



UNIVERSIDADE FEDERAL DO ACRE
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

WEKAPAR: UMA EXTENSÃO DA FERRAMENTA WEKA PARA AUXILIAR O
PÓS-PROCESSAMENTO DE REGRAS DE ASSOCIAÇÃO

RIO BRANCO

2016

DANIEL AUGUSTO NUNES DA SILVA

**WEKAPAR: UMA EXTENSÃO DA FERRAMENTA WEKA PARA AUXILIAR O
PÓS-PROCESSAMENTO DE REGRAS DE ASSOCIAÇÃO**

Monografia apresentada como exigência final para obtenção do grau de bacharel em Sistemas de Informação da Universidade Federal do Acre.

Orientador: Prof. Me. Daricélio Moreira Soares

RIO BRANCO

2016

TERMO DE APROVAÇÃO

DANIEL AUGUSTO NUNES DA SILVA

WEKAPAR: UMA EXTENSÃO DA FERRAMENTA WEKA PARA AUXILIAR O PÓS-PROCESSAMENTO DE REGRAS DE ASSOCIAÇÃO

Esta monografia foi apresentada como trabalho de conclusão de Curso de Bacharelado em Sistemas de Informação da Universidade Federal do Acre, sendo aprovado pela banca constituída pelos membros abaixo mencionados.

Compuseram a banca:

Prof. Me. Daricélio Moreira Soares (Orientador)
Curso de Bacharelado em Sistemas de Informação

Prof. Dr. Macilon Araújo Costa Neto
Curso de Bacharelado em Sistemas de Informação

Prof. Me. Manoel Limeira de Lima Júnior
Curso de Bacharelado em Sistemas de Informação

Rio Branco – AC, 24 de junho de 2016

*Dedico este trabalho às mulheres que são minha fonte
diária de inspiração: Josiane, Heloísa e Isabela.*

AGRADECIMENTOS

Além de uma oportunidade para expressar gratidão aos que contribuíram para esta conquista, escrever os agradecimentos do trabalho de conclusão de curso é também uma atividade de reflexão. Assim, registro neste espaço meu reconhecimento àqueles que foram importantes não somente para a realização deste trabalho, mas também no decorrer da graduação.

Agradeço a minha família pelas inúmeras oportunidades nas quais pude contar com a ajuda necessária para me dedicar às atividades acadêmicas. E agradeço especialmente a minha esposa, Josiane, pela dedicação e apoio incondicional que tornou possível a concretização deste projeto.

Ao orientador, professor Daricélio Moreira Soares, pelas valiosas contribuições e pelos conhecimentos compartilhados, ajudando a fazer destes meses de trabalho um período de grande aprendizado. Agradeço também aos professores Jair Figueiredo e Catarina Costa, pelo apoio nas disciplinas de TCC I e II. E manifesto ainda minha gratidão aos demais docentes do curso por ajudar a desenvolver competências de grande relevância para o desenvolvimento deste trabalho de pesquisa, bem como para vida profissional.

Aos colegas do curso de Sistemas de Informação, pelo companheirismo nos muitos desafios enfrentados ao longo do curso.

A todos, meus sinceros agradecimentos.

*"O homem deve criar as oportunidades e não somente
encontrá-las." (Francis Bacon)*

RESUMO

A descoberta de regras de associação é uma importante tarefa de mineração de dados, cujo caráter descritivo indica a importância da etapa de análise e interpretação dos resultados. Um dos principais problemas desta etapa de pós-processamento é o elevado número de padrões obtidos, que decorre, principalmente, do grande volume de dados disponível de onde pretende-se obter as regras. Outros problemas relacionados a esta etapa incluem a existência de regras redundantes ou ainda regras que não representam informações úteis. Na ferramenta de mineração de dados WEKA não há nenhum recurso nativo para lidar com estes problemas. Neste contexto, o presente trabalho apresenta uma proposta de solução integrada ao WEKA que fornece recursos para análise dos padrões descobertos. A solução foi implementada na forma de uma extensão para WEKA com funcionalidades de visualização e filtragem das regras geradas. Os resultados obtidos com a validação das funcionalidades implementadas e a avaliação realizada com usuários, demonstram que a solução cumpre com o objetivo principal de permitir maior facilidade ao especialista de domínio na etapa de pós-processamento de regras de associação.

