UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO LUIZ GRAVE GROSS

URSA: Um framework para agrupamento de dados e validação de resultados

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Leandro Krug Wives

Porto Alegre, novembro de 2014.

“If I have seen further than others,
it is by standing upon the shoulders of giants.“
— Isaac Newton

AGRADECIMENTOS

Gostaria de agradecer ao apoio que tive dos meus pais, sempre me motivando e me dando orientação a todo momento que era necessário, principalmente no desenvolvimento deste trabalho.

À minha namorada por sempre estar ao meu lado, me dando apoio nos momentos mais difíceis.

Aos meus colegas da graduação, que me acompanharam durante todo o tempo e que tornaram o transcorrer do curso uma experiência muito mais agradável e prazerosa.

Aos professores e professoras da graduação para com os quais nutri um sentimento de admiração pelo belo trabalho que exercem e pelo domínio que possuem dos conteúdos lecionados, que serviram de estímulo a seguir estudando computação e a gostar cada vez mais desta área do conhecimento.

Aos meus colegas de bolsa no GPPD, realizada em 2010, com os quais tive o primeiro contato no desenvolvimento de trabalhos científicos, com meu projeto de iniciação científica.

Aos meus colegas de bolsa no PET, realizada em 2010, que embora tenhamos trabalhado por pouco tempo juntos, foi suficiente para estabelecer bons laços de amizade, aprendizado e descontração.

Por fim gostaria de agradecer ao meu orientador pela paciência e auxílio que tem me dado desde o início do desenvolvimento deste trabalho, me guiando da melhor forma possível para solucionar todos os problemas com os quais me deparei.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
ELKI	Environment for Developing KDD-Applications Supported by Index-Structures
GPL	General Public License
GPPD	Grupo de Processamento Paralelo e Distribuído
GSM	Grau de Similaridade Mínima
PET	Programa de Educação Tutorial
SI	Sistemas da Informação
STING	Statistical Information Grid-based method
SQL	Structured Query Language
UFRGS	Universidade Federal do Rio Grande do Sul
WEKA	Waikato Environment for Knowledge Analysis

RESUMO

Devido ao avanço da tecnologia e da geração contínua de grandes volumes de dados, técnicas mais sofisticadas para extrair informações relevantes desses dados se mostraram necessárias. A técnica de agrupamento de dados (*clustering analysis*) tem como objetivo separar um conjunto de objetos em grupos, na qual seus elementos possuem características similares entre si. Dessa forma, é possível analisar cada grupo de modo que a compreensão de informações seja mais eficiente. O problema de realizar agrupamentos de dados foi abordado na monografia "Um *framework* para agrupamento de dados", de Guilherme Haag Ribacki, em janeiro de 2013. Logo, o presente trabalho tem como objetivo estender este *framework*, incluindo novas técnicas de similaridade, seleção, agrupamento e validação de resultados, além de ser mais flexível quanto aos tipos de dados suportados, incluindo suporte a arquivos de áudio e de vídeo, além do suporte a documentos de texto. Dentre os novos algoritmos de agrupamento estão as técnicas *K-Means*, *K-Medoids*, *DBSCAN* e aglomerativo hierárquico. Novas técnicas de similaridade e seleção para os novos tipos de dados também foram implementadas. Os algoritmos de agrupamento implementados são executados e seus resultados analisados pelos algoritmos de validação disponíveis no *framework*, tais como Medida-F (*F-Measure*), Silhueta (*Silhouette*), Puridade (*Purity*) e Entropia (*Entropy*). São utilizados diferentes conjuntos de dados para a validação dos algoritmos de agrupamento do *framework*, três conjuntos para arquivos de texto, sendo dois pequenos, com 12 (o mesmo utilizado por Ribacki (2013)) e 13 documentos, respectivamente, e um grande (*Reuters-21578 10 categories Apte' split*), com 1.248 documentos, um conjunto de áudio com 30 arquivos e um de vídeo com 21 arquivos. Para os conjuntos pequenos de arquivos de texto e para o conjunto de arquivos de áudio foi possível agrupar corretamente os dados ao se utilizar todos os algoritmos de agrupamento disponíveis no *framework*. Já o conjunto grande de arquivos de texto e o conjunto de arquivos de vídeo não foram agrupados corretamente. O primeiro devido a problemas inerentes ao próprio conjunto e o segundo devido a presença de *metadados* com relação fraca ao conteúdo dos arquivos do conjunto.

Palavras-chave: Agrupamento de Dados, Avaliação de Agrupamentos, Extração de Dados, Framework, Projeto Open Source.

URSA: A framework for data clustering and data analysis

ABSTRACT

Due to advancements in technology and the continuous generation of large volumes of data, more sophisticated techniques to extract relevant information from these data were necessary. The *clustering analysis* technique aims to separate a set of objects into groups, in which its elements have similar characteristics to each other. Thus, it is possible to analyze each group so that comprehension of information is more efficient. The problem of calculating data clusters has been addressed in the monograph "A *framework* for data clustering", written by Guilherme Haag Ribacki in January 2013. Therefore, the objective of this study is to extend this *framework*, including new techniques for similarity, selection, data clustering and cluster validity, and is intended to be more flexible in the types of supported data, including support for audio and video files, in addition to the support of text files. Among the new clustering algorithms implemented, we can highlight *K-Means*, *K-Medoids*, *DBSCAN* and an agglomerative hierarchical clustering algorithm. New techniques of similarity and selection techniques for the new types of data were also implemented. The implemented clustering algorithms are executed and their results analyzed by the validation algorithms available in the *framework*, such as *F-Measure*, Silhouette, Purity and Entropy. Different data sets for analysis are used to validate the clustering algorithms from the *framework*, three sets for text files, two of them small, with 12 (the same used by Ribacki (2013)) and 13 documents, respectively, and a large one (*Reuters-21578 10 categories Apte' split*), with 1,248 documents, a set for audio with 30 files and one for video with 21 files. For the small sets of text files and for the set of audio files, the clusters were identified correctly when executing all clustering algorithms available in the *framework*. For the large set of text files and for the set of video files the clusters were not identified correctly. The first one due to problems inherent to the set itself and the second one due to the presence of *metadata* weakly related to the contents of the files in the set.

Keywords: Data Clustering, Data Analysis, Data Extraction, Framework, Open Source Project.

1 INTRODUÇÃO

Este capítulo apresenta uma introdução ao trabalho. Esta monografia tem como objetivo estender o framework desenvolvido por Ribacki (2013), implementando novos algoritmos de agrupamento de dados, análise de resultados, e também permitir o suporte a arquivos de áudio e de vídeo, além do suporte atual a arquivos de texto. Nas seções seguintes são apresentadas a motivação, objetivo e a organização do texto deste trabalho.

1.1 Motivação

Utilizamos constantemente ferramentas e serviços que manipulam quantidades enormes de dados, seja através da Internet ou localmente. A facilidade com que se pode hoje adquirir componentes de hardware e a evolução das velocidades de conexão de Internet, bem como o aumento significativo das capacidades de armazenamento de computadores pessoais e servidores, possibilita que cada vez mais informações estejam disponíveis a pessoas e sistemas, e que estas as possam acessar. Devido a isso, ferramentas capazes de buscar informações precisas sobre determinado assunto tornam-se essenciais, seja para buscar textos, imagens ou mesmo arquivos de multimídia como áudio e vídeo.

A correta referência a documentos específicos dentro de um universo de documentos na Internet, por exemplo, não é uma tarefa trivial; pode-se ter acesso a vários dados, porém o efetivo acesso à informação desejada é um processo mais complexo, o que nos remete ao dilema elucidado por (ESCHRICH et al., 2003), "riqueza de dados, pobreza de conhecimento". Logo, para contornar esse dilema, uma forma de identificar características semânticas dos arquivos se faz necessária para a obtenção de resultados relevantes, quando da realização de uma busca de informações sobre determinado assunto.

A mineração de dados lida com tal problema. Nos últimos anos, aliada a novas estratégias de manipulação de banco de dados e à evolução da inteligência artificial, ela foi capaz de processar grandes coleções de dados brutos e, a partir das características de cada arquivo, descobrir informações escondidas, até então ignoradas (ZHENG, 2012).

Em face da necessidade de se processar grandes volumes de dados, geralmente utiliza-se o processo de agrupamento de dados (*clustering analysis*), constituído de diversas eta-

pas que combinadas separam o conjunto de dados em partições, ou grupos, e desses pode-se extrair informações pertinentes à aplicação desenvolvida. O agrupamento de dados é uma importante operação realizada na mineração de dados e pode ser utilizada como uma ferramenta *stand-alone*, que além de determinar os agrupamentos, também pode analisar a qualidade dos mesmos, através da análise de agrupamentos (ZHENG, 2012).

O Eurekha é uma ferramenta desenvolvida por Wives (1999) com o objetivo de identificar agrupamentos para coleções de arquivos de texto, que implementa técnicas de extração de dados (*data extraction*), técnicas para cálculo do grau de similaridade entre os arquivos e também diferentes algoritmos de agrupamento. Devido a gama de algoritmos de agrupamento de dados oferecida pela ferramenta, é possível comparar o desempenho de cada uma, determinando qual delas é mais adequada a cada conjunto de dados. Porém a ferramenta possui algumas limitações, como incompatibilidade com outros sistemas operacionais atuais, impossibilidade de extensões e suporte a apenas arquivos de texto.

Devido a tais limitações, Ribacki (2013) iniciou o desenvolvimento de um *framework* com características semelhantes à da ferramenta Eurekha, mas compatível com diferentes sistemas operacionais e com a possibilidade de extensão, mantendo ainda suporte a arquivos de texto. O *framework* se ateve à reestruturação do código, não implementando algoritmos diferentes daqueles previstos no Eurekha inicialmente. Ele também se mostrou mais versátil do que a ferramenta desenvolvida por Wives (1999), pelas melhorias implementadas e também por poder ser estendido ou incorporada a projetos desenvolvidos por terceiros que necessitem montar e analisar agrupamentos de arquivos.

1.2 Objetivo

Com base nas limitações destes projetos prévios, seja quanto ao suporte a apenas arquivos de texto, seja pela quantidade limitada de algoritmos de agrupamento de dados, o presente trabalho tem como objetivo estender o *framework* desenvolvido por Ribacki (2013), de modo que este possua suporte a arquivos de áudio e vídeo, além do suporte atual a arquivos de texto, e também implementar novas técnicas de agrupamento de dados e análise de resultados. Este trabalho também tem como objetivo comprovar o correto funcionamento dos algoritmos de agrupamento implementados, ao realizar testes com coleções de dados de referência, conhecidas e controladas, bem como avaliar a qualidade dos resultados apresentados aplicando as técnicas de validação de agrupamentos.

Como objetivo complementar se propõe disponibilizar o projeto na plataforma aberta e colaborativa *GitHub*[1], para que outros desenvolvedores possam utilizar o *framework*, bem como acompanhar o seu desenvolvimento e contribuir com novos recursos. Não menos importante, todo o projeto será *open source*, propiciando, portanto, seu reuso em outros projetos ou mesmo sua utilização plena como ferramenta de apoio.

[1] `https://github.com/`. Acessado em: 17/09/2014

1.3 Organização do texto

No próximo capítulo encontra-se o referencial teórico, no qual é exposto com detalhes o funcionamento de cada um dos algoritmos implementados, os tipos de dados suportados e suas particularidades, e também o processo de extração de informações, cálculo do grau de similaridade entre arquivos, agrupamento de dados e de avaliação de agrupamentos.

No capítulo terceiro é apresentada a estrutura do *framework*, suas classes e funcionalidades, e a relação entre estas. No capítulo quarto são apresentados os resultados obtidos pelas métricas de avaliação de agrupamentos, ao analisar-se os resultados de cada algoritmo de agrupamento para as coleções de dados de arquivos de texto, áudio e vídeo. Por fim, no capítulo quinto são expostas as conclusões obtidas no decorrer deste estudo.

2 REFERENCIAL TEÓRICO

Este capítulo se propõe a dar a explicação necessária para o entendimento do trabalho proposto. Aqui é apresentado o processo de agrupamento de dados, com as suas diversas etapas, que é o principal assunto de interesse desse trabalho. Também são apresentados todos os algoritmos de seleção de dados utilizados, bem como os algoritmos de cálculo de agrupamentos e as métricas de avaliação usadas para analisar a qualidade dos resultados (grupos) gerados pelos algoritmos de *clustering*.

2.1 Agrupamento de dados

Na área de *Sistemas da Informação (SI)*, a tarefa de *agrupamento*, também conhecida como *clustering*, recebe diferentes interpretações, cada qual definida segundo a sua aplicação. No âmbito das aplicações de alto desempenho, em aplicações de processamento paralelo e balanceamento de carga, por exemplo, *clustering* é considerada uma forma de conectar dois ou mais computadores de modo que eles se comportem como um só computador; já no contexto de banco de dados, *clustering* diz respeito a agrupar registros similares (ou de mesma chave) em regiões próximas, facilitando o acesso.

Essas definições porém não devem se confundir àquela que é de interesse a este trabalho. Dentro do estudo de mineração de dados (*data mining*), *clustering* é a divisão de dados em grupos contento objetos similares entre si (BERKHIN, 2006), buscando estabelecer classes significativas de dados (CAO et al., 2006).

As técnicas de *agrupamento* de dados são utilizadas em aplicações das mais diversas, dentre elas pode-se destacar a compressão de dados, geração e teste de hipóteses, predição baseada em grupos, marketing, biologia computacional, mineração de dados na Web, segmentação de imagens e muitas outras (BERKHIN, 2006) (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001). Tais aplicações se utilizam das técnicas de *clustering* pelo fato de que a quantidade de informações sobre os dados é muito pequena, e as suposições realizadas pelo usuário ou sistema devem ser minimizadas ou inexistirem.

Em muitas dessas aplicações as técnicas de *agrupamento* são utilizadas para auxiliar a *classificação*, logo deve-se ter o cuidado de não confundir *agrupamento* e *classificação*

(WIVES, 1999), dois conceitos distintos que serão mais profundamente abordados nas seções seguintes.

2.1.1 Definição

A técnica de agrupamento de dados, segundo Wives (1999), consiste em analisar uma série de objetos desorganizados quanto às suas características e, a partir destas, estabelecer entre eles associações ou correlações de modo a organizá-los em grupos de objetos similares. Esses grupos, também chamados de *aglomerados*, geralmente constituem uma *classe*, que possui um título mais genérico capaz de representar todos os elementos nela contidos, e frequentemente o termo *classe* é utilizado como um sinônimo aos termos *aglomerado*, *agrupamento* ou mesmo *grupo* (KOWALSKI, 1997). Logo, neste trabalho estes termos são usados indistintamente.

Essa técnica é bastante útil em casos onde não há a possibilidade de se alocar um especialista para a tarefa de agrupar os arquivos do conjunto de dados em classes ou também pelo simples fato desta tarefa ser inviável no caso de um enorme volume de dados ou mesmo por não haver um especialista disponível. O processo, portanto, ocorre de modo não supervisionado, sem que haja a necessidade de informações prévias a respeito do conjunto de dados.

O processo de agrupamento de dados, porém não se destina apenas à identificação de conglomerados, preocupando-se também com a análise dos resultados obtidos. Por isso ele pode ser denominado de *Análise de Conglomerados* (*cluster analysis*) (WIVES, 2004). Há também outras denominações encontradas na literatura que referem-se ao mesmo processo, são elas: *Análise de Classificação*, *Aprendizado não Supervisionado*, *Quantização de Vetores*, *Aprendizado por Observação* e *Análise de dados esparsos* (JAIN; MURTY; FLYNN, 1999).

É importante entender a diferença entre agrupamento de dados (*clustering*) e classificação (JAIN; MURTY; FLYNN, 1999). A classificação é um processo supervisionado com o objetivo de identificar à qual classe cada objeto pertence, ou seja, pressupõem que as classes sejam conhecidas e informadas previamente (RUBINOV; SOUKHOROKOVA; UGON, 2006). Fayyad (1996) apresenta outra definição, mais concisa, para o termo classificação, como o procedimento de atribuir elementos do conjunto de dados a um conjunto de categorias pré-definido. Por outro lado o processo de agrupamento é um processo não supervisionado, no qual não se possui conhecimento algum sobre as classes, sendo estas estabelecidas ao término no agrupamento.

Cabe aqui uma definição mais formal ao termo *aglomerado*, conforme expressado por Everitt (2011):

1. Um aglomerado é um conjunto de entidades similares, e entidades de diferentes aglomerados não são similares;

2. Um aglomerado é uma agregação de pontos no espaço tal que a distância entre dois pontos em um mesmo aglomerado é menor do que a distância entre qualquer ponto do aglomerado e qualquer ponto não pertencente a ele;

3. Aglomerados podem ser descritos como regiões conexas de um espaço multidimensional contendo uma densidade relativamente alta de pontos, separadas umas das outras por uma região contendo uma relativamente baixa densidade de pontos.

2.1.2 Etapas do processo de agrupamento

O processo típico de agrupamento de dados envolve os seguintes passos (JAIN; MURTY; FLYNN, 1999):

1. Identificação, seleção e/ou extração de características;

2. Cálculo de similaridades;

3. Identificação de aglomerados;

4. Abstração dos dados (caso necessário);

5. Validação de agrupamentos.

O primeiro passo é dividido em 3 subetapas. Inicialmente ocorre a identificação do tipo de dados e de suas características. Como já comentado, o *framework* desenvolvido neste trabalho se propõe a dar suporte a mais dois tipos de dados, a arquivos de áudio e a arquivos de vídeo, além dos já suportados documentos de texto, portanto, essa etapa é muito importante para o correto funcionamento da subetapa seguinte. Depois, as características que melhor representam o objeto são selecionadas e a estas atribuídas diferentes graus de relevância conforme sua importância. A relevância pode considerar apenas as características do próprio objeto ou também de toda coleção de objetos, mas isso é dependente do tipo de estratégia utilizada. Como resultado desta etapa, obtém-se listas de características de cada objeto.

O segundo passo recebe as listas de características obtidas no primeiro passo e calcula as similaridades entre os objetos. As similaridades são armazenadas em uma matriz de duas dimensões, que contém os valores de similaridades entre todos os objetos. A similaridade entre dois objetos será tão maior quanto forem a quantidade de características de que eles possuem em comum. Para cada tipo de dados uma estratégia de cálculo de similaridades específica é utilizada, por haver diferentes características a serem consideradas em cada tipo de dados.

No terceiro passo ocorre o agrupamento dos dados. Com base na matriz de similaridades calculada no passo anterior os objetos similares entre si, segundo algum critério da própria estratégia de agrupamento, são separados em grupos. Cada grupo, portanto,

possui objetos similares entre si e estes não são similares a objetos de outros grupos. Ao término desta etapa é concluído o agrupamento de dados propriamente dito, também conhecido por *data clustering*.

