comparative overview, **Statistical Analysis and Data Mining**, 3 (2010), 209 – 235.
16. Salton, G. e Buckley, C., Term weighting approaches in automatic text retrieval, **Information Processing and Management**, (1988).
17. Liu, L., Kang, J., Yu, J., e Wang, Z., A comparative study on unsupervised feature selection methods for text clustering, **IEEE International Conference on Natural Language Processing and Knowledge Engineering**, (2005), 597-601.
18. Levenshtein, V., Binary codes capable of correcting deletions, insertions, and reversals, **Soviet Physics Doklady** 10, (1966), 707–10.
19. Mirkin, B., Clustering for data mining: A data recovery approach, **Chapman and Hall**, (2005).
20. Fred, A. L. N., e Jain, A. K., Combining Multiple Clusterings Using Evidence Accumulation, **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 27-6, (2005), 835-850.
21. Hubert, L., e Arabie, P., Comparing partitions, **Journal of Classification**, 2 (1985), 193 – 218.