Palavras-chave: Pós-processamento de regras de associação. Mineração de dados. WEKA.

ABSTRACT

The discovery of association rules is an important data mining task, whose descriptive disposition indicates the importance of the analysis and interpretation step. One of the main problems of the post-processing step is the high number of obtained patterns that occur mainly due to the large volume of available data from where the rules are obtained. Other problems related to this step include redundant rules or rules that do not represent useful information. In WEKA data-mining tool there is no native feature to deal with these problems. This paper proposes a solution integrated with WEKA that provides resources for analysis of discovered patterns. The solution was implemented as an extension for WEKA with features for visualization and filtering of generated rules. The results obtained from the validation of the implemented functionalities and the evaluation applied out to users reveals that the solution satisfies the main objective, which is providing an easier approach for the domain expert in the post-processing step of association rules.

Key-words: Post-processing of association rules. Data mining. WEKA

LISTA DE FIGURAS

FIGURA 1 – USO DAS FERRAMENTAS DE MINERAÇÃO DE DADOS	18
FIGURA 2 – VISÃO GERAL DAS ETAPAS DO PROCESSO DE KDD.....	25
FIGURA 3 – REGRAS DE ASSOCIAÇÃO GERADAS NO AMBIENTE DO R	37
FIGURA 4 – VISUALIZAÇÃO DE REGRAS DE ASSOCIAÇÃO NO RATTLE	38
FIGURA 5 – CONSTRUINDO UM PROCESSO NO RAPIDMINER	40
FIGURA 6 – EXIBIÇÃO DE REGRAS DE ASSOCIAÇÃO NO RAPIDMINER	40
FIGURA 7 – WEKA EXIBINDO AS REGRAS GERADAS PELO APRIORI.....	42
FIGURA 8 – O MÓDULO <i>KNOWLEDGEFLOW</i> DA FERRAMENTA WEKA.....	43
FIGURA 9 – ENGENHARIA DE SOFTWARE ORIENTADA A REÚSO.....	49
FIGURA 10 – PROCESSO DE ENGENHARIA DE REQUISITOS	51
FIGURA 11 – PROCESSO DE EXTRAÇÃO DE CONHECIMENTO	55
FIGURA 12 – CÓDIGO QUE IMPLEMENTA O <i>PLUG-IN</i> DE VISUALIZAÇÃO.....	57
FIGURA 13 – ESTRUTURA DO PACOTE	60
FIGURA 14 – O ARQUIVO <i>EXPLORER.PROPS</i>	61
FIGURA 15 – O ARQUIVO <i>DESCRIPTION.PROPS</i>	61
FIGURA 16 – TRECHO DO ARQUIVO DE CONFIGURAÇÃO DO ANT	62
FIGURA 17 – OPÇÃO DE VISUALIZAÇÃO DAS REGRAS	63
FIGURA 18 – VISÃO GERAL DA GUIA DE PÓS-PROCESSAMENTO	64
FIGURA 19 – MENSAGEM DE ALERTA SOBRE MEMÓRIA DISPONÍVEL.....	66
FIGURA 20 – BARRA DE STATUS E <i>LOGS</i>	67
FIGURA 21 – EXEMPLO DE RESULTADO DA APLICAÇÃO DO FILTRO	71
FIGURA 22 – APLICAÇÃO DO FILTRO APÓS REDUÇÃO DE MÉTRICAS	72

FIGURA 23 – NÚMERO DE REGRAS E VALORES DE MÉTRICAS75

LISTA DE QUADROS

QUADRO 1 – VANTAGENS E DESVANTAGENS DO REÚSO DE SOFTWARE ...	48
QUADRO 2 – REQUISITOS FUNCIONAIS	55
QUADRO 3 – EXEMPLOS DE TERMOS UTILIZADOS NOS FILTROS	59
QUADRO 4 – ATRIBUTOS UTILIZADOS NA EXTRAÇÃO DAS REGRAS	69