O quarto passo, abstração de dados, é o processo de extração de uma representação simples e compacta do conjunto de dados. Essa representação pode ser utilizada em análises automáticas para que máquinas possam realizar de forma eficiente processamentos mais aprofundados sobre o conjunto de dados, ou também para que humanos consigam visualizar de forma fácil e intuitiva mais informações dos objetos. Esta etapa não é obrigatória e não é utilizada neste trabalho.

Na etapa de validação de agrupamentos a qualidade dos *clusters* é verificada. A validação dos *clusters* pode ser realizada de três formas diferentes. A validação *externa* analisa os agrupamentos com base em uma informação de estrutura fornecida à aplicação, logo requer um conhecimento prévio das classes do conjunto de dados ou uma expectativa de resultados. A validação *interna* determina se os agrupamentos são intrinsecamente apropriados aos dados; não há necessidade de conhecimento prévio, a análise ocorre considerando os próprios objetos em cada grupo e suas características. E a validação *relativa* compara duas estruturas de agrupamento de dados, fornecidas pelo mesmo algoritmo de agrupamento, mas obtidas em execuções com parâmetros distintos.

O processo de agrupamento de dados utilizado neste trabalho é apresentado na Figura 2.1. Nele observam-se todas as etapas apresentadas por Jain (1999), à exceção da abstração de dados, que é opcional. Uma etapa adicional nessa imagem é a *interpretação*, que é uma etapa natural no processo de *cluster analysis*, visto que a partir do momento que os dados forem separados em grupos e analisados pelos algoritmos de validação de resultados, o sistema ou usuário irá interpretar os resultados, e por fim será gerado conhecimento sobre o conjunto de dados fornecido.

Cada uma das etapas do processo de agrupamento é independente das demais, exigindo somente que o resultado de uma etapa esteja correto para servir de entrada à próxima etapa. Devido a essa independência o processo apresenta certa modularidade, no sentido de que em cada etapa pode-se utilizar uma estratégia diferente sem que haja qualquer dependência dela nas demais etapas. A modularização das etapas é explorada neste trabalho, e permite ao *framework* flexibilidade quanto a quantidade e tipos de algoritmos desenvolvidos para cada uma. Essa abordagem permite, por exemplo, que um trabalho futuro seja desenvolvido focando em apenas uma das etapas do processo de agrupamento, permitindo que os algoritmos existentes sejam aprimorados e que novos sejam implementados, garantindo maior qualidade aos resultados fornecidos pelo *framework*.

2.1.2.1 Identificação, seleção e extração de características

De acordo com Wives (1999), as técnicas de agrupamento de dados são todas de alguma forma baseadas nas diferenças de similaridade entre os objetos. Assim, para

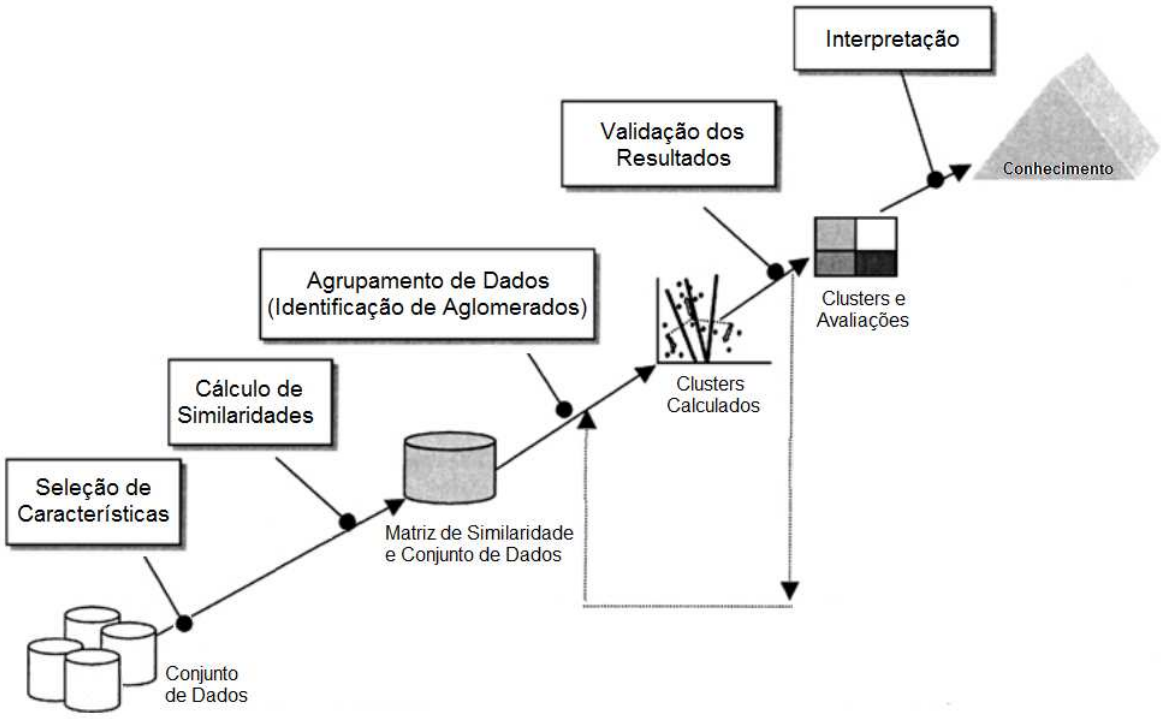

Figura 2.1: O processo de agrupamento de dados
Fonte: Adaptado de HALKIDI; BATISTAKIS; VAZIRGIANNIS 2001, p.2

que os dados sejam agrupados é necessário estabelecer o grau de similaridade entre eles. Porém a similaridade só pode ser obtida se houverem características que diferenciem esses objetos, a partir das quais eles poderão ser agrupados em diferentes grupos. O processo de seleção de características, segundo Jain (1999), inclui três subetapas:

1. Identificação de características;

2. Seleção de características;

3. Extração de características.

A identificação das características ocorre somente após a identificação do tipo de dados, pois apenas assim irá se saber quais as características disponíveis e de interesse. Os tipos de dados possíveis são arquivos de texto, de áudio ou de vídeo, cada um com seu conjunto particular de características.

A seleção de características é opcional e tem como responsabilidade filtrar as características de modo que as que restarem representem o objeto sem perda ou distorções de seu conteúdo. Elas é que irão determinar a obtenção de bons resultados ao calcular-se a matriz de similaridades na etapa seguinte. Para arquivos de texto, nesta etapa ocorre a eliminação de palavras irrelevantes, que não representam o conteúdo do objeto. Essas palavras indesejadas são chamadas de *palavras negativas* (*stop-words*). *Stop-words* são

palavras comuns à maioria dos textos e que auxiliam na construção de orações, como por exemplo, preposições, conjunções e artigos, as quais não agregam semântica ao objeto (MANNING; RAGHAVAN; SCHüTZE, 2008). Em alguns casos, o número de características pode ser grande demais, deixando o cálculo de similaridades lento. Para resolver esse problema uma estratégia adotada é selecionar apenas as características, ou palavras do texto, que sejam mais frequentes e eliminar as menos frequentes.

Para arquivos de texto a seleção de características envolve, além da eliminação das *stop-words*, um passo de atribuição de graus de importância a cada característica (termo ou palavra) selecionada. Isso ocorre, pois documentos de texto que tratam de um mesmo assunto, tendem a pertencer à mesma classe, por possuírem termos semelhantes que caracterizam este determinado assunto. Assim, ao estabelecer graus de importância às características mais frequentes do texto, também evidenciamos a classe à qual este arquivo pertence.

Para arquivos de áudio e de vídeo a seleção das características ocorre a partir dos *metadados* que cada objeto possui. Arquivos de áudio e arquivos de vídeo já possuem uma estrutura padrão de *metadados*, e, portanto, a seleção das características ocorre de forma ágil nesses dois casos, por já se saber quais as informações que serão coletadas.

A técnica mais comum de identificação das características (termos) marcantes, em arquivo de texto, é a *frequência relativa* (SALTON; MCGILL, 1983), que expressa o grau de importância de determinada palavra dentro do texto, de acordo com o seu número de ocorrências.

$$F_{rel}x = \frac{F_{abs}x}{N}$$

Figura 2.2: Equação da frequência relativa
Fonte: Wives (1999, p.23)

A *frequência relativa* (F_{rel}) de uma palavra x é calculada com base na *frequência absoluta* (F_{abs}) da palavra x no texto, ou seja, a quantidade de vezes que esta palavra aparece no texto, divido pelo número total de palavras selecionadas (N).

A última subetapa, de extração de características, também é opcional e é a ação de transformar uma ou mais características dos dados para produzir novas características relevantes (RIBACKI, 2013). Esta técnica é recomendada em processamentos muito demoradas, ou em aplicações que necessitem de resposta rápida, e que não permitam qualquer perda ou omissão das características dos objetos, assim, com base em um conjunto grande de características, um grupo menor de característica é criado, beneficiando o processamento e também a inspeção visual por humanos, quando aplicável (JAIN; MURTY; FLYNN, 1999).

2.1.2.2 Cálculo de similaridade

A similaridade entre todos os objetos é geralmente calculada utilizando-se uma função que analisa toda a lista de características dos objetos, obtida no primeiro passo do agrupamento de dados, e estabelece o grau de semelhança entre eles. Existem diversas estratégias para o cálculo do grau de semelhança entre objetos, mas as principais são representadas por *medidas de distância* e *medidas difusas* (*fuzzy*) (WIVES, 1999).

As medidas de distância normalmente consideram cada objeto como um ponto em um espaço euclidiano de n dimensões, onde n é o número de características máximas que um objeto pode possuir. Cada característica do objeto é representada por uma componente de coordenada e a similaridade é dada pela proximidade (distância) dos objetos neste espaço.

Algumas dessas funções são consideradas binárias, pois utilizam apenas dois valores; zero (0), caso a característica não esteja presente e um (1), caso contrário. Outras funções consideram o quanto uma característica influencia no objeto, ou seja, o quão discriminativa ela é, e levam em conta quantas vezes ela aparece nele, algumas vezes considerando ainda o número total de objetos em que a característica está presente. Nesse caso os valores são geralmente normalizados no intervalo [0, 1], onde zero (0) indica que a característica não tem relevância nenhuma para o objeto e um (1) indica que o objeto é totalmente dependente da característica (pois o caracteriza significativamente).

As medidas difusas, ou medidas *fuzzy*, utilizam a lógica difusa (*fuzzy*) para o cálculo das similaridades entre objetos. São várias as funções *fuzzy* existentes, mas a mais simples é a de inclusão simples (*set theoric inclusion*), que avalia a presença de características nos dois objetos sendo comparados. Se a característica estiver presente nos dois objetos, soma-se um (1) ao contador, caso contrário, soma-se zero (0). Ao final do processo, o grau de similaridade entre os objetos será um valor *fuzzy* no intervalo [0, 1], calculado pelo valor total do contador dividido pelo número total de características nos dois objetos (desconsiderando repetições).

Para o cálculo de similaridades entre arquivos de texto, áudio e vídeo, utiliza-se neste trabalho a estratégia *fuzzy*. Um exemplo de matriz de similaridades pode ser observado na Tabela 2.1. Os valores nela expressos representam o grau de similaridade entre dois objetos, ou seja, o quão semelhantes eles são. Também é importante observar que se um objeto x for 40% similar a um objeto y, então o objeto y também é 40% similar ao objeto x. Isso faz com que a matriz seja simétrica, logo, a metade inferior pode ser ignorada.

O cálculo de semelhanças tem como resultado uma matriz de similaridades de duas dimensões, como uma matriz semelhante à da Tabela 2.1. O cálculo é realizado em uma etapa separada e anterior à identificação de conglomerados. Essa estratégia reforça a noção de modularidade e independência entre as etapas do agrupamento de dados, além de evitar que uma sobrecarga de processamento por parte do algoritmo de agrupamento, por não haver necessidade de calcular as similaridades entre os objetos, já que isso é fornecido.

Tabela 2.1: Matriz de similaridades

	OBJ1	OBJ2	OBJ3	OBJ4	OBJ5
OBJ1	1.0	0.2	0.5	0.7	0.9
OBJ2	0.2	1.0	0.8	0.3	0.1
OBJ3	0.5	0.8	1.0	0.7	0.4
OBJ4	0.7	0.3	0.7	1.0	0.6
OBJ5	0.9	0.1	0.4	0.6	1.0

Neste trabalho são apenas consideradas estratégias para cálculo de similaridades entre objetos. Há também diversas formas de cálculo de similaridade entre grupos de objetos (EVERITT et al., 2011), porém estes algoritmos não são explorados aqui.

2.1.2.2.1 Similaridade em arquivos de texto

Para o cálculo de similaridades para arquivos de texto é utilizada a estratégia *fuzzy* neste trabalho. A equação utilizada para o cálculo das similaridades foi apresentada por Oliveira (1996) e implementada por Wives (1999), e pode ser observada na Figura 2.3.

$$gs(X,Y) = \frac{\sum_{h=1}^{k} gi * h(a,b)}{n}$$

Figura 2.3: Equação da média por operadores *fuzzy*
Fonte: Wives (1999, p.41)

Para esta equação tem-se que gs é o grau de similaridade entre os arquivos X e Y; gi é o grau de igualdade entre os pesos do termo h (peso a no arquivo X e peso b no arquivo Y); h é um índice para os termos comuns aos dois arquivos; k é o número total de termos comuns aos dois arquivos; e n é o número total de termos nos dois arquivos (sem repetição de termos). Seu funcionamento é simples: um contador vai acumulando pontos toda vez que um termo é encontrado em dois objetos que estão sendo comparados. Aliada a esta equação, outra equação é necessária, pois apesar de um determinado termo aparecer em dois arquivos de texto diferentes, eles podem ter níveis de importância diferentes em cada documento. Essa equação está presente na Figura 2.4.

$$gi(a,b) = \frac{1}{2}[(a \rightarrow b) \wedge (b \rightarrow a) + (\overline{a} \rightarrow \overline{b}) \wedge (\overline{b} \rightarrow \overline{a})]$$

Figura 2.4: Equação para o cálculo do grau de igualdade entre pesos
Fonte: Wives (1999, p.41)

Na equação da Figura 2.4, onde $\overline{x} = x - 1$, $(a \rightarrow b) = max\{c \in [0,1] | ac \leq b\}$ e

$\wedge = min$. Segundo ela, se um termo estiver presente nos dois arquivos de texto comparados, mas os pesos forem muito diferentes, o grau de igualdade torna-se baixo. Isso também é válido caso um arquivo possua um termo e outro arquivo não, pois esse termo irá contribuir com grau de igualdade zero ao somatório total, diminuindo a semelhança.

É importante observar que caso dois arquivos possuam 100% de semelhança isso não quer dizer que os arquivos são iguais, mas apenas que possuem as mesmas palavras (e na mesma frequência) depois que as palavras negativas foram removidas. Por fim, após serem realizados todos os cálculos entre os arquivos do conjunto de dados, obtém-se a matriz de similaridades.

2.1.2.3 Identificação de aglomerados

As técnicas de agrupamento de dados podem ser separadas em quatro grandes grupos: agrupamento por partições totais, agrupamento hierárquico, agrupamento baseado em densidade (*density-based clustering*) e agrupamento baseado em grades ou redes (*grid-based clustering*) (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001). Todas essas classes de algoritmos são explicadas com mais detalhes nas subseções 2.1.2.3.1, 2.1.2.3.2, 2.1.2.3.3 e 2.1.2.3.4, respectivamente.

Segundo Jain (1999), pode-se ainda classificar os algoritmos em diversas outras categorias, conforme as características de cada um: aglomerativos ou divisivos, monotéticos ou politéticos, fortes ou difusos, determinísticos ou estocásticos, incrementais ou não incrementais.

A classificação dos algoritmos como aglomerativos ou divisivos diz respeito à estrutura do algoritmo e sua operação. Os algoritmos aglomerativos consideram inicialmente cada objeto como um grupo de um único elemento e sucessivamente une grupos até que um critério de parada seja satisfeito. Os algoritmos divisivos, por outro lado, começam com todos os objetos em um mesmo grupo e este e seus subgrupos são divididos em grupos menores até que um critério de parada seja atingido.

Algoritmos monotéticos ou politéticos dizem respeito ao uso sequencial ou simultâneo das características dos objetos para o processo de agrupamento. A maioria dos algoritmos são politéticos, ou seja, todas as características são usadas simultaneamente no cálculo das distâncias entre os objetos e as decisões são tomadas com base nessas distâncias. Já algoritmos monotéticos consideram as características sequencialmente para dividir os objetos em grupos.

Um algoritmo também pode ser rígido (forte) ou difuso (*fuzzy*). Um algoritmo de agrupamento rígido (ou forte) aloca cada objeto em apenas um grupo, ou seja, não há repetição dos objetos em grupos distintos. Um algoritmo de agrupamento (*fuzzy*) estabelece graus de associação de cada objeto com vários grupos. Um algoritmo (*fuzzy*) pode ser convertido para um algoritmo rígido ao considerar que cada objeto esteja inserido no grupo com o qual possua o maior grau de associação (cada objeto em apenas um grupo).

A maioria dos algoritmos é determinístico, porém alguns podem ser estocásticos, isto é, podem utilizar métodos probabilísticos para otimizar os resultados em um processo iterativo. Algoritmos estocásticos podem ser utilizados, por exemplo, em abordagens que utilizam partições projetados para otimizar funções de erro quadrático.

Por fim, a última classificação proposta por Jain (1999) é a de algoritmos incrementais e não incrementais. Um algoritmo é incremental se questões como tempo de execução e memória são levados em consideração, o que geralmente ocorre para o agrupamento de um grande volume de dados. Inicialmente não existem muitos exemplos de algoritmos incrementais, porém a mineração de dados incentivou o desenvolvimento de algoritmos de agrupamento que minimizam o número de passadas pelo conjunto de objetos, reduzem o número de objetos examinados durante a execução ou reduzem o tamanho de estruturas de dados utilizadas pelas operações do algoritmo.

2.1.2.3.1 Agrupamento por partição total

Um algoritmo de agrupamento por partição total obtém uma única partição de objetos ao invés de uma estrutura de agrupamento, como os *dendrogramas*[1] gerados pelas técnicas hierárquicas. A técnica gera grupos distintos sem relação entre eles. Os objetos são agrupados de forma tal que todos os elementos de um mesmo aglomerado possuem um grau mínimo de semelhança, que é indicado pelo número de características que possuem em comum.

Apesar de constituir grupos distintos, os objetos podem estar em um ou mais grupos. Quando os objetos são atribuídos a um único grupo diz-se que o processo é *disjunto*. Caso um objeto seja atribuído a mais de um grupo, devido às semelhanças que suas características possuem com as características de objetos inseridos em outros grupos, diz-se que o processo não é disjunto. Embora um arquivo possa estar em mais de um grupo, os aglomerados são completamente isolados entre si.

Uma das principais desvantagens dessa técnica, é que pela ausência de uma estrutura de agrupamento, como o dendrograma, o usuário fica impossibilitado de observar possíveis relações entre grupos distintos, como por exemplo, assuntos mais abrangentes no caso dos arquivos de texto. Porém, uma das vantagens é que a execução é mais ágil do que os algoritmos hierárquicos, pois o cálculo dos aglomerados é executado em apenas um passo, enquanto os algoritmos hierárquicos necessitam de várias iterações até que se atinja o critério de parada.

[1] Um dendrograma é um tipo específico de diagrama ou representação icônica que organiza determinados fatores ou variáveis em uma estrutura de árvore. Para algoritmos de agrupamento hierárquicos um dendrograma apresenta as uniões ou divisões de grupos de objetos em níveis, sendo cada nível uma etapa de processamento do algoritmo.

2.1.2.3.2 Agrupamento hierárquico

Um agrupamento hierárquico é uma sequência de partições nas quais cada partição está aninhada com a próxima partição da sequência. Um algoritmo aglomerativo para agrupamento hierárquico começa com o agrupamento disjunto, colocando cada um dos n objetos em um *cluster* individual. O algoritmo de agrupamento utilizado diz como a matriz de similaridades deverá ser interpretada para unir dois ou mais *clusters* triviais, aninhando o agrupamento trivial em uma segunda partição. O processo se repete para formar uma sequência de agrupamentos aninhados nos quais o número de aglomerados diminui à medida que a sequência progride, até que apenas um aglomerado possua todos o n objetos. Um algoritmo *divisivo* realiza esse processo na ordem reversa (JAIN; DUBES, 1988).

O resultado do algoritmo é uma árvore de agrupamentos, chamada de dendrograma, que mostra como os grupos estão relacionados. Cortando o dendrograma em um nível desejado, o agrupamento dos dados em grupos disjuntos é obtido (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001). Um exemplo de dendrograma é mostrado na Figura 2.5. Nele pode-se observar que a medida que o grau de similaridade é incrementado (da direita para a esquerda) os grupos são unidos, formando aglomerados cada vez maiores, até que se atinja um determinado ponto no qual todos os objetos estão inseridos em um único aglomerado. Essa abordagem consiste em uma estratégia aglomerativa hierárquica, ou seja, os grupos são unidos à medida que o grau de similaridade, considerado pelo algoritmo para unir os grupos, sobe. Por outro lado, uma estratégia divisiva hierárquica considera um grau de similaridade cada vez menor (olhando da esquerda para a direita), e a medida que isso acontece os grupos são divididos até que cada um dos objetos esteja em um grupo independente.

Também pode-se observar que uma espécie de "histórico" de associações entre os objetos fica registrada no dendrograma, nas suas diversas partições. Assim é possível, por exemplo, que o sistema ou seu usuário consigam estabelecer relações mais completas entre os objetos, o que não ocorre na técnica de agrupamento por partição total, onde esse "histórico"não existe, por haver apenas uma partição dos agrupamentos.

Logo, a vantagem dessa técnica está justamente na possibilidade de visualizar relações mais completas entre os objetos do conjunto de dados, com a desvantagem de demandar mais processamento e consequentemente mais tempo de execução. Mas, se o tempo de execução não for um fator determinante para a escolha do algoritmo de agrupamento, esta é uma técnica a se considerar.

No trabalho proposto não é realizada a construção de um dendrograma, pois o interesse primário norteou os resultados finais e não análises mais apuradas sobre os relacionamentos entre os objetos do conjunto de dados. Esse tópico pode ser abordado em trabalhos futuros, juntamente com ferramentas de visualização para auxiliar na etapa de interpretação dos resultados.

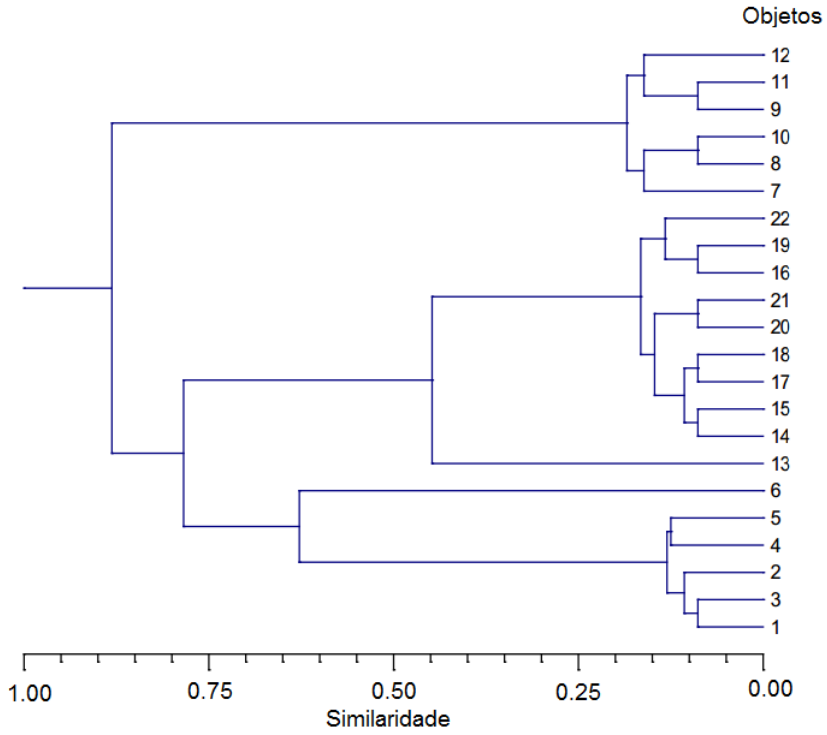

Figura 2.5: Dendrograma
Fonte: Adaptado de (NCSS, 2014)

2.1.2.3.3 Agrupamento baseado em densidade

A ideia chave desse tipo de agrupamento de dados é agrupar objetos vizinhos de um grupo de objetos em um *cluster* baseado em condições de densidade (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001). Esses algoritmos tipicamente consideram como aglomerados regiões densas de objetos no espaço de objetos, que estão separados por regiões de baixa densidade.

Um algoritmo bastante conhecido nesta categoria é o *DBSCAN* (ESTER et al., 1996). O funcionamento do *DBSCAN* baseia-se na ideia de que para cada objeto em um aglomerado, a vizinhança deve possuir uma quantidade mínima de objetos dentro de um dado raio. O *DBSCAN* pode lidar com ruído (*outliers*) e descobrir aglomerados de formato arbitrário. Além disso, o conceito de funcionamento do *DBSCAN* é usado em algoritmos de agrupamento incremental, ou seja, algoritmos nos quais os objetos são adicionados aos grupos, um a um, sem que o grupo seja comprometido. Isso é possível devido a sua natureza baseada em densidade, a inserção ou remoção de um objeto no *cluster* afeta somente a vizinhança do objeto, mantendo o restante do *cluster* intacto (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001).

Devido a essa característica do algoritmo *DBSCAN* possibilitar um agrupamento incremental de objetos no aglomerado, *clusters* como os apresentados na Figura 2.6 são possíveis. Observa-se que o aglomerado da direita apresenta um formato incomum, o

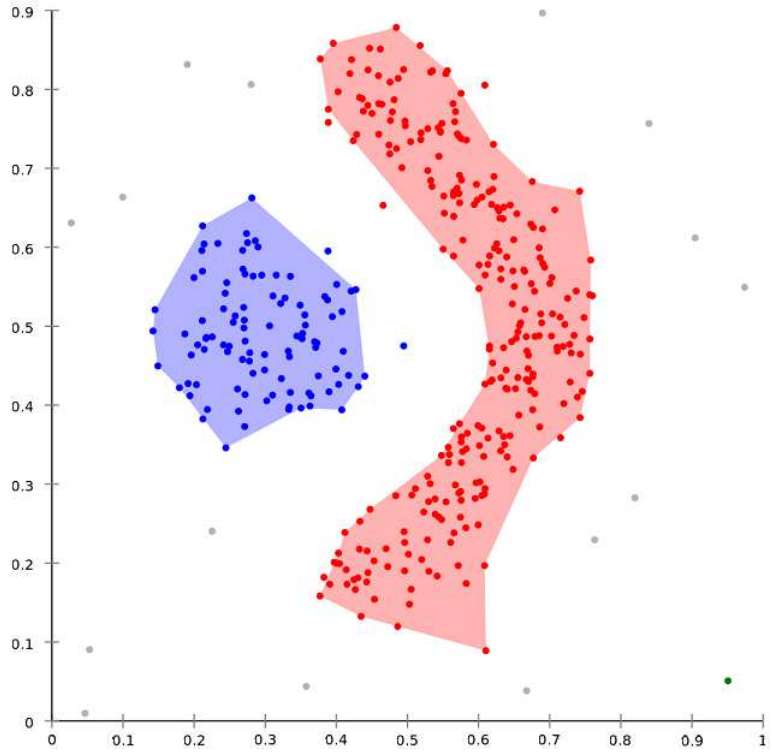

Figura 2.6: Agrupamentos realizados pelo algoritmo *DBSCAN*
Fonte: (WIKIPEDIA, 2014)

que não pode ser obtido em algoritmos de partição total ou hierárquicos. Neste *cluster*, mesmo que os pontos (objetos) dos extremos superior e inferior não sejam similares, eles pertencem ao mesmo grupo por estarem na mesma zona de densidade.

Segundo alguns autores, como Kriegal (2011), os algoritmos baseados em densidade obtêm como resultado *clusters* considerados "*clusters* naturais", pois eles são particularmente adequados para algumas aplicações inspiradas na natureza. Por exemplo, em conjuntos de dados espaciais, grupos de pontos (objetos) no espaço podem se formar ao longo de estruturas naturais como rios, estradas, falhas sísmicas, cadeias de montanhas, etc. Outra motivação natural também pode ser encontrada na zoologia. As espécies divergem umas das outras, porém ocorre frequentemente que muitas possuem um ancestral em comum. Assim, os zoologistas definem famílias de diferentes espécies, onde os membros não exibem necessariamente uma extraordinária quantidade de semelhanças com todos os membros, mas possuem uma certa semelhança com pelo menos alguns membros da família.

2.1.2.3.4 Agrupamento baseado em grade

Este tipo de algoritmo é utilizado principalmente para mineração de dados espacial. Sua principal característica é que eles quantizam o espaço em um número finito de células e então realizam todas as operações nesse espaço quantizado.

Um dos algoritmos mais representativos dessa categoria é o *STING*. Primeiro ele divide a área espacial em células retangulares usando uma estrutura hierárquica e depois computa parâmetros estatísticos como média, variância, mínimo e máximo para cada característica dos objetos presentes nas células. Então é gerada uma estrutura hierárquica da rede de células para representar as informações agrupadas em diferentes níveis. Baseado nessas estruturas, o *STING* utiliza as informações agrupadas para buscar por relacionamentos entre os objetos ou para atribuir de forma eficiente um novo objeto aos agrupamentos (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001).

Esse tipo de algoritmo de agrupamento de dados não será abordado neste trabalho.

2.1.2.4 Validação de agrupamentos

A última etapa do agrupamento de dados é a validação de agrupamentos, em inglês, *cluster validity*. Nesta etapa os agrupamentos calculados na etapa anterior, de agrupamento de dados, são avaliados quanto a sua qualidade, levando-se em consideração diversas métricas de avaliação.

É perceptível que um problema recorrente no processo de agrupamento de dados é, por exemplo, determinar de forma ótima quantos grupos são necessários para agrupar um conjunto de dados (caso do algoritmo *K-Means*, por exemplo). Na maioria dos experimentos disponíveis para consulta, conjuntos de duas dimensões são utilizados para que o leitor seja capaz de visualmente verificar a eficácia de determinado algoritmo de agrupamento. Porém quando se trabalha com conjuntos de dados multimensionais (com mais de três dimensões), a visualização dos resultados começa a se tornar uma tarefa difícil. Logo, é necessária uma forma diferente de avaliar a qualidade dos resultados.

Segundo Haldiki (2002), a maioria dos algoritmos de agrupamento de dados comportam-se de forma diferente, dependendo das características do conjunto de dados, dos parâmetros utilizados e das primeiras suposições realizadas, assim é necessária alguma forma de avaliação dos resultados para validá-los. Os algoritmos de validação de agrupamentos podem ser classificados em três grandes classes, sendo as duas principais baseadas em critérios *externos* e critérios *internos*, e a terceira baseada em um critério *relativo* (THEODORIDIS; KOUTROUMBAS, 2002).

O critério *externo* avalia os resultados de um algoritmo de agrupamento baseado em uma estrutura pré-especificada, que nada mais é do que uma expectativa de resultados de agrupamentos para o conjunto de dados (TAN; STEINBACH; KUMAR, 2006).

O critério *interno* avalia os agrupamentos com base nas próprias características dos objetos agrupados, como a matriz de similaridades e as listas de características dos objetos, por exemplo.

Já o critério *relativo* compara duas estruturas de agrupamento de dados, fornecidas pelo mesmo algoritmo de agrupamento, porém com parâmetros diferentes.

É importante ressaltar que os métodos apresentados indicam a qualidade dos resul-

tados obtidos, logo eles devem apenas ser utilizados como uma ferramenta à disposição dos especialistas para avaliar os aglomerados resultantes. Mas isso não impede que seja realizado, por exemplo, um processo de automático e iterativo entre as etapas de agrupamento de dados e validação, ocorrendo da seguinte forma: os aglomerados são calculados e avaliados; se estiverem dentro dos índices esperados pela avaliação o processo termina, caso contrário é realizado novo cálculo de agrupamentos com outros parâmetros. O processo se repete até um determinado critério de parada. Isso pode ser implementado em uma tentativa de conseguir ainda melhor resultados para que os especialistas os possam analisar.

2.2 Algoritmos de agrupamento

O objetivo dessa seção é apresentar com mais detalhes todos os algoritmos implementados no *framework* proposto neste trabalho. Todos os algoritmos que serão apresentados aqui respeitam o processo de agrupamento, ou seja, são dependentes de dados fornecidos pelas etapas anteriores. A matriz de similaridades é o principal parâmetro que todas as estratégias utilizam para calcular os agrupamentos que se adequam ao conjunto de dados.

Devido a gama de algoritmos implementados, e suas particularidades, alguns deles necessitam de outros parâmetros, além da matriz de similaridades. Os algoritmos *Cliques*, *Single Link*, *Stars*, *Best-Star*, *Full-Stars* e *Strings*, pertencentes à classe de algoritmos por partição total, consideram um fator mínimo de aceitação (*threshold*), também conhecido como *Grau de Similaridade Mínima* (GSM), onde são ditos similares os objetos com grau de similaridade maior ou igual a este número. O GSM é passado como parâmetro no início da execução dessas estratégias.

Os algoritmos *Single Link* e *Strings* não foram implementados no *framework* devido a sua simplicidade e também por haver outras técnicas consagradas que foram priorizadas.

Os algoritmos *K-Médias* (*K-Means*) e *K-Medoids* também são pertencentes à classe de algoritmos por partição total, porém não utilizam o GSM como parâmetro. Esses dois algoritmos são baseados em minimização de erros quadráticos e a eles é informado o número de clusters (k) que o algoritmo deve retornar.

Já os algoritmos *DBSCAN* e *Aglomerativo Hierárquico* pertencem a diferentes classes de algoritmos. O *DBSCAN* é um algoritmo baseado em densidade que recebe como parâmetros um valor de *épsilon* (ϵ), que deve ser entendido como a máxima diferença de similaridade permitida entre dois objetos para que estes sejam de um mesmo grupo, e o número mínimo de objetos necessários para que um grupo seja formado. Enquanto o *Aglomerativo Hierárquico* pertence à classe de algoritmos hierárquicos. Ele utiliza dois parâmetros, o primeiro para estabelecer a similaridade mínimo necessária para que dois grupos sejam unidos e o segundo parâmetro é o critério de parada.

2.2.1 Cliques

Todos os objetos são comparados uns com os outros e atribuídos ao grupo onde os objetos são semelhantes entre si. Um objeto será semelhante a outro se o valor obtido na matriz de similaridades for igual ou superior ao GSM estabelecido. Este algoritmo possui uma execução mais demorada, pois todos os objetos são comparados entre si.

1. Selecionar um objeto que não faz parte de nenhum grupo e adicioná-lo a um novo grupo;
2. Selecionar um novo objeto e compará-los;
3. Se o objeto selecionado for similar a todos os objetos do grupo, adiciona-o ao grupo;
4. Enquanto houver objetos para se comparar, voltar ao passo 2;
5. Enquanto houver objetos não adicionados a nenhum grupo, voltar para o passo 1.

Este algoritmo pode ser ou não disjunto, dependendo da forma como a implementação é realizada, ou seja, um mesmo objeto pode estar em apenas um grupo (disjunto) ou em vários grupos (não disjunto).

Um exemplo de execução de uma versão disjunta, considerando-se a Tabela 2.1 e GSM igual a 0.6 é mostrado a seguir:

Grupo 1 : OBJ1, OBJ4, OBJ5;

Grupo 2 : OBJ2, OBJ3;

2.2.2 Single Link

O agrupamento *Single Link* necessita que pelo menos um dos objetos de um grupo seja similar a um novo objeto para que este seja adicionado ao mesmo grupo. O algoritmo é como segue:

1. Selecionar um objeto que não faz parte de nenhum grupo e adicioná-lo a um novo grupo;
2. Inserir neste grupo todos os objetos similares a ele;
3. Para cada um dos objetos inseridos no grupo repetir o passo 2;
4. Quando não houver mais inserções de elementos no passo 2, ir para o passo 1;
5. O algoritmo termina quando todos os objetos estiverem inseridos em um grupo.

Da mesma forma como comentado na seção 2.2.1 este algoritmo pode ou não ser disjunto (cada objeto em apenas um grupo) dependendo da implementação. A seguir segue o resultado de execução deste algoritmo, na sua versão disjunta, utilizando-se a Tabela 2.1 como matriz de similaridades e GSM 0.6.

Grupo 1 : OBJ1, OBJ4, OBJ5, OBJ3;

Grupo 2 : OBJ2.

2.2.3 Stars

No algoritmo *Stars*, seleciona-se um objeto para formar um novo grupo e todos os novos objetos devem ser similares a este primeiro. Assim, é criado um grafo, com um elemento central e os demais conectados a este centro, formando uma figura muito parecida com uma estrela (por isso o nome *Stars*). O algoritmo é como segue:

1. Selecionar um objeto que não faz parte de nenhum grupo e adicioná-lo a um novo grupo;
2. Inserir no mesmo grupo todos os objetos similares a ele;
3. Se ainda houver objetos sem grupo, repetir passos 1 e 2.

Aplicando este algoritmo à Tabela 2.1 e utilizando-se GSM igual a 0.6, obtém-se:

Grupo 1 : OBJ1, OBJ4, OBJ5;

Grupo 2 : OBJ2, OBJ3.