LISTA DE TABELAS

TABELA 1 – EXEMPLO DE UMA BASE DE TRANSAÇÕES DE COMPRAS	29
TABELA 2 – CONJUNTOS DE ITENS FREQUENTES	30
TABELA 3 – REGRAS EXTRAÍDAS DE UMA BASE TRANSACIONAL	30
TABELA 4 – EXEMPLOS DE REGRAS DE ASSOCIAÇÃO.....	35
TABELA 5 – EXEMPLO DE REGRA DE ASSOCIAÇÃO REDUNDANTE	35
TABELA 6 – SELEÇÃO DE REGRAS COM BASE EM ATRIBUTOS.....	73
TABELA 7 – FILTROS COM VALORES COMBINADOS DE MÉTRICAS.....	76
TABELA 8 – VARIAÇÃO DAS MÉTRICAS AO LONGO DO TEMPO	77
TABELA 9 – SUBCONJUNTOS DE REGRAS QUE INDICAM A REJEIÇÃO DE UM <i>PULL REQUEST</i>	79
TABELA 10 – SUBCONJUNTOS DE REGRAS QUE INDICAM A ACEITAÇÃO DE <i>UM PULL REQUEST</i>	80
TABELA 11 – REGRA INVERSA RATIFICANDO A CORRELAÇÃO ENTRE OS LADOS DA REGRA	81
TABELA 12 – REGRA INVERSA EVIDENCIANDO REGRAS IRRELEVANTES....	82
TABELA 13 – RESULTADO DO QUESTIONÁRIO DE AVALIAÇÃO	83

SUMÁRIO

1	INTRODUÇÃO.....	14
1.1	PROBLEMA DA PESQUISA	15
1.2	OBJETIVOS DA PESQUISA	16
1.2.1	Objetivo geral.....	16
1.2.2	Objetivos específicos	16
1.3	JUSTIFICATIVA DA PESQUISA.....	17
1.4	METODOLOGIA	18
1.5	ORGANIZAÇÃO DO ESTUDO.....	20
2	MINERAÇÃO DE DADOS E REGRAS DE ASSOCIAÇÃO	22
2.1	PROCESSO DE KDD	23
2.2	TAREFAS DE MINERAÇÃO DE DADOS	25
2.3	MINERAÇÃO DE REGRAS DE ASSOCIAÇÃO	27
2.3.1	Processo de extração das regras	29
2.3.2	Medidas de interesse.....	31
2.3.3	A etapa de pós-processamento	33
2.4	FERRAMENTAS DE MINERAÇÃO DE DADOS	36
2.4.1	A linguagem R e a ferramenta Rattle.....	37
2.4.2	RapidMiner	39
2.4.3	WEKA	41
2.5	CONCLUSÃO	43
3	ENGENHARIA DE SOFTWARE	45
3.1	PROCESSO DE SOFTWARE	46

3.2	REÚSO DE SOFTWARE	47
3.3	ENGENHARIA DE REQUISITOS	50
3.4	VERIFICAÇÃO E VALIDAÇÃO DE SOFTWARE.....	52
3.5	CONCLUSÃO	53
4	WEKAPAR: POST-PROCESSING OF ASSOCIATION RULES FOR WEKA	54
4.1	IMPLEMENTAÇÃO.....	57
4.2	CRIAÇÃO DO PACOTE PARA WEKA	59
4.3	VISÃO GERAL.....	63
4.3.1	Integração com a WEKA	65
4.4	CONCLUSÃO	67
5	AVALIAÇÃO DA SOLUÇÃO PROPOSTA.....	68
5.1	VALIDAÇÃO DAS FUNCIONALIDADES IMPLEMENTADAS.....	69
5.1.1	Base de dados.....	69
5.1.2	Aplicação de filtros.....	70
5.1.2.1	Seleção de atributos.....	72
5.1.2.2	Medidas de interesse	74
5.1.2.3	Combinando filtros de atributos e métricas	77
5.1.2.4	Subconjunto de regras	78
5.1.2.5	Regra inversa	80
5.2	AVALIAÇÃO COM USUÁRIOS.....	82
5.3	CONCLUSÃO	84
6	CONSIDERAÇÕES FINAIS.....	86
6.1	CONTRIBUIÇÕES	87
6.2	RECOMENDAÇÕES.....	88
	REFERÊNCIAS.....	90
	APÊNDICES	93
	APÊNDICE A – DOCUMENTO DE REQUISITOS	94
	APÊNDICE B – RESULTADOS DO QUESTIONÁRIO DE AVALIAÇÃO	102

1 INTRODUÇÃO

Com a evolução das tecnologias da informação, um grande volume de dados vem sendo armazenados em diversos repositórios, criando um contexto onde o processo de descoberta de conhecimento se faz útil para a obtenção de novas informações a partir destes dados armazenados. Estas novas informações podem ter grande valor para as organizações, notadamente aquelas que necessitam conhecer melhor seu cliente e oferecer produtos e serviços de acordo com seu perfil.