2.2.4 Best-Star

Este algoritmo foi desenvolvido por Wives (1999) e contorna um problema presente no algoritmo *Stars* em que um objeto é atribuído ao primeiro grupo onde possuir similaridade maior do que o GSM. Na abordagem do *Best-Star*, um objeto é atribuído ao grupo com o qual possuir a maior similaridade, maximizando, portanto, a afinidade entre os objetos dos grupos, e logo, mais adequado. O algoritmo do *Best-Star* é como segue:

1. Selecionar um objeto que não faz parte de nenhum grupo e adicioná-lo a um novo grupo;
2. Selecionar um novo objeto e adicioná-lo ao grupo com o qual possuir a maior similaridade;
3. Os objetos que não são centros de grupos são automaticamente realocados para o grupo com o qual possuir a maior similaridade;

4. Executar os passos 1, 2 e 3 até que todos os objetos estejam inseridos em algum grupo.

Aplicando este algoritmo à Tabela 2.1 e utilizando-se GSM igual a 0.5, obtém-se:

Grupo 1 : OBJ1, OBJ4, OBJ5;

Grupo 2 : OBJ2, OBJ3.

Se o GSM fosse alterado para 0.8, teríamos o seguinte resultado de execução:

Grupo 1 : OBJ1, OBJ5;

Grupo 2 : OBJ2, OBJ3;

Grupo 3 : OBJ4.

2.2.5 Full-Stars

O *Full-Stars* funciona de forma análoga ao *Best-Star* e foi desenvolvido com o mesmo propósito do *Best-Star*, ou seja, contornar os problemas inerentes ao algoritmo *Stars*. Porém, a diferença desse algoritmo é que ele não é disjunto, admitindo que um objeto seja inserido em todos os grupos com os quais seja similar. O algoritmo é como segue:

1. Selecionar um objeto que não faz parte de nenhum grupo e adicioná-lo a um novo grupo;
2. Selecionar um novo objeto e adicioná-lo a todos os grupos com os quais for similar;
3. Executar os passos 1 e 2 até que todos os objetos estejam inseridos em pelo menos um grupo.

Aplicando este algoritmo à Tabela 2.1 e utilizando-se GSM igual a 0.5, obtém-se:

Grupo 1 : OBJ1, OBJ3, OBJ4, OBJ5;

Grupo 2 : OBJ2, OBJ3.

2.2.6 Strings

Este algoritmo baseia-se na ideia de construir vetores de objetos similares, como em um vetor de caracteres (*string*), criando uma cadeia de objetos justapostos e similares entre si. Tem-se, portanto, o objeto A similar ao objeto B, o objeto B similar ao objeto C, e assim sucessivamente. O algoritmo é como segue:

1. Selecionar um objeto que não faz parte de nenhum grupo e adicioná-lo a um novo grupo. Este objeto é identificado como *NODO*;

2. Selecionar um novo objeto e adicioná-lo ao primeiro grupo no qual ele seja similar ao *NODO* do grupo.

3. O novo objeto torna-se o *NODO* do grupo;

4. Repetir os passos 1, 2 e 3 até que todos os objetos estejam em algum grupo.

Aplicando este algoritmo à Tabela 2.1 e utilizando-se GSM igual a 0.5, obtém-se:

Grupo 1 : OBJ1, OBJ3, OBJ2;

Grupo 2 : OBJ4, OBJ5;

2.2.7 Aglomerativo Hierárquico

O algoritmo *Aglomerativo Hierárquico* funciona baseado na ideia de junção de grupos. Cada objeto inicialmente é alocado a um grupo distinto. À medida que o algoritmo realiza iterações, o grau de similaridade necessário para unir grupos é diminuído gradativamente, até chegar em 0.0, onde há apenas um grupo com todos os objetos. Como já comentado anteriormente, todas as etapas do algoritmo podem ser visualizadas em um dendrograma para melhor análise. O algoritmo é como segue:

1. Inicializar cada objeto em um grupo distinto;

2. Encontrar dois grupos xi e xj similares segundo o grau de similaridade corrente;

3. Unir os grupos xi e xj, criando um novo grupo;

4. Realizar passos 2 e 3 até que não se possa mais unir grupos.

5. Atualizar grau de similaridade.

6. Realizar passos 4 e 5 até que o grau de similaridade seja zero.

2.2.8 K-Médias

O algoritmo *K-Médias*, também chamado de *K-Means*, requer que seja estabelecido o número k de grupos que deseja-se obter, bem como a matriz de similaridades. Este algoritmo funciona com a ideia de que cada grupo possui um centro, calculado com base nas similaridades entre os objetos desse mesmo grupo. A cada novo objeto inserido no grupo, um novo centro é calculado. O cálculo do novo centro, ou centroide, é realizado com base na média das características dos objetos do grupo. Novos objetos analisam suas semelhanças com o centroide de cada grupo e são inseridos naquele em que a similaridade for maior.

O algoritmo da estratégia de agrupamento *K-Média* é como segue:

1. Selecionar k objetos. Cada um será inicialmente o centroide de um grupo;
2. Selecionar um objeto não agrupado e adicioná-lo no grupo com a maior similaridade;
3. Calcular o novo centroide assim que um novo objeto for adicionado ao grupo;
4. Repetir os passos 2 e 3 até que todos os objetos estejam em algum grupo.

Como pode-se observar o algoritmo realiza o passo 1 uma única vez. Devido a isso o algoritmo *K-Médias* é extremamente dependente da escolha dos primeiros centroides para obter bons resultados. Logo, caso em uma execução os resultados não forem tão bons quanto os esperados, é possível realizar nova execução com os primeiros k centroides diferentes. Isso irá propiciar novos resultados e não raro, melhores resultados poderão ser obtidos.

2.2.9 K-Medoids

O algoritmo *K-Medoids* funciona de modo semelhante ao *K-Médias*, porém o cálculo do centroide de cada grupo é realizado de modo diferente. Dentre os objetos do grupo um dos objetos é eleito o centroide, ao invés de ser calculado um centro para o grupo. Desse modo uma operação que afeta consideravelmente os resultados obtidos com o algoritmo *K-Medoids* é a escolha dos novos centroides a cada iteração, pois dependendo da escolha, diferentes objetos são inseridos em cada grupo, proporcionando resultados finais diferentes.

Outro fator que influi neste algoritmo, assim como no *K-Médias* é a escolha dos primeiros k centroides. Uma boa estratégia de escolha trará melhores resultados finais, assim como elucidado na explicação do algoritmo *K-Médias*.

O algoritmo da estratégia de agrupamento *K-Medoids* é como segue:

1. Selecionar k objetos. Cada um será inicialmente o centroide de um grupo;
2. Selecionar um objeto não agrupado e adicioná-lo no grupo com a maior similaridade;
3. Eleger entre os objetos do grupo um novo centroide;
4. Repetir os passos 2 e 3 até que todos os objetos estejam em algum grupo.

2.2.10 DBSCAN

O *DBSCAN* é um algoritmo baseado em densidade que utiliza a ideia de vizinhança-épsilon para seu funcionamento. Dado um objeto A, estão em sua vizinhança-épsilon todos os objetos com similaridade maior ou igual a 1.0 menos épsilon. O objeto A e os objetos de sua vizinhança são adicionados ao mesmo grupo. Cada um dos objetos da

vizinhança de A também calculam a sua vizinhança-épsilon e estes novos objetos também são inseridos no mesmo grupo. O processo se repete até que todos os objetos estejam em um grupo.

Considera-se um *objeto núcleo* (*core object*) se ele possui um número mínimo de objetos na sua vizinhança-épsilon (valor fornecido como parâmetro ao executar o programa). Esse objeto e seus vizinhos podem formar um novo grupo. Objetos que não possuem uma vizinhança-épsilon mínima, mas que estão em um grupo são chamados de *objetos de fronteira* (*border objects*). Aqueles que não se classificam em nenhuma das duas definições são objetos *ruído* (*noise*). Vale notar que *objetos de fronteira* também possuem vizinhança-épsilon, mas geralmente possuem significativamente menos vizinhos do que *objetos núcleo*.

O algoritmo pode ser transcrito nos passos que seguem (ESTER et al., 1996):

1. Selecionar um objeto não visitado e encontrar sua vizinhança-épsilon;
2. Se o número de vizinhos é maior ou igual ao número mínimo, marcar o objeto como núcleo e inserir o objeto e objetos vizinhos em um novo grupo;
3. Remover rótulo de "ruído" em objetos inseridos no grupo;
4. Rotular objetos com vizinhança-épsilon inferior mínimo como "ruído";
5. Havendo grupos com algum objeto em comum eles são unidos;
6. Repetir passo 1 até que todos os objetos tenham sido visitados.

Ao *DBSCAN* são fornecidos o número mínimo de objetos da vizinhança-épsilon, o valor de épsilon e a matriz de similaridades. A quantidade de grupos é estabelecida pelo próprio algoritmo, à medida que os grupos são formados. Outra vantagem é a formação de grupos de formato arbitrário, como mostrado na Figura 2.6.

2.3 Métricas de avaliação

Nesta seção são apresentadas as métricas de avaliação utilizadas pelo *framework* na etapa de validação dos resultados. Utilizam-se cinco (5) estratégias baseadas em critérios externos, precisão, revocação, *medida-F*, puridade e entropia, e uma estratégia baseada em critérios internos, o coeficiente silhueta.

2.3.1 Precisão, Revocação e *Medida-F*

Segundo Tan (2006), duas métricas de avaliação comumente utilizadas são precisão (*precision*) e revocação (*recall*). Elas são utilizadas em casos onde a correta identificação de uma das classes é considerada mais importante do que detectar todas. Ambas são métricas supervisionadas, bem como a *medida-F*.

A precisão determina a fração de objetos que se mostram pertencentes a uma classe declarada. Quanto maior a precisão, menor será o número de objetos "falsos positivos" (objetos que não pertencem a uma determinada classe, porém foram alocados a ela). Logo, se um algoritmo retornar n grupos para n objetos, cada grupo terá precisão máxima (100%).

A revocação, também chamada de sensitividade, mede a fração de objetos corretamente alocados à determinada classe. Grupos com valor de revocação alto possuem poucos objetos que foram incorretamente alocados a ele. Porém deve-se ter cuidado com essa medida, pois em situações em que poucos grupos são retornados pelo algoritmo de agrupamento, a tendência é de que mais objetos, do que aqueles pertencentes à classe do grupo, estejam inseridos nele. Por exemplo, se o algoritmo retornar apenas um grupo com todos os objetos, a revocação será máxima (100%), pois para qualquer classe analisada, todos os objetos da classe estarão no grupo.

A *medida-F*, por sua vez, considera os valores de precisão e revocação no seu cálculo. Ela mede se um grupo possui apenas objetos de uma determinada classe e também se possui todos os objetos daquela classe. A equação para o cálculo da *medida-F* pode ser observada na Figura 2.7.

$$F(i,j) = \frac{2 \times precision(i,j) \times recall(i,j)}{precision(i,j) + recall(i,j)}$$

Figura 2.7: Equação para cálculo da Medida-F

Na equação presente na Figura 2.7, $precision(i,j)$ corresponde à precisão de um *cluster* i com respeito à classe j e $recall(i,j)$ corresponde à revocação de um *cluster* i com respeito à classe j.

2.3.2 Puridade e Entropia

Puridade e entropia são métricas de avaliação supervisionadas, que requerem que a elas sejam passadas informações a respeito do resultado esperado para que a avaliação possa ser feita.

A puridade avalia o número de objetos no grupo pertencentes à classe de dados mais frequente. Isso quer dizer que se um grupo tiver objetos de duas classes diferentes o grupo não será 100% puro, mas quanto mais objetos tiver de uma mesma classe e menos de outras, a puridade será maior, tendendo ao máximo (100%). Esta é uma métrica importante, pois nos diz se a divisão dos dados entre os grupos está sendo feita corretamente. Por exemplo, se arquivos de texto abordando assuntos do mercado financeiro e de entretenimento estivessem no mesmo grupo, poderia ser questionado se a seleção das características foi eficiente ou mesmo se o algoritmo de agrupamento está cumprindo o prometido.

Segundo Tan (2006), a puridade de um *cluster* é calculada como $p_i = max_j p_{ij}$, onde $p_i j$ corresponde à razão entre a quantidade de objetos do grupo i que pertencem à classe

j. Já a puridade total dos agrupamentos é dada pela equação da Figura 2.8, onde m_i corresponde ao número de objetos no grupo i e m é o número total de objetos daquela classe. A puridade que se almeja é o valor 1 (ou 100% de pureza), pois nesse caso todos os objetos estarão separados em grupos conforme as classes adequadas àquele conjunto de dados.

$$puridade = \sum_{i=1}^{K} \frac{m_i}{m} p_i$$

Figura 2.8: Equação para o cálculo da puridade total do sistema

A entropia avalia o quão dispersas as classes de objetos estão dentro de um grupo. Primeiro é calculada a distribuição das classes de objetos no grupo, ou seja, para uma classe j é calculado seu p_{ij}, que corresponde a probabilidade de um objeto do grupo i pertencer à classe j. p_{ij} é calculado como $p_{ij} = m_{ij}/m_i$, onde m_i é o número de objetos no grupo i e m_{ij} é o número de objetos da classe j no grupo i.

Utilizando-se cada distribuição das classes, a entropia de cada grupo i é calculada pela fórmula $e_i = -\sum_{j=1}^{L} p_{ij} \log_2 p_{ij}$, onde L é o número de classes. A entropia total é calculada como a soma das entropias de cada grupo pesadas pela quantidade de objetos em cada grupo, como mostrado na Figura 2.9, onde K é o número de grupos e m o número total de objetos.

$$entropia = \sum_{i=1}^{K} \frac{m_i}{m} e_i$$

Figura 2.9: Equação para o cálculo da entropia total do sistema

2.3.3 Silhueta

O coeficiente silhueta (*silhouette coefficient*) é um método de validação não supervisionado, que avalia os agrupamentos com base nas próprias características dos objetos e da matriz de similaridades. Ele combina duas métricas: coesão e separação. A coesão pode ser definida como a soma das similaridades dos objetos considerando o centroide de um grupo. De forma semelhante, a separação, pode ser definida como a proximidade, ou grau de afinidade, que centroides de diferentes grupos apresentam. A Figura 2.10 ilustra com mais clareza essas relações.

Neste método, para cada objeto agrupado, calcula-se a similaridade média deste com todos os demais objetos do mesmo grupo e compara-se com a similaridade média desse objeto com todos os objetos do grupo mais próximo. O resultado é um valor no intervalo [-1, 1], onde valores próximos a -1 indicam que o objeto deveria estar no grupo vizinho,

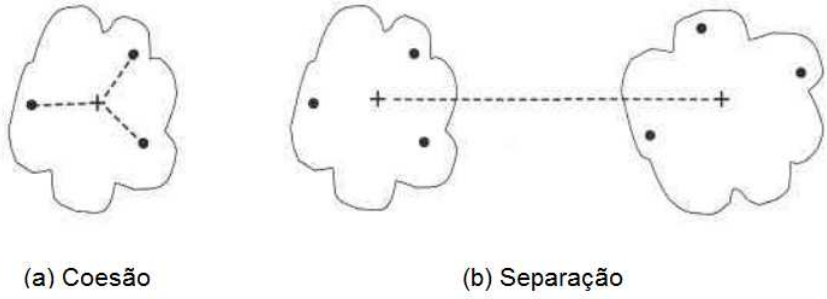

Figura 2.10: Coesão e separação de grupos com base nos seus centroides
Fonte: Adaptado de (TAN; STEINBACH; KUMAR 2006, p.538)

e valores mais próximos a 1 indicam a correta inserção do objeto no grupo atual. O coeficiente silhueta de um objeto i qualquer é como segue:

$$s_i = \begin{cases} 1 - b'_i/a'_i & \text{se } a'_i > b'_i \\ 0 & \text{se } a'_i = b'_i \\ a'_i/b'_i - 1 & \text{se } a'_i < b'_i \end{cases}$$

Figura 2.11: Coeficiente silhueta de um objeto i

Na equação presente na Figura 2.11, a'_i representa a média de similaridades entre o objeto i e todos os demais objetos do mesmo grupo e b'_i representa a média de similaridades entre i e todos os objetos do grupo mais próximo.

Aqui pode ser visto o coeficiente silhueta de um único objeto i, mas a silhueta total do sistema pode ser obtida pela soma de todas as silhuetas calculadas dividido pela quantidade de somas (número de objetos) realizadas. Lewis (1992) aborda essa questão ao apresentar a técnica de *Macroaveraging*, que nada mais é do que obter uma média dos diversos coeficientes de um mesmo tipo que são calculados, obtendo, portanto, um valor único que represente aquele coeficiente no conjunto de agrupamentos. Para o coeficiente silhueta, por exemplo, o cálculo da silhueta total é obtido conforme a equação da Figura 2.12. Para valores totais de precisão, revocação e medida-F o cálculo é análogo.

$$silhueta = \frac{\sum_{i=1}^{n} s_i}{n}$$

Figura 2.12: Cálculo da silhueta total do sistema de agrupamentos

2.4 Trabalhos Relacionados

São diversas as ferramentas de mineração de dados e aprendizado de máquina disponíveis atualmente. Dentre elas destacam-se algumas, como a ferramenta Weka[2] (*Waikato Environment for Knowledge Analysis*), um *software* livre, de código aberto (*open source*), disponível sob a licença GPLv3. Ela possui uma coleção de ferramentas e algoritmos para análise de resultados, mineração de dados e aprendizado de máquina. O Weka também possui uma interface gráfica de fácil utilização, na qual é possível realizar diversas operações, como pré-processamento, agrupamento de dados, classificação, regressão, visualização de dados e seleção de características. Seu foco, não é, portanto, de agrupamento de dados, visto que ele agrega diversas funcionalidades. Sua entrada de dados é um único arquivo no qual são descritas as relações entre os dados e os atributos de cada objeto. Importante também comentar que o Weka permite interação com banco de dados SQL.

A ferramenta Eurekha, desenvolvida por Wives (1999), possui foco em seleção de características de arquivos de texto e implementa um conjunto de algoritmos de agrupamento de dados. Ela também possui alguns recursos para visualização dos resultados, porém sua utilização é restrita ao sistema operacional Windows XP e não há a possibilidade de extensão ou consulta ao código fonte para desenvolvimento de outros projetos científicos.

Outra ferramenta é o ELKI[3] (*Environment for Developing KDD-Applications Supported by Index-Structures*), um *software open source* sob a licença GLPv3, escrito em Java e destinada à mineração de dados. Seu foco é para pesquisa em algoritmos, com ênfase em métodos não supervisionados para *cluster analysis*. O ELKI também foi desenvolvido com o intuito de ser estendido por estudantes e pesquisadores. Porém ele possui suporte limitado a somente arquivos de texto.

O objetivo do *framework* proposto nesse trabalho é que ele seja uma alternativa aos *softwares* apresentados e aos demais existentes atualmente, com foco em aplicações de mineração de dados, proporcionando diversas técnicas de agrupamento de dados e métricas de validação de resultados, bem como suporte a vários tipos de dados (texto, áudio e vídeo) para análise. Também se prima pela sua fácil incorporação em aplicações, e sua edição e extensão, permitindo flexibilidade aos desenvolvedores para realizarem alterações nos códigos fonte quando necessárias. Para tanto o projeto é *open source*, sob a licença GPLv3 e está disponível no GitHub. Não só isso, mas também como objetivo secundário, pretende-se obter um aprendizado sólido no campo do conhecimento ao qual o *framework* se destina, e também que ele se torne uma ferramenta cada vez mais completa a medida que conteúdos e funcionalidades são agregadas por contribuidores.

[2] `http://www.cs.waikato.ac.nz/ml/weka/`. Acessado em 02/10/2014
[3] `http://elki.dbs.ifi.lmu.de/`. Acessado em 02/10/2014

3 ESTRUTURA DO FRAMEWORK

Este capítulo tem o intuito de detalhar a estrutura do *framework*, expondo as classes utilizadas e seus relacionamentos, bem como as decisões de projeto.

3.1 Core classes

As *core classes*, ou classes núcleo, são as classes principais do *framework* para o processo de agrupamento de dados. A partir delas é que todos os algoritmos de seleção de características, cálculo de similaridades, agrupamento de dados e validação dos resultados são baseados. Na Figura 3.1 pode-se observar as classes mencionadas.

Para cada etapa do processo de agrupamento de dados uma classe base foi criada. Portanto se tem quatro classes: *FeatureSelectionStrategy*, *SimilarityStrategy*, *ClusteringStrategy* e *AnalysisStrategy*. Todas estendem uma classe base, chamada *BaseStrategy*, e especializam as suas funcionalidades com métodos específicos.

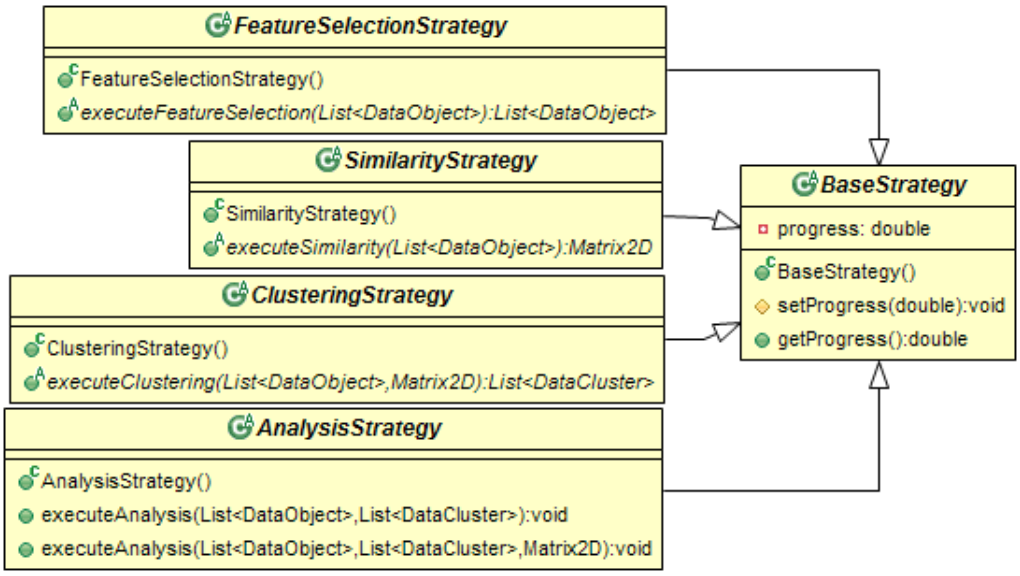

Figura 3.1: Core classes do processo de agrupamento de dados

3.2 Classes de algoritmos de seleção de características

O *framework* proposto possui suporte a três tipos de dados distintos: arquivos de texto, arquivos de áudio e arquivos de vídeo. Para cada um desses tipos de dados uma estratégia específica de seleção de características foi desenvolvida, conforme consta na Figura 3.2.

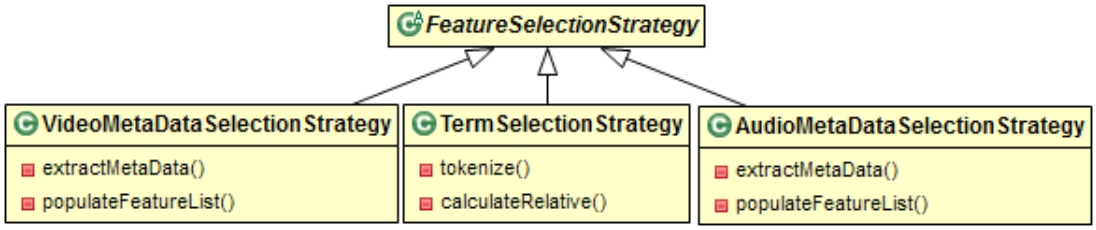

Figura 3.2: Classes para a seleção de características

Nas classes *VideoMetaDataSelectionStrategy* e *AudioMetaDataSelectionStrategy*, ao executar-se o método *executeFeatureSelection*, inicia-se o processo de seleção de características dos objetos. Métodos auxiliares são utilizados para extrair os *metadados* e construir a lista de *features* (características) de cada objeto.

De modo semelhante, a classe *TermSelectionStrategy*, realiza a seleção de características em arquivos de texto. A seleção ocorre após a remoção das palavras negativas (*stop-words*) e da operação de *tokenização*, ou seja, separação das palavras. Por fim, para cada palavra calcula-se a sua frequência relativa dentro do texto.

3.2.1 Xuggler API: extração de *metadados*

Xuggler é uma API utilizada para descomprimir, modificar e comprimir (re-comprimir) qualquer tipo de mídia ou *stream* de mídia em aplicações Java. Ela é uma ferramenta gratuita e *open source*, e seu uso é permitido sob a licença *GPL Version 3*.

Neste trabalho utiliza-se a API Xuggler para a extração de *metadados* de arquivos de áudio e de vídeo. Para arquivos de áudio, por exemplo, são extraídas informações como nome do artista, gênero musical, ano do álbum, nome do álbum, nome da música, *bitrate*, faixa da música, duração e tamanho. Já em arquivos de vídeo, as informações extraídas são duração, tamanho do arquivo, *bitrate*, *framerate* do vídeo (quantidade de quadros por segundo), dimensões do vídeo e *codec* de vídeo e de áudio utilizados. Essas informações depois são utilizadas para o cálculo das similaridades entre todos os arquivos, para depois ser realizado o agrupamento.

No Capítulo 4, de Análise dos Resultados, são apresentados testes realizados com arquivos de mídia. O leitor irá perceber que para arquivos de áudio utilizam-se arquivos no formato .mp3, enquanto para arquivos de vídeo utilizam-se arquivos nos formatos .mp4, .mkv e .avi. Isso ocorre devido a disponibilidade do autor a arquivos com *metadados* suficientes para a realização dos testes, mas o *framework* é capaz de suportar diversos outros formatos de áudio e de vídeo, devido a compatibilidade com a API Xuggler.

A API Xuggler, além de fácil utilização é uma ferramenta compatível com diversos sistemas operacionais, proporcionando portabilidade ao *framework*.

3.3 Classes de algoritmos de cálculo de similaridades

De modo análogo à Seção 3.2, no cálculo de similaridades, cada tipo de dados possui uma estratégia específica, conforme descrito na Figura 3.3.

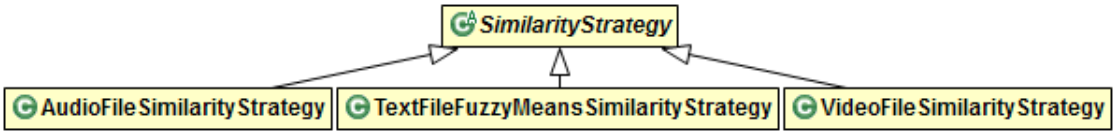

Figura 3.3: Classes para o cálculo das similaridades

3.3.1 Similaridade em arquivos de áudio e vídeo

Para arquivos de áudio e vídeo foram desenvolvidas estratégias específicas para o cálculo de similaridades. As novas estratégias também adotam a abordagem *fuzzy*, porém de um modo mais simplista do que para arquivos de texto. A grande diferença, na verdade, é com relação ao grau de igualdade, que não está mais presente nas equações utilizadas aqui. Em seu lugar é utilizado o nr, ou *nível de representatividade*. Esse valor indica qual a importância de determinada característica quando esta é comparada. A equação pode ser observada na Figura 3.4.

$$gs(X,Y) = \frac{\sum_{h=1}^{k} nr_h * h(a,b)}{\sum_{h=1}^{k} nr_h}$$

Figura 3.4: Equação para o cálculo de similaridades entre arquivos de áudio e de vídeo

A variável nr_h constitui um valor diferente para cada característica h, que indica a sua importância para o cálculo de similaridades. Para arquivos de áudio, por exemplo, são avaliadas características como nome da banda, nome do álbum, nome da música, ano do álbum, duração da música, *bitrate* e gênero musical. A cada uma dessas informações foi atribuído um nível de representatividade diferente, pois algumas informações, como nome do artista ou nome do álbum não mais importantes do que *bitrate* ou duração da música, por exemplo. Já o resultado retornado por $h(a,b)$ será sempre zero (0), quando a característica testada for diferente entre os objetos, ou um (1), quando a características testada for comum a ambos objetos.

Já para arquivos de vídeo as características avaliadas são duração, tamanho do arquivo, *bitrate*, *framerate* (quantidade de quadros por segundo), dimensões do vídeo e *codec* de vídeo e de áudio utilizados. O cálculo da similaridade para arquivos de vídeo ocorre de modo análogo ao cálculo para arquivos de áudio.

Logo, ao compararmos dois arquivos, se uma característica for comum à ambos, é somado o nível de representatividade daquela característica ao contador, caso contrário, soma-se zero. A divisão final é feita pelo somatório de todos os níveis de representatividade. Essa mesma abordagem é utilizada para o cálculo da matriz de similaridades para aquivos de vídeo.

3.4 Classes de algoritmos de agrupamento de dados

Como apresentado na Seção 2.2 de Algoritmos de agrupamento de dados, no *framework* proposto estão implementadas diversas estratégias de agrupamento de dados. A Figura 3.5 mostra a hierarquia de classes para esses algoritmos.

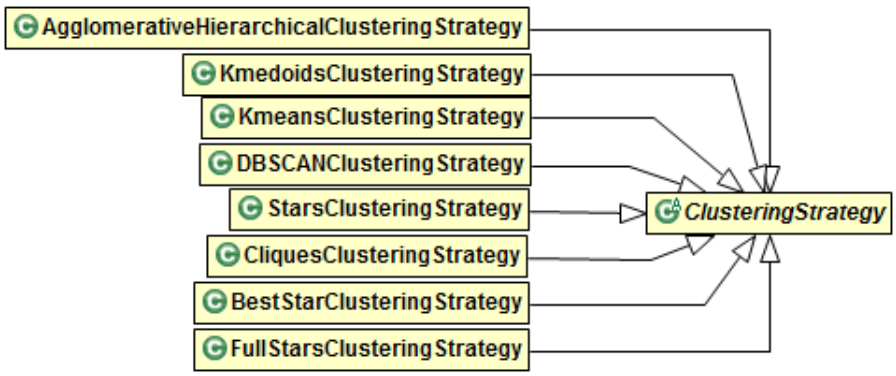

Figura 3.5: Classes para agrupamento de dados

Cada um dos algoritmos inicia sua execução executando o método *executeClustering*, herdado da classe base *ClusteringStrategy*. Alguns dos algoritmos também possuem métodos auxiliares para cálculos específicos de sua estratégia, como é o caso dos algoritmos *K-Medoids*, *K-Médias*, *DBSCAN* e *Aglomerativo Hierárquico*.

Os algoritmos *K-Medoids*, *K-Means*, *DBSCAN* e *Aglomerativo Hierárquico* são estratégias novas incorporadas no *framework*. A estratégia *Aglomerativa Hierárquica* foi escolhida por fazer parte da família de métodos hierárquicos, até então não presentes no *framework*. O mesmo ocorre para o *DBSCAN*, uma estratégia baseada em densidade, também não presente até o momento. Já as estratégias *K-Medoids* e *K-Means* foram escolhidas por serem bastante conhecidas na literatura e por apresentarem bons resultados de agrupamento.

3.5 Classes de validação de resultados

Outro conjunto de algoritmos implementado pode ser visto na Figura 3.6. Para as métricas de validação há quatro (4) classes, cada uma responsável por uma métrica diferente. Uma ressalva a respeito da *Medida-F*, é que nesta classe também estão implementadas as métricas de precisão e revocação, usadas no cálculo da *Medida-F*.

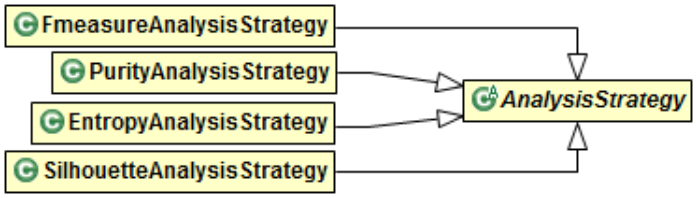

Figura 3.6: Classes de validação de resultados

3.6 Classes dos tipos de dados

Na Figura 3.7 pode-se observar o modo como é representado internamente cada objeto no *framework*, bem como a relação entre objetos e *clusters*, expressa pela classe *DataCluster*.

Um *DataObject* é um objeto que possui uma lista de características, ou uma lista de *DataFeature*. Um objeto pode ser de dois tipos, *TextFile* ou *MediaFile*, sendo que um *MediaFile* pode ter duas especializações, *VideoMediaFile* ou *AudioMediaFile*.

Uma *DataFeature* pode ser de dois tipos, *Term* ou *MetaData*. Em arquivos de texto, um *Term*, ou termo, é uma palavra do conteúdo do documento e a ela estão associadas informações como frequência absoluta e frequência relativa, utilizadas para o cálculo das similaridades entre objetos. Em arquivos de vídeo ou de áudio, um *MetaData* é uma característica extraída do conjunto de *metadados* do arquivo. Nele é identificado o tipo de informação (variável *id*) e o seu conteúdo (variável *value*).

3.7 Classes legadas

A classe *ClusteringProcess*, representada na Figura 3.8, foi desenvolvida com o intuito de organizar em apenas uma classe o processo de agrupamento de dados. Nela são estabelecidas todas as estratégias que pretende-se utilizar em uma dada execução, bem como o tipo de dados e a localização dos mesmos. Basta realizar a devida configuração de uma instância dessa classe para executar o processo de agrupamento de dados, e a validação de seus resultados, segundo uma das métricas de avaliação.

Há também a possibilidade de visualizar os resultados em uma ferramenta gráfica e acompanhar a conclusão de cada etapa do processo de agrupamento de dados e dos

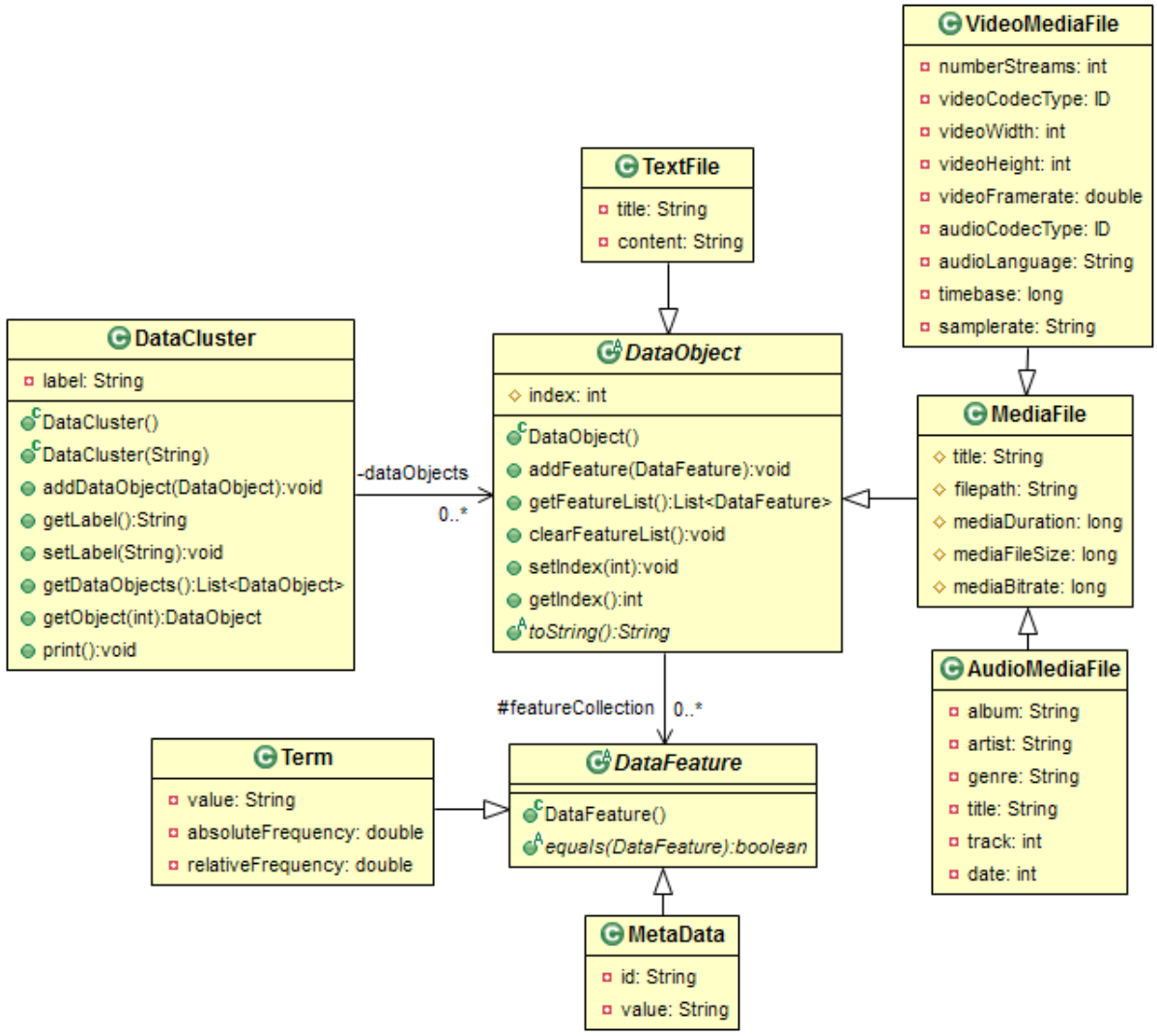

Figura 3.7: Classes dos tipos de dados suportados pelo *framework*

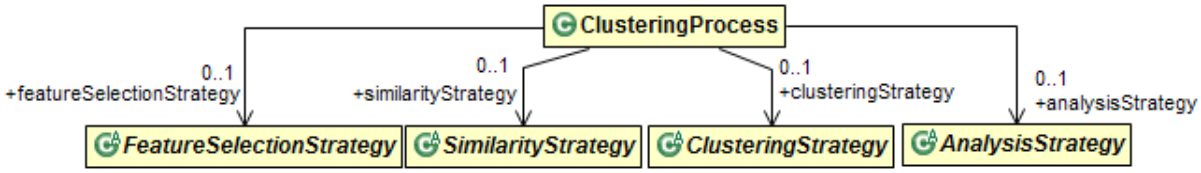

Figura 3.8: Classe para execução do processo de agrupamento de dados

resultados parciais de agrupamento com a classe *ConsoleObserver*. Essas classes e funcionalidades foram mantidas no novo *framework* sem alterações, e por não serem o foco deste trabalho, elas podem ser exploradas com mais detalhes em Ribacki (2013).