Este processo de descoberta de conhecimento em bases de dados (*Knowledge-Discovery in Databases* – KDD) tem várias etapas. Dentre elas, a mineração de dados constitui-se como a principal, e consiste no processamento dos dados para obtenção de novas informações. Uma das técnicas utilizadas para este fim é a utilização de algoritmos para a extração de regras de associação, uma tarefa descritiva de mineração de dados.

A descoberta de regras de associação permite identificar determinados padrões em bases de dados, possibilitando, após a sua interpretação, adquirir conhecimento específico acerca do problema em análise. No entanto, a grande dimensão das bases de dados atuais leva a um elevado número de regras. Isso transforma a interpretação das regras em um novo problema. Apesar dos estudos desenvolvidos por alguns pesquisadores sobre o problema, não há ainda uma solução ideal.

Para realização das tarefas de mineração de dados, considerando sua complexidade, é imperativo a utilização de softwares para auxiliar no processo.

Existem diversas ferramentas disponíveis, incluindo softwares de uso comercial, de uso livre e, também, de código aberto. Alguns sistemas gerenciadores de banco de dados também oferecem funcionalidades para a mineração de dados. Uma das opções de ferramenta disponível é o WEKA (*Waikato Environment for Knowledge Analysis*), que reúne em um ambiente integrado vários recursos para realizar as etapas do processo de KDD.

Este projeto propõe, portanto, otimizar o processo de interpretação dos resultados na descoberta de regras de associação, fornecendo ao minerador de dados¹ uma solução integrada a ferramenta WEKA, que permita maior facilidade no trabalho de pós-processamento de regras de associação.

1.1 PROBLEMA DA PESQUISA

Ao analisar os resultados da extração de regras de associação, o trabalho do minerador de dados pode demandar muito tempo quando é necessário analisar um número elevado de regras extraídas. O problema decorre, dentre outros fatores, do grande volume de dados disponível de onde pretende-se obter conhecimento.

O minerador de dados busca, portanto, a utilização de ferramentas no intuito de auxiliar a filtragem destas regras, por intermédio de parâmetros, de forma a obter um conjunto de regras relevantes para o domínio estudado. A análise e interpretação dos resultados é uma etapa importante do processo de KDD, uma vez que é comum a existência de redundâncias ou relações irrelevantes nos padrões descobertos.

No WEKA, considerando a tarefa de extração de regras de associação, os recursos para análise dos padrões descobertos são limitados, principalmente no tocante a visualização das regras extraídas. Isto obriga o usuário a buscar outras ferramentas para complementar o trabalho de mineração de dados iniciado no WEKA.

¹ O termo minerador de dados é utilizado neste trabalho como sinônimo de especialista de domínio, que utiliza técnicas e ferramentas de mineração de dados, além de seu conhecimento do domínio, na descoberta de informações úteis (HAN; KAMBER; PEI, 2011; LIU, 2011; MACCUE, 2007; TAN; STEINBACH; KUMAR, 2006).

Neste contexto, questiona-se: como permitir ao minerador de dados, utilizador da ferramenta WEKA, mais facilidade no pós-processamento de regras de associação, fornecendo recursos para análise dos padrões descobertos?

1.2 OBJETIVOS DA PESQUISA

O objetivo geral da pesquisa, bem como seus objetivos específicos são apresentados nesta seção.

1.2.1 Objetivo geral

Desenvolver uma extensão para a ferramenta de mineração de dados WEKA que permita ao minerador de dados a seleção de regras de associação por intermédio de parâmetros na etapa de pós-processamento.

1.2.2 Objetivos específicos

Para atingir o objetivo geral, têm-se os seguintes objetivos específicos:

- Realizar revisão bibliográfica acerca da mineração de dados, pós-processamento de regras de associação e a engenharia de software;
- Elicitar requisitos e especificar a solução a ser desenvolvida;
- Implementar em linguagem de programação a ferramenta proposta;
- Executar o processo de verificação e validação.