4 ANÁLISE DOS RESULTADOS

Neste capítulo são apresentados os conjuntos de dados utilizados para testar os algoritmos implementados no *framework* proposto, bem como os resultados obtidos nestas execuções e as conclusões advindas destes. O objetivo dos testes é validar o funcionamento dos algoritmos implementados conforme a sua especificação.

4.1 Os conjuntos de dados

Foram cinco (5) os conjuntos de dados utilizados na bateria de testes do novo *framework*. Destes, três compõem conjuntos de arquivos de texto, um conjunto para arquivos de áudio e um para arquivos de vídeo. Dois dos conjuntos de dados de arquivos de texto são de artigos publicados na *Wikipedia*[1], e são chamados de $wikipedia12$ e $wikipedia13$, por possuírem respectivamente 12 e 13 arquivos. O conjunto de dados $wikipedia12$ é o mesmo utilizado por Ribacki (2013) nos testes realizados com *framework* desenvolvido em seu trabalho, mantendo os mesmos arquivos, e, portanto, serve para comparativo dos resultados obtidos no antigo *framework* e no proposto neste trabalho. Os 12 arquivos estão distribuídos em cinco classes: *animals* (2 arquivos), *languages* (3 arquivos), *moons* (2 arquivos), *tv-shows* (3 arquivos), *video-games* (2 arquivos). Já o conjunto de dados $wikipedia13$, possui novos arquivos de texto, todos diferentes do conjunto *wikipedia12*, e tem por objetivo diversificar os resultados, buscando comprovar o correto funcionando dos algoritmos de agrupamento. Os 13 arquivos estão distribuídos em três classes: *animals* (5 arquivos), *tv-shows* (4 arquivos) e *movies* (4 arquivos).

Também é utilizado um terceiro conjunto de arquivos de texto, chamado $reutersTop10$, o qual é composto de arquivos do pacote *Reuters-21578*. Este pacote possui 21.578 arquivos, devidamente classificados, e é fornecido pela agência de notícias Reuters[2] para uso em pesquisas científicas. Inicialmente optou-se por utilizar todo o pacote *Reuters-21578* para a realização de testes, porém, devido a gama de arquivos, a execução tornava-se muito lenta, chegando a demorar dias, e, então, resolveu-se readequar esse pacote de dados. Foram escolhidas as dez (10) classes mais populosas do pacote *Reuters-21578* e

[1] `http://www.wikipedia.org/`. Acessado em: 04/10/2014
[2] `http://br.reuters.com/`. Acessado em: 05/10/2014

estas passaram a compor o conjunto $reutersTop10$. Assim, o total de arquivos do conjunto $reutersTop10$ é de 1.248, e alguns dos arquivos pertencem a mais de uma das dez classes existentes.

O conjunto de arquivos de áudio, chamado $audio30$, é composto por 30 arquivos de áudio no formato .mp3. Eles pertencem a 6 bandas de música, *Maroon 5*, *Focus*, *Grand Funk Railroad*, *Audioslave*, *Paramore* e *Jason Becker*, e foram extraídos de 7 álbuns diferentes dessas mesmas bandas, *Caught In The Act (Grand Funk Railroad)*, *Hands All Over (Maroon 5)*, *Out Of Exile (Audioslave)*, *Riot! (Paramore)*, *Ship Of Memmories (Focus)*, *Songs About Jane (Maroon 5)*, *The Blackberry Jams (Jason Becker)*. Já o conjunto de arquivos de vídeo, chamado $video21$, é composto por 21 arquivos de vídeo nos formatos .mkv, .mp4 e .avi. Eles são pertencentes a 5 séries de TV distintas, *Sword Art Online II*, *Shingeki no Kyojin*, *Death Note*, *Hunter x Hunter* e *Shurato*. Os conjuntos $audio30$ e $video21$ foram construídos pelo próprio autor.

4.2 Coleção de arquivos $wikipedia12$

Para todas coleções de arquivos de texto, aí incluso o conjunto $wikipedia12$, testou-se todos os algoritmos desenvolvidos para o novo *framework*, ou seja, *Best-Star*, *Full-Stars*, *Stars*, *Cliques*, *Aglomerativo Hierárquico*, *K-Means*, *K-Medoids* e *DBSCAN*. Para os primeiros cinco algoritmos foram realizadas 21 execuções, cada uma com um valor diferente de GSM, de 0.0 a 1.0. Foram realizadas 11 execuções com GSM variando de 0.01 em 0.01 na faixa de 0.0 a 0.1, pois nessa faixa pequenas alterações no GSM resultaram em grandes mudanças nos resultados finais, logo optou-se por ter mais testes nessa faixa. Na faixa de GSM 0.2 a 1.0 foram realizadas 10 execuções com GSM variando de 0.1 em 0.1, pois para GSM 0.2 em diante não houveram grandes variações nos resultados finais, e, portanto, menos execuções foram necessárias. Para o algoritmo *DBSCAN* também foram realizadas 21 execuções, porém com variações do valor de *épsilon* de 1.0 a 0.0. Por fim, os algoritmos *K-Means* e *K-Medoids* foram executados com 7 valores de *k* diferentes, um deles sendo o k (número de grupos) ideal para o conjunto de dados, 3 abaixo de k ($k-1$, $k-2$ e $k-3$) e 3 acima de k ($k+1$, $k+2$ e $k+3$). Para esses dois algoritmos também foram utilizadas diferentes estratégias para escolha dos primeiros centroides e dos novos centroides a cada iteração.

Na primeira bateria de testes, foram escolhidos os algoritmos *Best-Star*, *Full-Star*, *Stars*, *Cliques*, *Aglomerativo Hierárquico*, por receberem o GSM como parâmetro e também por apresentarem funcionamento semelhante. Todos os algoritmos conseguiram agrupar corretamente os objetos nos seus agrupamentos ideais. Também percebeu-se que o *Best-Star* obteve resultados levemente melhores se comparados aos demais algoritmos, segundo as métricas de validação de resultados. A Figura 4.1 mostra uma compilação dos resultados obtidos por cada um dos algoritmos em cada uma das seis métricas de

avaliação utilizadas para a validação dos resultados.

Na Figura 4.1 os gráficos não apresentam os resultados de todas as 21 execuções, com a variação de GSM de 0.0 a 1.0. Isso se deve ao fato de que do ponto de corte, no eixo das abcissas (GSM), em diante, os resultados são os mesmos, e, portanto, apenas os resultados de interesse foram mantidos, facilitando sua visualização.

Percebe-se que de modo geral o *Best-Star* obteve um desempenho superior aos demais algoritmos. Estes resultados tendem a ser ideais para o conjunto $wikipedia12$ ao se atingir o GSM 0.1. Com GSM 0.1 a precisão e revocação do *Best-Star* são totais, ou seja, atingem o valor 1.0 (100%), bem como a medida-F. A entropia é 0.0 e a puridade 1.0, também valores ideais. Apenas o coeficiente silhueta que ficou aquém do esperado, em 0.6, sendo que o valor ideal seria 1.0, mas de qualquer forma um resultado também muito bom.

Esses resultados vão de acordo com as percepções de Ribacki (2013), no qual são confrontados os algoritmos *Best-Star*, *Full-Stars*, *Stars* e *Cliques*, onde o Best-Star obteve os melhores resultados ao utilizar-se o conjunto de dados $wikipedia12$.

Os algoritmos *Full-Stars* e *Aglomerativo Hierárquico* apresentaram os piores resultados. Do *Full-Stars* é esperado que os resultados não sejam muito bons, pois devido às suas características, durante a execução podem ser construídos *clusters* que compartilham um ou mais objetos. Porém, o conjunto $wikipedia12$ não admite inserção de objetos em mais de um *cluster*, e, portanto, os resultados tendem a ser piores, e isso foi constatado nos resultados obtidos. Por exemplo, na medida de puridade com GSM 0.15, o algoritmo *Full-Stars* obteve puridade superior a 1.0, mais precisamente 1.08, sendo que o limite é 1.0. Logo, a presença de um objeto em múltiplos grupos de fato influencia negativamente nos resultados das métricas de validação.

Já o algoritmo *Aglomerativo Hierárquico* deveria apresentar resultados semelhantes ao *Stars*, mas não foi o caso. Os critérios de união de grupos, estabelecidos no algoritmo *Aglomerativo Hierárquico*, podem ter influenciado negativamente nos resultados, dificultando o correto agrupamento dos objetos. Estes estão sujeitos a nova análise e melhorias em trabalhos futuros.

O algoritmo *DBSCAN* foi executado com variação de *épsilon* de 1.0 a 0.0. Porém para valores de *épsilon* de 0.5 a 0.0 foram gerados menos de dois *clusters*, e esses resultados foram descartados. A Figura 4.2 apresenta os resultados obtidos.

Os melhores resultados do *DBSCAN* são encontrados na execução com *épsilon* igual a 0.85, pois nessa execução os objetos foram agrupados conforme os grupos esperados. O *épsilon* deve ser entendido com a máxima diferença de similaridades entre dois objetos para que estes pertençam ao mesmo grupo, ou seja, se a similaridade entre dois objetos for 0.15 ou maior, então eles podem pertencer ao mesmo grupo. Para *épsilon* igual a 0.85, apenas o coeficiente silhueta não apresentou o melhor resultado possível, ficando em 0.6, ao invés de 1.0.

Percebe-se que os melhores resultados são encontrados apenas depois que as várias

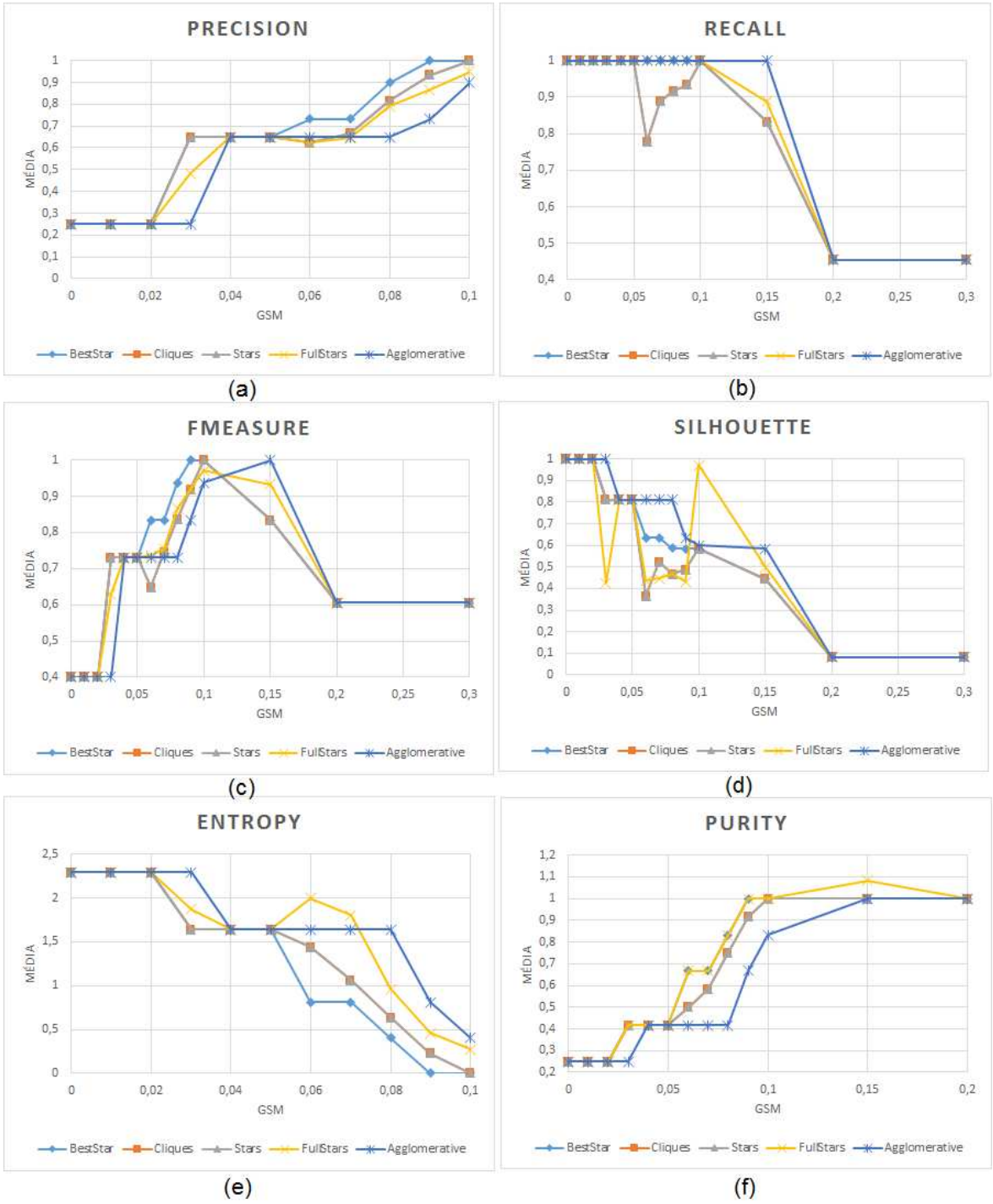

Figura 4.1: Resultados das métricas de validação para o conjunto $wikipedia12$

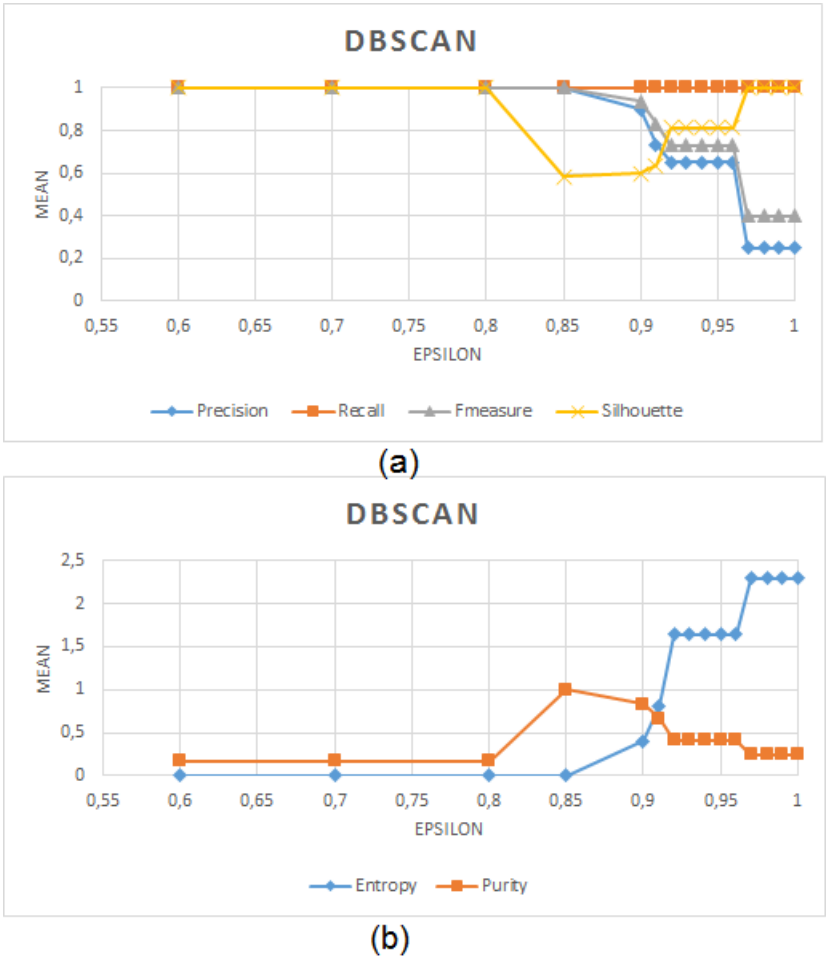

Figura 4.2: Resultados das métricas de validação para o *DBSCAN* ($wikipedia12$)

execuções, com parâmetros (GSM ou *épsilon*) diferentes, são realizadas. Assim, conclui-se que para obter o melhor agrupamento para um dado conjunto de dados, é necessário que o algoritmo de agrupamento de dados escolhido seja submetido a diversas execuções, e que seus resultados sejam validados pelas métricas de avaliação. Neste trabalho esse processo foi realizado manualmente e a descoberta dos melhores resultados também foi manual, mas nada impede que em trabalhos futuros seja desenvolvido um procedimento automático que avalia a necessidade ou não de realizar nova execução com parâmetros diferenciados, agilizando a obtenção dos melhores resultados. Com isso mais dados poderiam ser analisados, aumentando a capacidade do sistema de agrupar e analisar dados.

O algoritmo *K-Means* não conseguiu agrupar os objetos corretamente. Isso ocorreu devido a problemas de implementação, pois o cálculo do novo centroide dos *clusters* em cada iteração não se mostrou eficiente. A Figura 4.3 mostra os resultados obtidos.

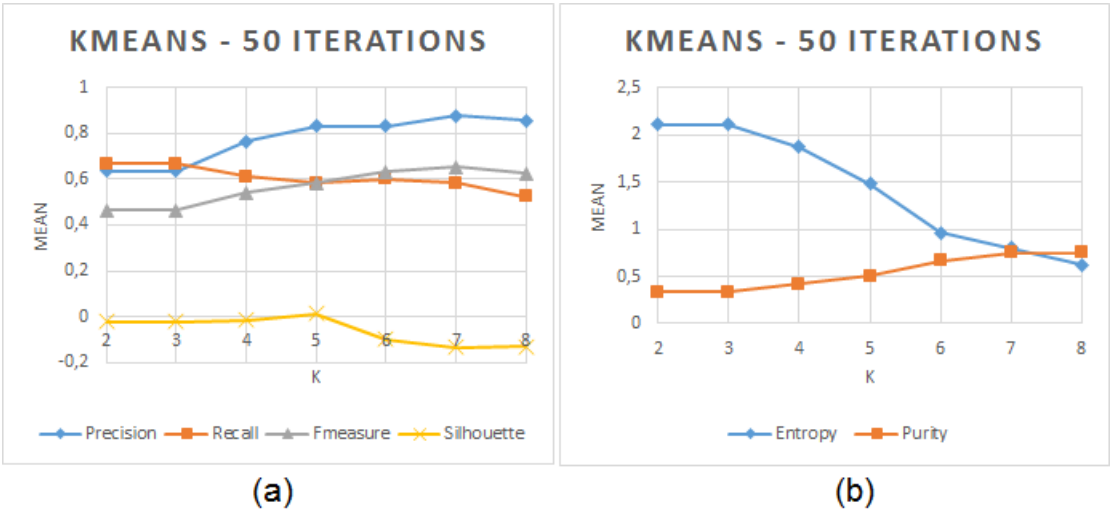

Figura 4.3: Resultados das métricas de validação para o *K-Means* ($wikipedia12$)

Ao variar-se o valor de k, o comportamento dos resultados não oscilou muito. De um modo geral os agrupamentos calculados foram semelhantes aos agrupamentos ideais para esse conjunto de dados, principalmente considerando as métricas de precisão, revocação, *medida-F* e silhueta.

Os resultados dos dois gráficos da Figura 4.3 foram obtidos para execuções com 50 iterações. Essas 50 iterações dizem respeito ao número limite de iterações que cada execução realiza, que ao ser atingido finaliza a execução. Também foram realizados testes com 100 iterações, porém os resultados foram praticamente os mesmos, sem haver expressivas diferenças nos agrupamentos encontrados.

Para testar o algoritmo *K-Medoids* foram realizados quatro (4) testes diferentes. Uma particularidade do *K-Medoids* é que foram implementadas duas estratégias para a seleção dos primeiros centroides, uma estratégia de seleção sequencial e outra aleatória, e duas estratégias para o cálculo o novo centroide dos grupos a cada iteração, e, portanto, ao

mesclar-se essas quatro estratégias obtive-se quatro testes. Cabe ressaltar que da mesma forma como realizado com o algoritmo *K-Means*, o k foi variado em cada um dos testes. A Figura 4.4 mostra os resultados obtidos nessas execuções.

De modo geral, as execuções que utilizaram seleção aleatória dos primeiros centroides conseguiram identificar agrupamentos mais semelhantes aos agrupamentos ideais, segundo as métricas de avaliação. Para as execuções com seleção aleatória, em cada k utilizado, foram feitas cinco execuções e a melhor execução destas foi escolhida para compor os resultados. Para os testes com seleção sequencial, realizou-se apenas uma execução para cada k, pois estas execuções são todas determinísticas, ou seja, não alteram o resultado de uma execução para outra.

Assim conclui-se que o algoritmo *K-Medoids* é extremamente dependente da escolha dos primeiros centroides, pois a correta escolha destes é que garante bons resultados finais. Não há, porém, uma forma simples de inferir quais são os centroides iniciais que devem ser escolhidos. A estratégia de múltiplas execuções, e a posterior escolha da melhor delas é uma opção, porém seu custo computacional é alto e pode não ser viável em algumas aplicações. Outra alternativa é fazer uso de heurísticas que fazem um prévia análise dos objetos do conjunto de dados e determinam quais os centroides com maior potencial de desenvolver bons resultados finais. Essa técnica seria menos custosa do que múltiplas execuções, mas poderia pecar em não saber indicar com a precisão necessária quais centroides escolher. Por este motivo é que se adotou a alternativa aleatória, embora custosa, garante melhores resultados.

A dependência da escolha dos primeiros centroides para obter bons agrupamentos não é exclusividade do algoritmo *K-Medoids*. O algoritmo *K-Means* também possui essa mesma dependência, visto que ambos os algoritmos tem funcionamentos bastante semelhantes.

4.3 Coleção de arquivos $wikipedia13$ e $reutersTop10$

A execução do conjunto de dados $wikipedia13$ apenas reforçou o correto funcionamento dos algoritmos de agrupamentos de dados em praticamente todos os casos. Novamente obteve-se resultados pouco satisfatórios para o algoritmo *K-Means*, e como já explicado na Seção 4.2, o motivo desses resultados é devido aos critérios de escolha dos novos centroides, que se mostraram ineficazes. Também ficou evidente a influência distorciva do algoritmo *Full-Stars* na métrica de puridade, por apresentar resultado acima do limite de 1.0. Na Figura 4.5 pode-se observar os saltos de puridade acima do limite de 100%, devido a inclusão de objetos em mais de um grupo.

O conjunto de dados $reutersTop10$, quando executado pelos algoritmos de agrupamento de dados, obteve agrupamentos semelhantes aos ideais, superando a expectativa inicial, pois devido à grande quantidade de objetos, era esperado que haveria mais difi-

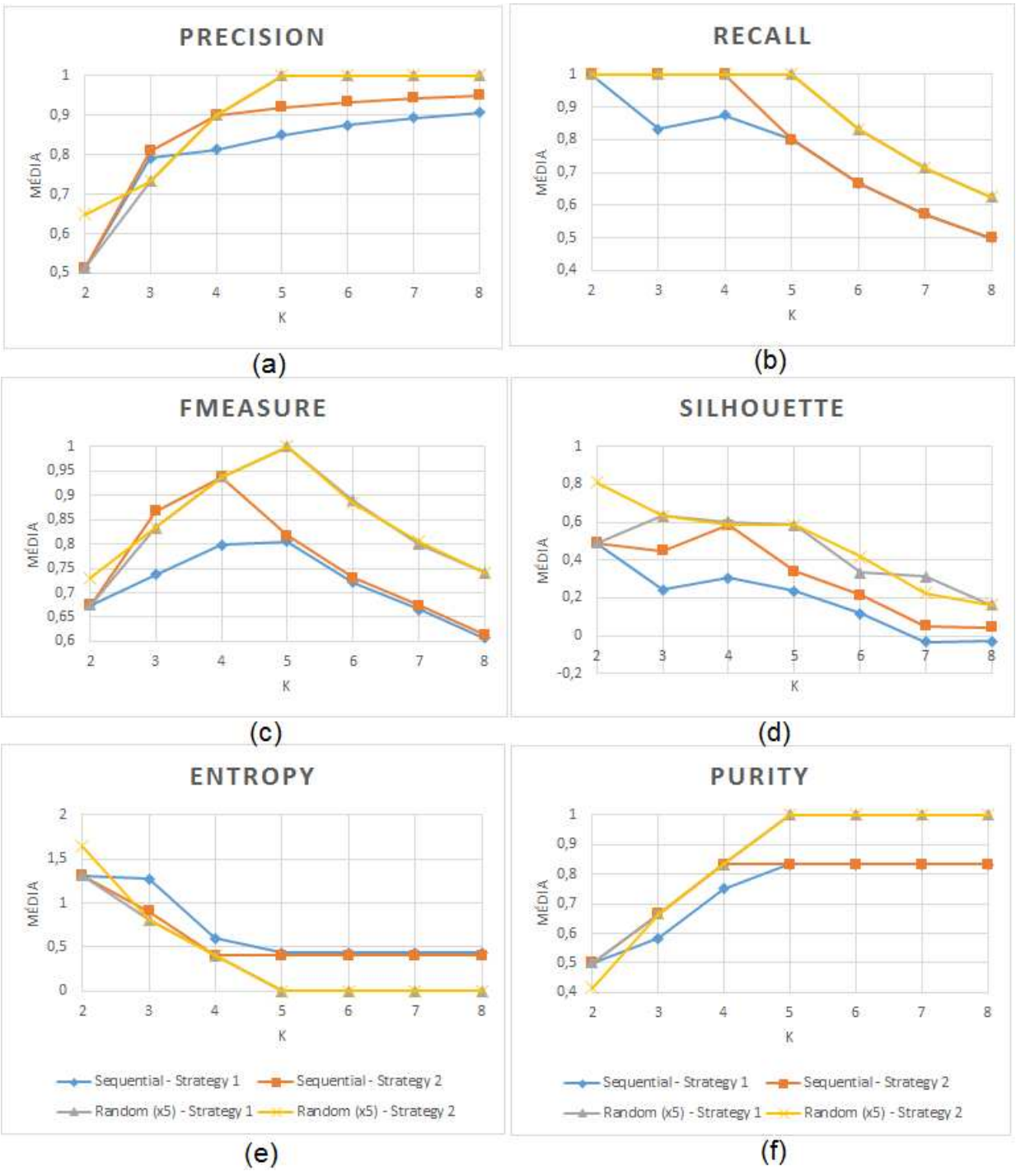

Figura 4.4: Resultados das métricas de validação para o *K-Medoids* ($wikipedia12$)

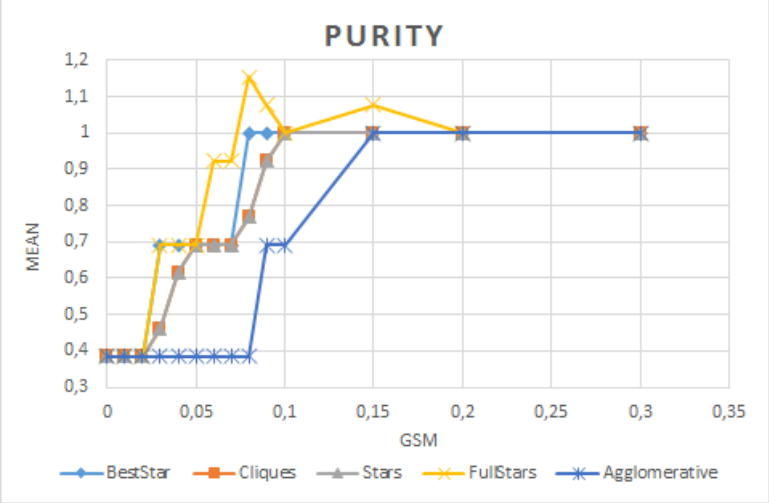

Figura 4.5: Resultados da métrica de puridade ($wikipedia13$)

culdade em agrupar os objetos corretamente segundo as classes pré-definidas no conjunto *Reuters-21578*. Devido à grande quantidade de arquivos, os agrupamentos calculados estão mais sujeitos a apresentarem diferenças com relação às classes naturais dos objetos. O que foi observado é que os agrupamentos mais próximos aos ideais foram obtidos para valores menores de GSM se comparados com os conjuntos $wikipedia12$ e $wikipedia13$. Uma explicação para isso é a dificuldade que o *framework* ainda enfrenta ao calcular similaridades representativas entre os objetos, condizentes com o conteúdo de cada um, pois a seleção e extração de características ainda permite muitas melhorias para aperfeiçoar esse processo. Em todo o caso, considerando-se o estado atual de desenvolvimento do *framework*, os resultados obtidos pelos algoritmos *Best-Star*, *Full-Stars*, *Stars*, *Cliques* e *Aglomerativo Hierárquico*, são mostrados na Figura 4.6.

Os melhores agrupamentos foram obtidos pelo algoritmo *Best-Star* para GSM igual a 0.06. Este é um valor bastante baixo de GSM, e quanto menor o GSM maior é a chance de haver objetos em grupos aos quais eles não deviam ser destinados, e, portanto, os resultados tendem a ser piores.

As execuções com os algoritmos *DBSCAN*, *K-Means* e *K-Medoids* não obtiveram bons agrupamentos. O algoritmo *DBSCAN*, por exemplo, teve os melhores resultados para valores de *épsilon* próximos a 0.90, porém todas as métricas indicavam agrupamentos com objetos de grupos distintos e com quantidade de grupos diferente da ideal para este conjunto de dados, ou seja, 10 grupos. Já o algoritmo *K-Medoids* também não foi capaz de descobrir os 10 agrupamentos corretamente, apresentando problemas semelhantes aos do *DBSCAN*. A estratégia de seleção aleatória dos primeiros centroides propiciou uma pequena melhoria na correta identificação dos agrupamentos, porém ainda diferente dos agrupamentos ideais.

Com base nos resultados do conjunto de dados $reutersTop10$, conclui-se que o *framework* necessita melhorar as suas estratégias de coleta de características, de modo a des-

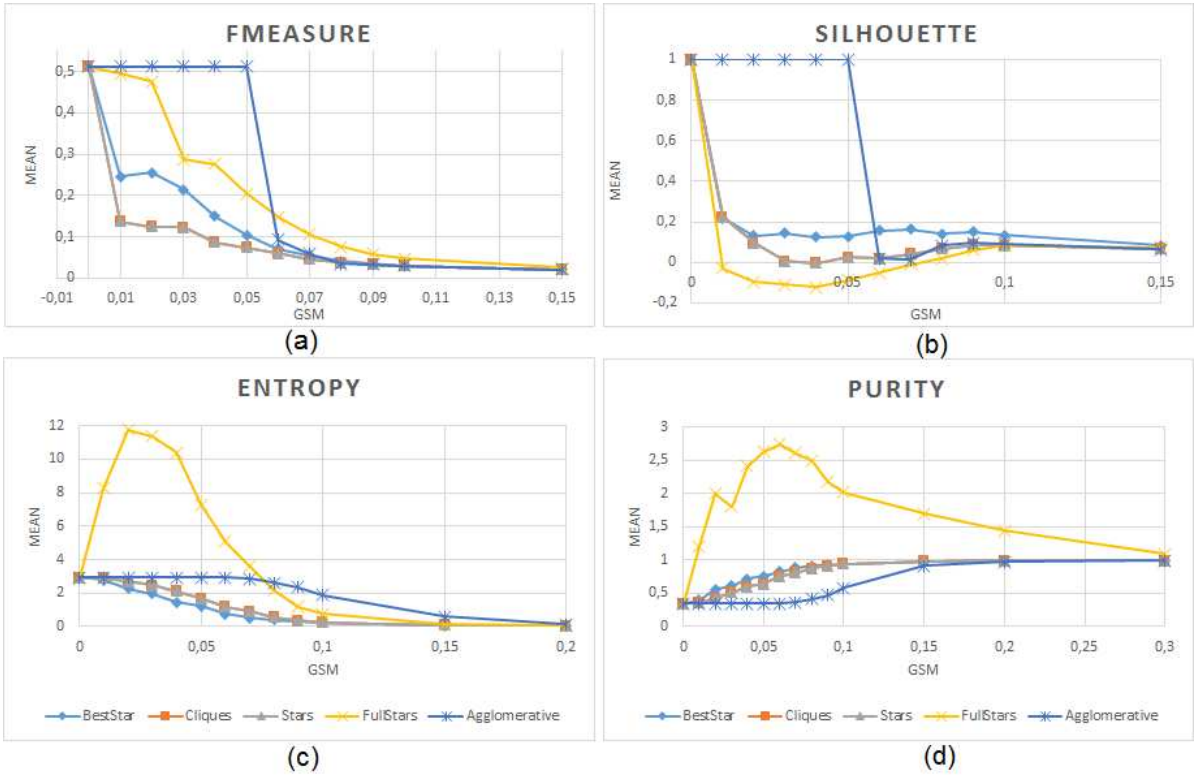

Figura 4.6: Resultados das métricas de validação para o conjunto $reutersTop10$

tacar características que remetam ao assunto que determinado arquivo de texto aborda, ou seja, elaborar estratégias de seleção que sejam capazes de analisar com mais eficiência a semântica de cada documento. Isso ajudaria a aumentar as similaridades entre os arquivos de texto que de fato são similares e assim os resultados seriam mais confiáveis.

4.4 Coleção de arquivos $audio30$

Na execução do conjunto de dados $audio30$, foram obtidos os agrupamentos corretamente, ao serem utilizados todos os algoritmos de agrupamento de dados disponíveis no *framework*, à exceção do *K-Means*. Dentre os algoritmos *Cliques*, *Stars*, *Full-Stars*, *Best-Star* e *Aglomerativo Hierárquico* os resultados foram praticamente os mesmos. Isso se deve ao fato de que os objetos possuíam *metadados* bastante representativos, estabelecendo uma relação forte com o conteúdo de cada arquivo. Desse modo, as similaridades entre objetos que pertenciam a uma mesma classe foram elevadas, enquanto objetos pertencentes a classes diferentes mostraram semelhanças muito baixas, fatores esses determinantes para a correta separação dos objetos nas suas classes naturais.

A Figura 4.7 mostra os resultados obtidos pelo conjunto de dados $audio30$ ao se utilizar o algoritmo *Best-Star*. A execução com os melhores resultados foi aquela com GSM igual a 0.6, onde praticamente todas as métricas de avaliação ficaram com seus valores ótimos, indicando correta identificação das classes naturais do conjunto de dados. Este

é um valor alto de GSM, se comparado aos GSMs ótimos encontrados para os conjuntos $wikipedia12$ e $wikipedia13$. Porém a diferença existe, pois o cálculo das similaridades para o conjunto $audio30$ considera menos características do que em arquivos de texto, então características comuns em dois arquivos de áudio aumentam consideravelmente a semelhança entre eles.

Figura 4.7: Resultados das métricas de validação utilizando o algoritmo *Best-Star* ($audio30$)

O algoritmo *DBSCAN* também apresentou bons resultados. Para valores de *épsilon* iguais a 0.3 e 0.4 obteve-se a distribuição dos objetos em grupos iguais àqueles das suas classes naturais. Já o algoritmo *K-Medoids* identificou os grupos ideias, ao ser utilizado k igual a 7 e a estratégia aleatória de seleção dos primeiros centroides (vide Figura 4.8), indicando bom funcionamento do algoritmo, visto que são 7 as classes do conjunto $audio30$. Novamente ressalta-se a necessidade de escolher corretamente os primeiros k centroides de modo que o algoritmo *K-Medoids* obtenha bons resultados de agrupamento. Isso foi apenas possível com o auxílio das estratégias aleatórias, já que as estratégias sequenciais dificilmente acertam na escolha dos melhores centroides.