1.3 JUSTIFICATIVA DA PESQUISA

O processo de KDD pode ser aplicado a várias áreas de conhecimento, não somente aquelas relacionadas as tecnologias da informação, mas também nas engenharias, ciências humanas, da saúde, dentre outras. É, portanto, um instrumento que permite a descoberta de conhecimento em diferentes domínios. A extração de conhecimento representa, atualmente, uma importante aliada no processo de tomada de decisão, estando muitas vezes incorporada a um Sistema de Apoio à Decisão (SAD).

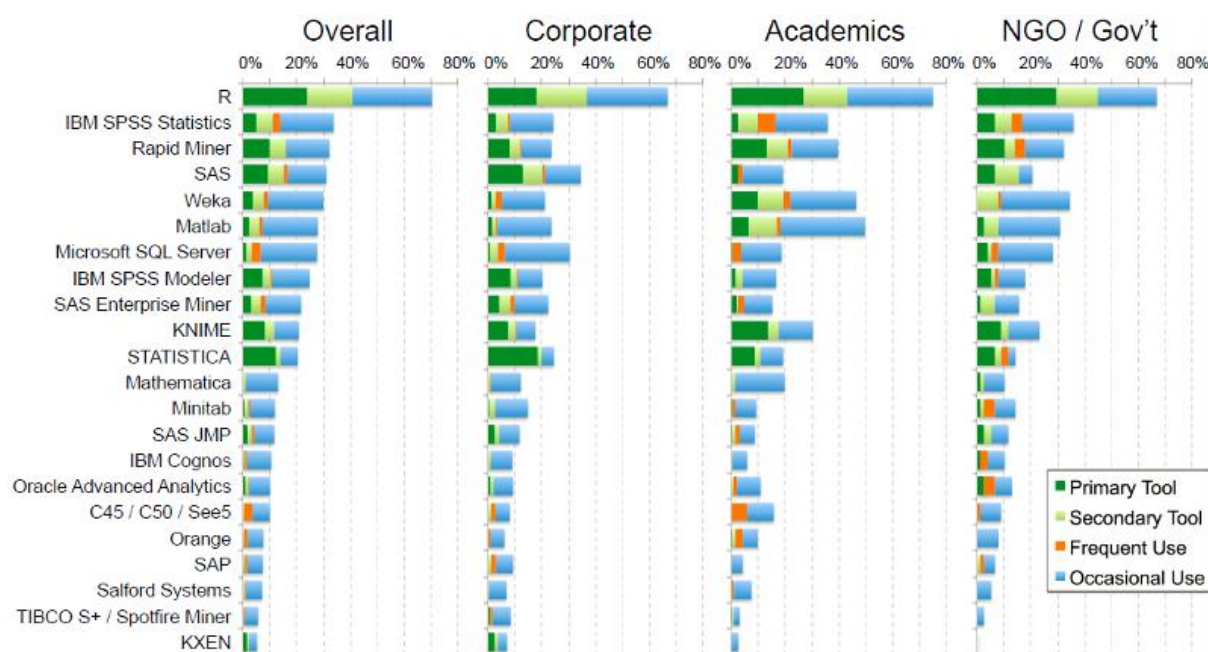
A descoberta de novos padrões representados pelas regras de associação é uma das tarefas mais comuns no processo de KDD, mas pode trazer algumas dificuldades ao processo. Considerando o domínio estudado e o volume de dados, há a possibilidade de o número de regras geradas ser suficientemente grande para dificultar ou inviabilizar a análise pelo especialista de domínio. Desta forma, o minerador de dados precisa trabalhar com ferramentas que o auxiliem no processo de análise das regras obtidas. O pós-processamento de regras de associação figura, portanto, como uma etapa importante no processo de descoberta de conhecimento.

Dentre as ferramentas disponíveis para mineração de dados, o WEKA é uma opção com forte presença no mercado, figurando entre soluções proprietárias produzidas por grandes empresas como a IBM e a Oracle, mesmo sendo um software de código aberto distribuído gratuitamente. A ferramenta, além de ser uma das mais utilizadas, é também muito difundida no meio acadêmico, como pode ser verificado na Figura 1. É neste domínio que se concentra a maior parte de seus usuários que a utilizam como principal ferramenta para realizar tarefas de mineração de dados (REXER, 2013).