4.5 Coleção de arquivos $video21$

Para o conjunto de dados $video21$ houve dificuldade em encontrar corretamento os agrupamentos ideais. Os melhores resultados de agrupamento foram obtidos com os algoritmos *Best-Star* e *K-Medoids* com a estratégia de seleção aleatória dos primeiros centroides. Os agrupamentos encontrados não foram tão bons quanto aqueles obtidos com o conjunto de dados $audio30$, pois os *metadados* do conjunto $video21$ faziam referência a informações técnicas da mídia, ao invés de referenciar o conteúdo do arquivo. Por isso as

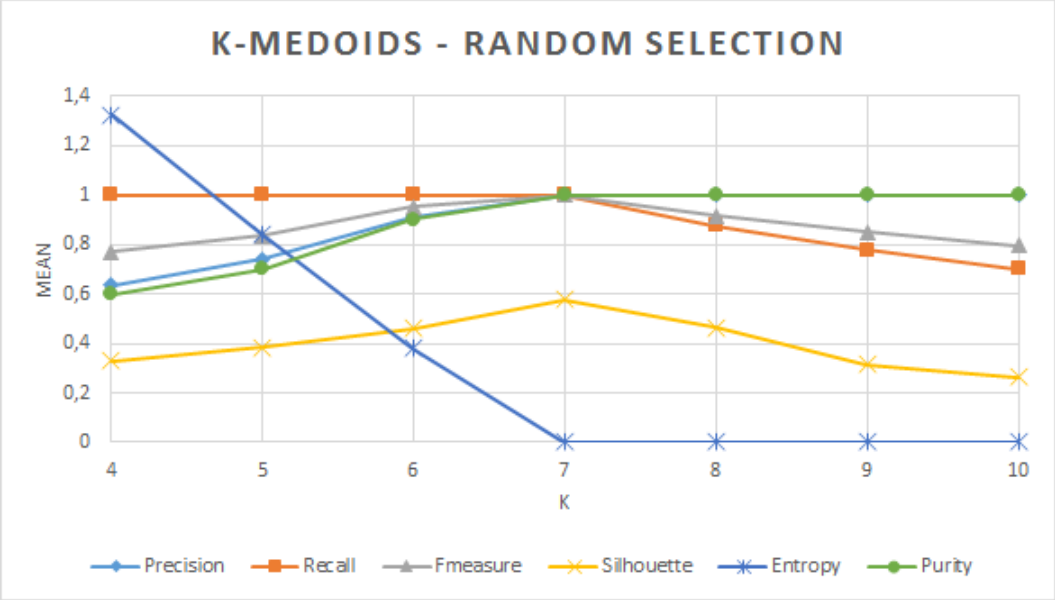

Figura 4.8: Resultados das métricas de validação utilizando o algoritmo *K-Medoids* e a estratégia de seleção aleatória ($audio30$)

similaridades entre todos os objetos acabaram parecidas e elevadas, mesmo entre aqueles arquivos que deveriam ter valores de similaridade baixos.

Devido aos resultados apresentados, conclui-se que para haver agrupamentos mais fiéis às classes naturais dos objetos é preciso refinar a coleta de *metadados* em arquivos de vídeo. Em arquivos de áudio não há esse problema, pois os *metadados* geralmente disponíveis em arquivos de áudio musicais remetem ao conteúdo do arquivo, diferentemente de arquivos de vídeo, onde os *metadados* geralmente apresentam informações técnicas, como tamanho do arquivo, *frames* por segundo e quantidade e tipos de *streams* (*streams* de áudio e vídeo, por exemplo). Logo, em trabalhos futuros pode ser aperfeiçoada a coleta de *metadados* em arquivos de vídeo, para que se obtenham melhores resultados de agrupamento.

5 CONCLUSÃO

O presente trabalho teve como objeto a extensão do *framework* desenvolvido por Guilherme Haag Ribacki em janeiro de 2013. Na atual proposta foram implementados novos algoritmos de seleção de características, cálculo de similaridades, agrupamento de dados e validação de resultados. Dentre esses, destacam-se a implementação dos algoritmos de agrupamento *K-Means*, *K-Medoids*, *DBSCAN* e *Aglomerativo Hierárquico* e de novas técnicas de validação, como *Puridade* e *Entropia*. Além disso a nova versão do *framework* possibilitou o agrupamento e análise de mais dois tipos de dados, arquivos de áudio e arquivos de vídeo, além do suporte a arquivos de texto.

O *framework* também foi concebido para ser estendido por outros desenvolvedores e, portanto, sua edição é livre, segundo a licença GPLv3, e está disponível para acesso no *GitHub*[1] por qualquer pessoa.

Um estudo minucioso sobre o processo de agrupamento de dados e sobre os algoritmos de agrupamento de dados e as métricas de validação de resultados, também foi realizado. Foram descritos com detalhes cada uma das etapas do processo de agrupamento de dados, bem como o funcionamento de cada um dos algoritmos implementados. Logo, como objetivo secundário, uma contribuição que este trabalho também realiza é a documentação do estado da arte do processo de agrupamento de dados e dos algoritmos implementados no *framework*, presente no Capítulo 2 Referencial Teórico, permitindo que outros estudiosos e cientistas possam se beneficiar deste estudo em seus próprios trabalhos.

Para validar a nova ferramenta, esta foi posta à prova com cinco conjuntos de dados distintos, sendo três para arquivos de texto, um para arquivos de áudio e um para arquivos de vídeo. Nos testes realizados com os conjuntos de dados de arquivos de texto obteve-se excelentes resultados de agrupamento com os dois conjuntos pequenos, de 12 e 13 documentos, respectivamente. Porém, o conjunto de dados com 1.248 arquivos não apresentou agrupamentos ideais, devido à grande quantidade de arquivos os algoritmos de agrupamento estiveram sujeitos a inserir nos grupos mais objetos não pertencentes as suas classes naturais. Além disso, observou-se que os algoritmos de seleção de características

[1] `https://github.com/jlggross/URSA`. Acessado em: 10/10/2014

ainda podem ser melhorados, de modo a extrair características que remetam ao conteúdo do documento, onde haja uma relação semântica entre as características.

Constatou-se também que para arquivos de áudio foi possível agrupar corretamente todos os objetos ao serem utilizados os algoritmos e agrupamento de dados disponíveis no *framework*. Já os agrupamentos obtidos para arquivos de vídeo não foram tão bons. Isso explica-se devido à falta de características representativas do conteúdo de cada arquivo que são consideradas pelo algoritmo de seleção de características. Assim as similaridades entre todos os objetos acabaram parecidas, dificultando a obtenção de bons resultados de agrupamento.

Os melhores resultados de agrupamento só puderam ser obtidos após realizadas diversas execuções com cada um dos conjuntos de dados, onde a cada execução os parâmetros foram modificados. Apenas assim obteve-se os melhores resultados de agrupamento em cada caso, o que evidenciou a necessidade de um processo iterativo no qual os agrupamentos de um conjunto de dados são calculados, segundo certos parâmetros, e estes resultados avaliados pelas métricas de avaliação; se os índices de qualidade de resultados forem positivos, o processo termina com os agrupamentos calculados, caso contrário uma nova rodada de execuções com outros parâmetros se inicia, buscando agrupamentos mais fiéis às classes naturais do conjunto de dados.

Devido ao tempo limitado e a restrição do escopo deste trabalho, muitas melhorias tiveram que ficar de fora. Uma delas, crucial no processo de interpretação dos resultados, é a visualização dos agrupamentos obtidos. Atualmente há uma opção de visualização dos resultados, legada na primeira versão do *framework*, porém muito simples e que necessita ser remodelada. Também fica a cargo de trabalhos futuros aperfeiçoar a seleção e extração de características dos objetos, assim como estender a ferramenta com novos algoritmos de agrupamento de dados, cálculo de similaridades e novas métricas de validação.

REFERÊNCIAS

BERKHIN, P. Survey Of Clustering Data Mining Techniques. In: KOGAN, J.; NICHOLAS, C.; TEBOULLE, M. (Ed.). **Grouping Multidimensional Data**. 1045 Forest Knoll Dr., San Jose, CA, 95129: Springer Berlin Heidelberg, 2006.

CAO, F. et al. Density-based clustering over an evolving data stream with noise. **SIAM Conference on Data Mining**, USA, p.328–339, 2006.

ESCHRICH, S. et al. Fast accurate fuzzy clustering through data reduction. **Fuzzy Systems, IEEE Transactions on**, Tampa, FL, USA, v.11, n.2, p.262–270, April 2003.

ESTER, M. et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases With Noise. **Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)**, 2275 East Bayshore Road, Suite 160, Palo Alto, California 94303, USA, p.226–231, 1996.

EVERITT, B. S. et al. **Cluster Analysis, 5th Edition**. John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom: Wiley, 2011. (Wiley Series in Probability and Statistics).

FAYYAD, U. M. et al. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On Clustering Validation Techniques. **J. Intell. Inf. Syst.**, Hingham, MA, USA, v.17, n.2-3, p.107–145, Dec. 2001.

JAIN, A. K.; DUBES, R. C. **Algorithms for Clustering Data**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data Clustering: a review. **ACM Computing Surveys**, New York, NY, USA, v.31, n.3, p.264–323, Sept. 1999.

KOWALSKI, G. **Information Retrieval Systems**: theory and implementation. 3300 AH Dordrecht, The Netherlands: Kluwer Academic Publishers, 1997. 282p. (Kluwer international series on information retrieval).

KRIEGEL, H.-P. et al. Density-based clustering. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Hoboken, New Jersey, USA, v.1, n.3, p.231–240, 2011.

LEWIS, D. D. Representation and Learning in Information Retrieval. **Amherst: University of Massachusetts, Department of Computer and Information Science**, Amherst, MA 01003, USA, 1992. PhD Thesis.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. **Introduction to Information Retrieval**. New York, NY, USA: Cambridge University Press, 2008.

MARIA HALKIDI YANNIS BATISTAKIS, M. V. Cluster Validity Methods: part i. **SIGMOD Record**, 28is Oktovriou 76, Athina, Grécia, v.31, n.2, Junho 2002.

NCSS. **Chapter 445**: hierarquical clustering / dendrograms. Disponível em: `http://goo.gl/IXq5n3`. Acessado em: 24/09/2014.

OLIVEIRA, H. M. Seleção de entes complexos usando lógica difusa. **Instituto de Informática da PUC-RS**, Porto Alegre, RIo Grande do Sul, Brasil, 1996. Dissertação de mestrado.

RIBACKI, G. H. Um framework para agrupamento de dados. **Instituto de Informática, Ciência da Computação, Universidade Federal do Rio Grande do Sul**, Porto Alegre, Rio Grande do Sul, Brasil, 2013.

RUBINOV, A.; SOUKHOROKOVA, N.; UGON, J. Classes and clusters in data analysis. **European Journal of Operational Research**, P.O. Box 663, Ballarat, Vic. 3353, Australia, v.173, n.3, p.849–865, 2006.

SALTON, G.; MCGILL, M. **Introduction to modern information retrieval**. New York, USA: McGraw-Hill, 1983. (McGraw-Hill computer science series).

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. 75 Arlington, Street, Suite 300, Boston, USA: Pearson Addison Wesley, 2006. (Pearson International Edition).

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition, 2nd Edition**. 525 B Street, Suite 1900, San Diego, CA 92101-4495, USA: Academic Press, 2002.

WIKIPEDIA. **DBSCAN**. Disponível em: `http://en.wikipedia.org/wiki/DBSCAN`. Acessado em: 22/09/2014.

WIVES, L. K. Um estudo sobre agrupamento de documentos textuais em processamento de informações não estruturadas usando técnicas de "clustering". **Universidade Federal**

do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação., Porto Alegre, Rio Grande do Sul, Brasil, 1999.

WIVES, L. K. Utilizando conceitos como descritores de textos para o processo de identificação de conglomerados (clustering) de documentos. **Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação.**, Porto Alegre, Rio Grande do Sul, Brasil, 2004.

ZHENG, Y. Clustering Methods in Data Miningwith its Applications in High Education. **International Conference on Education Technology and Computer - ICETC**, Lodz, Poland, v.43, p.203–209, 2012.

APÊNDICE A ENTRADA E SAÍDA DE DADOS NO *FRAMEWORK*

Ao realizar uma execução com os algoritmos do *framework* é necessário criar uma instância da classe *ClusteringProcess* e selecionar os algoritmos de seleção, similaridade, agrupamento de dados e métricas de avaliação que serão utilizados. Também é necessário indicar a origem dos dados, para que as estratégias implementadas no *framework* possam localizá-los.

Na Figura A.1 em (a) é apresentado o formato como os dados são enviados ao *framework* para análise. Todos os arquivos são colocados em uma mesma pasta, sem distinção entre eles, ficando a cargo das estratégias de *clustering* determinar quais os agrupamentos corretos para os objetos analisados. Em (b) é mostrado um resultado de agrupamento. A divisão dos objetos em grupos é exibida ao usuário em formato textual, onde o número indica quantos objetos o grupo possui e na sequencia quais objetos pertencem a este grupo. Esse agrupamento foi obtido para o conjunto de dados *wikipedia13* ao ser utilizado o algoritmo de agrupamento de dados *Best-Star* com GSM igual a 0.06. Por fim, em (c) são exibidos os resultados das métricas de avaliação para os aglomerados calculados em (b).

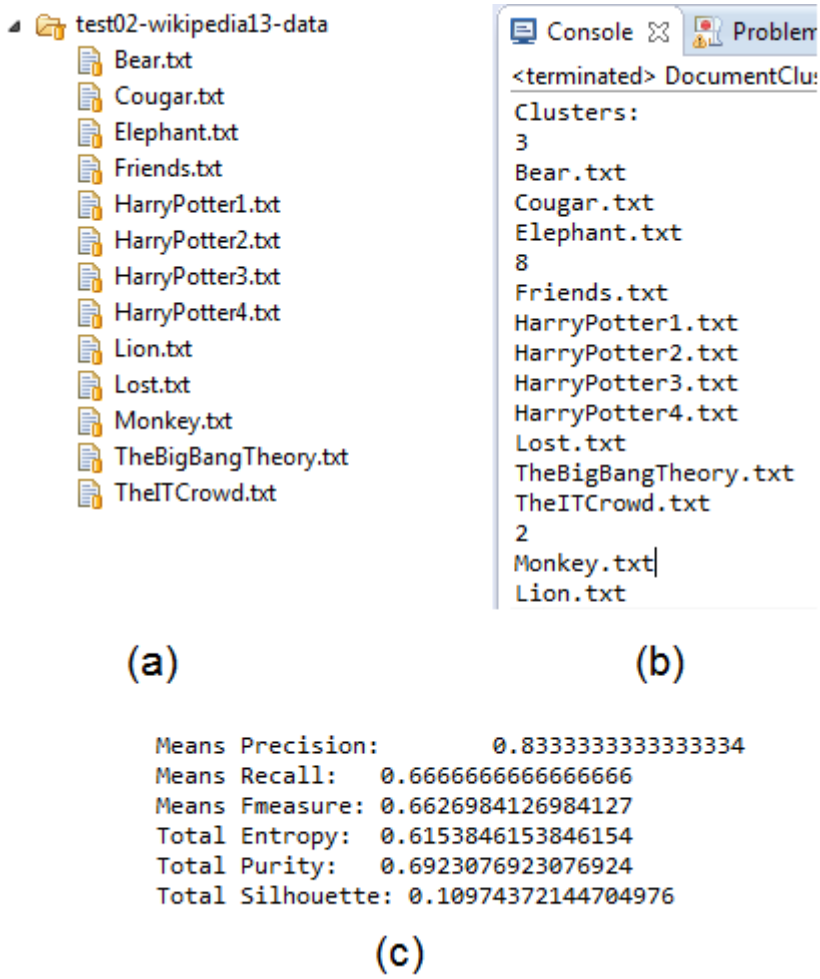

Figura A.1: (a) Entrada de dados no *framework*; (b) Resultado do algoritmo de agrupamento; (c) Resultado das métricas de avaliação

APÊNDICE B CONSTRUÇÃO DE UMA APLICAÇÃO USUÁRIA

Na página do *GitHub*[1] do projeto, estão disponíveis o *framework* propriamente dito (pasta *URSAFramework*) e uma aplicação usuária padrão (pasta *UserProject*). A aplicação usuária já está previamente configurada e pronta para extensão do *framework*. Na Figura B.1 pode-se observar a estrutura padrão da aplicação usuária.

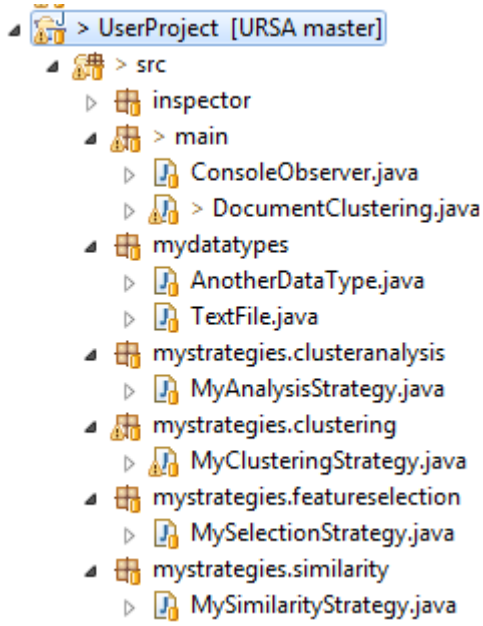

Figura B.1: Estrutura padrão da aplicação usuária

As classes *MyAnalysisStrategy.java*, *MyClusteringStrategy.java*, *MySelectionStrategy.java* e *MySimilarityStrategy.java* permitem que o usuário implemente as suas próprias estratégias de seleção de características, cálculo de similaridades, agrupamento de dados ou de validação de resultados, podendo utilizar estas no processo de agrupamento, exclusivamente com as suas estratégias implementadas, ou em uma mescla entre as suas estratégias e as nativas do *framework*.

[1] `https://github.com/jlggross/URSA`. Acessado em: 12/10/2014

Cada uma dessas classes citadas já possui os métodos essenciais para o seu funcionamento, os quais devem ser devidamente implementados. Na Figura B.2 pode-se observar os métodos pendentes de implementação na classe *MyClusteringStrategy.java*.

```java
public class MyClusteringStrategy extends ClusteringStrategy {

    private double threshold;

    /**
     * Definition: My algorithm constructor
     *
     * @param threshold : indicates the minimum similarity that an
     * object need to be part of a cluster.
     */
    public MyClusteringStrategy(double threshold) {
        this.threshold = threshold;
    }


    /**
     * Definition: My clustering algorithm execution.
     *
     * @param dataObjects : list of data objects.
     * @param similarityMatrix : similarity matrix with the similarity between every pair of objects.
     */
    public List<DataCluster> executeClustering(List<DataObject> dataObjects, Matrix2D similarityMatrix) {

        List<DataCluster> dataClusters = new ArrayList<DataCluster>();
        // TODO
        return dataClusters;
    }
}
```

Figura B.2: Classe *MyClusteringStrategy.java*

Logo, para esta classe, basta implementar o método *executeClustering()* para que a estratégia possa ser usada no processo de agrupamento de dados.

Também é possível implementar um tipo de dados próprio. A classe *AnotherDataType.java* pode ser alterada conforme as necessidades do novo tipo de dados. A Figura B.3 apresenta mais detalhes.

```java
public class AnotherDataType extends DataObject {

    public AnotherDataType() {
        // TODO
    }

    // TODO
    // Specific methods

    public String toString() {
        StringBuilder str = new StringBuilder();
        for (DataFeature feature : this.featureCollection) {
            str.append(feature.toString()).append("\n");
        }
        return str.toString();
    }
}
```

Figura B.3: Classe *AnotherDataType.java*

Por fim, o processo de agrupamento ocorre nos métodos desenvolvidos na classe *DocumentClustering.java*. Ali estão disponíveis diversos exemplos de execução, então basta

alterar um dos métodos conforme o novo tipo de dados desenvolvido, escolher as estratégias que serão usadas no processo de agrupamento e indicar a localização dos dados.