Uma das funções da ferramenta WEKA é a extração de conhecimento utilizando algoritmos para gerar regras de associação. No que diz respeito ao pós-processamento destes padrões descobertos, a ferramenta não dispõe de muitos recursos para facilitar a visualização e análise das regras. Por ser um software de código é aberto, isso facilita o surgimento de extensões que complementam as funções das ferramentas, como é o caso deste trabalho. No entanto, não há ainda

outras extensões semelhantes que implementam recursos para o pós-processamento de regras de associação.

Figura 1 – Uso das ferramentas de mineração de dados



Fonte: Rexer (2013, p. 31).

1.4 METODOLOGIA

Em relação aos aspectos metodológicos desta pesquisa, esta pode ser classificada segundo as formas clássicas elencadas por Silva e Menezes (2005):

- É uma pesquisa aplicada, do ponto de vista de sua natureza, pois tem aplicação prática na solução de um problema específico;
- Sobre a forma de abordagem do problema, é classificada como uma pesquisa qualitativa, uma vez que consiste em um processo descritivo, que identifica o problema e avalia a solução de forma qualitativa, e também é classificada como uma pesquisa quantitativa, na medida em que utiliza dados estatísticos para explicar alguns dos resultados obtidos;

- c) Do ponto de vista dos objetivos, é uma pesquisa exploratória, considerando tratar-se de um estudo de caso que acrescenta à ferramenta WEKA funcionalidades com intuito de solucionar o problema de pesquisa estabelecido;

Este trabalho de pesquisa foi dividido em quatro etapas:

1. A primeira etapa consistiu em uma revisão bibliográfica dos conceitos e etapas do processo mineração de dados e descoberta de conhecimento em bases de dados, onde foram detalhadas as fases que integram o processo, bem como as técnicas normalmente aplicadas. Nesta etapa também foi realizada uma pesquisa sobre os processos, métodos e técnicas da Engenharia de Software.
2. Na segunda etapa foram executadas atividades relacionadas a engenharia de requisitos para fins de especificação da extensão desenvolvida.
3. Na terceira etapa, a solução proposta foi implementada em linguagem de programação, de forma integrada a ferramenta WEKA, por meio dos recursos disponibilizados pela ferramenta para criação de extensões.
4. A última etapa consistiu na validação da solução proposta, onde foi utilizado um conjunto de dados de teste, que permitiu a análise da utilização das funcionalidades implementadas. Também fez parte da validação a aplicação de um questionário de avaliação com usuários da ferramenta WEKA.

Para o desenvolvimento da solução, além da própria ferramenta WEKA² e sua API (*Application Programming Interface*), os principais recursos utilizados foram a linguagem de programação Java³ e o IDE (*Integrated Development Environment*) Eclipse⁴.

² <http://www.cs.waikato.ac.nz/ml/weka/>

³ <https://www.oracle.com/java/>

⁴ <https://eclipse.org/>

Na avaliação foram utilizadas bases de dados de repositórios de software do GitHub⁵ para demonstrar o funcionamento dos recursos desenvolvidos. A aplicação do questionário para avaliar a solução proposta teve como participantes um grupo de alunos do curso de Pós-graduação em Desenvolvimento de Software e Infraestrutura para Internet, da UFAC.

1.5 ORGANIZAÇÃO DO ESTUDO

No que diz respeito a organização do estudo, além desta introdução, onde está contextualizada a problemática abordada, o trabalho é composto por mais 5 capítulos.

No Capítulo 2, é apresentado um referencial teórico acerca da mineração de dados e pós-processamento de regras de associação. Este capítulo aborda o processo de descoberta de conhecimento em bases de dados, as tarefas mais comuns em mineração de dados, com destaque a mineração de regras de associação e a etapa de pós-processamento. Além disso, também são apresentados alguns exemplos de ferramentas de mineração de dados e seus recursos para extração e pós-processamento de regras de associação, incluindo a ferramenta WEKA.

No Capítulo 3, são abordados conceitos relacionados à Engenharia de Software, incluindo o desenvolvimento orientado a reuso, o processo de engenharia de requisitos e a verificação e validação de software.

A solução proposta neste trabalho é apresentada no Capítulo 4, que aborda também aspectos de implementação e integração com a ferramenta WEKA.

O Capítulo 5 consiste na demonstração das funcionalidades implementadas para fins de avaliação da solução proposta. Neste capítulo, é discutido ainda a avaliação de acordo com a perspectiva de usuários em relação aos recursos disponíveis para o pós-processamento de regras de associação.

⁵ <https://github.com/>

Por fim, o Capítulo 6 é composto pelas considerações finais sobre o trabalho de pesquisa realizado, onde são discutidos o cumprimento dos objetivos estabelecidos, as contribuições mais relevantes deste trabalho, bem como recomendações de trabalhos futuros.

2 MINERAÇÃO DE DADOS E REGRAS DE ASSOCIAÇÃO

Tradicionalmente, um processo manual de análise e interpretação de dados pode ser considerado uma forma de transformar estes dados em conhecimento. No entanto, para muitos domínios esta forma manual torna-se impraticável, na medida em que o volume de dados armazenados cresce exponencialmente (FAYYAD; PIATETSKY-SHAPIO; SMYTH, 1996a). Com a consolidação de soluções avançadas para banco de dados, incluindo sistemas de *data warehouse*⁶, além de sua constante evolução, criou-se um ambiente propício para obtenção de novas informações a partir dos dados armazenados. Assim, a mineração de dados pode ser entendida como resultado de um processo evolutivo das tecnologias da informação (HAN; KAMBER; PEI, 2011).

A literatura define mineração de dados como o processo de descoberta de informações úteis em grandes repositórios de dados (HAN; KAMBER; PEI, 2011; TAN; STEINBACH; KUMAR, 2006; WITTEN; FRANK; HALL, 2011). Este processo é realizado por meio de técnicas cuja finalidade é identificar, nos dados, padrões que poderiam permanecer desconhecidos (TAN; STEINBACH; KUMAR, 2006).

Diante da impossibilidade de continuar aplicando procedimentos essencialmente manuais, surgem processos sistemáticos de transformação de dados

⁶ *Data warehouse* é uma coleção de dados coletados de várias fontes, armazenados de forma não volátil, integrada, variável com o tempo e orientado por assunto, com o objetivo de apoiar a tomada de decisão em nível gerencial (INMON, 2002).

em novos conhecimentos aplicados por meio das ferramentas de mineração de dados, ou seja, é exatamente o grande volume de dados disponíveis que exige a aplicação de processos mais consistentes. Os dados armazenados, sem a devida análise, não configuram nenhuma informação útil. Sem a utilização de técnicas e ferramentas para extração e análise de dados, o tomador de decisão conta apenas com sua intuição e experiência, considerando a impossibilidade de analisar todos os dados (HAN; KAMBER; PEI, 2011).

O objetivo deste capítulo é apresentar os conceitos que fundamentam este trabalho. São abordados o processo de descoberta de conhecimento em bases de dados, as tarefas de mineração de dados, com ênfase na extração de regras de associação e suas medidas de interesse, o pós-processamento de regras de associação e, ainda, exemplos de ferramentas de mineração de dados.

2.1 PROCESSO DE KDD

A grande quantidade de dados disponíveis para serem analisados caracteriza um ambiente favorável à aplicação do que a literatura define como o processo de descoberta de conhecimento em bases de dados (*Knowledge-Discovery in Databases*, KDD). É possível notar uma forte aproximação com o conceito que define mineração de dados. Han, Kamber e Pei (2011) destacam que muitos tratam o processo de KDD e a mineração de dados como sinônimos. No entanto, a mineração de dados pode ser vista como uma etapa do processo de KDD, mas certamente figura como a etapa mais importante. Fayyad, Piatetsky-Shapiro e Smyth (1996b, p. 40) definem o KDD como sendo um “processo de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis embutidos nos dados”.

De acordo com Witten, Frank e Hall (2011), os padrões descobertos por meio deste processo devem ser relevantes na medida em que possam representar alguma vantagem, geralmente de natureza econômica.

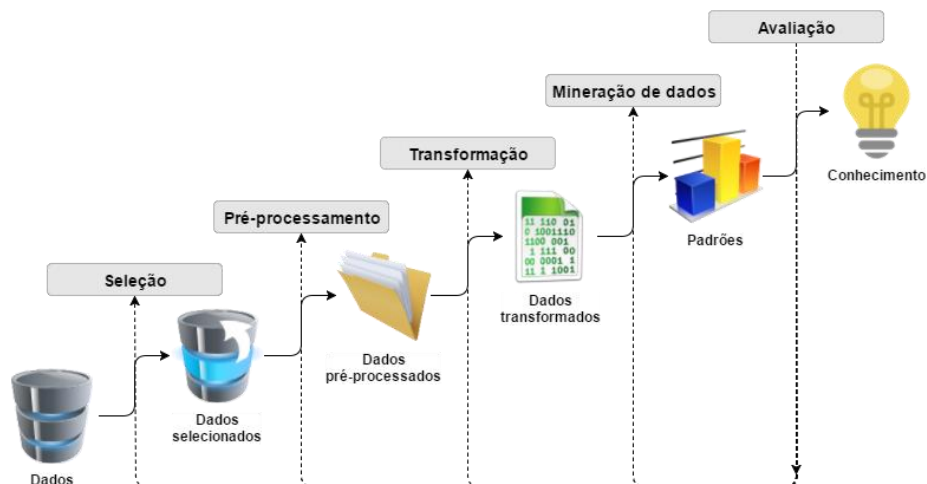
Segundo Fayyad, Piatetsky-Shapiro e Smyth (1996b), o processo de KDD compreende uma sequência de etapas, a saber:

- a) **Seleção:** etapa inicial do processo, onde são coletados os dados, que exige um conhecimento mínimo do domínio estudado para uma melhor seleção do subconjunto de dados;
- b) **Pré-processamento:** nesta etapa é quando ocorre a limpeza dos dados, executando técnicas como o tratamento de valores ausentes;
- c) **Transformação:** consiste na manipulação dos dados com o objetivo de enriquecê-los, através de tarefas como agrupamento de dados, transformação de tipos, normalização utilizando intervalos definidos, dentre outras;
- d) **Mineração de dados:** a partir do conjunto de dados selecionado é aplicado um algoritmo que tem por finalidade descobrir padrões;
- e) **Interpretação e avaliação:** etapa final do processo onde o especialista de domínio verifica se os resultados realmente contribuem para a solução do problema estabelecido, identificando a qualidade dos padrões descobertos, com o auxílio das métricas adequadas a cada caso.

A Figura 2 apresenta uma visão geral das etapas que compõe o processo de KDD. É possível identificar na figura os aspectos iterativo e iterativo do processo. É iterativo por envolver muitas decisões feitas pelo minerador de dados em cada etapa. É também iterativo devido a necessidade de retornar a uma das etapas anteriores quando os resultados obtidos não foram satisfatórios, seja para escolher um algoritmo diferente, aplicar outras técnicas de tratamento dos dados ou até mesmo realizar uma nova seleção dos dados. Assim, durante o processo podem ser realizadas várias iterações até que os objetivos sejam alcançados.

O processo de KDD é também visto como uma atividade multidisciplinar que agrega técnicas que não estão ligadas somente a uma disciplina específica (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b). Este aspecto multidisciplinar da mineração de dados contribui de forma significativa para que o processo seja aplicado com sucesso em um número cada vez mais amplo de domínios (HAN; KAMBER; PEI, 2011).

Figura 2 – Visão geral das etapas do processo de KDD



Fonte: Adaptado de Fayyad, Piatetsky-Shapiro e Smyth (1996b).

Em termos gerais, o processo de KDD pode ser aplicado em qualquer área onde seja identificada a necessidade de analisar um grande volume de dados e que, pelos motivos elencados neste trabalho, não é possível fazê-lo de forma manual. Fayyad, Piatetsky-Shapiro e Smyth (1996b) observam que, dentre as principais aplicações do KDD, estão aquelas relacionadas às atividades de negócios, tais como marketing, finanças, detecção de fraudes, telecomunicações, dentre outras. Han, Kamber e Pei (2011) destacam a inteligência de negócios (*Business Intelligence – BI*) como uma das áreas onde o KDD é aplicado com sucesso. Sem a aplicação do processo de KDD, bem como das técnicas e ferramentas da mineração de dados, as empresas não seriam capazes de analisar de forma eficaz o mercado, seus clientes, concorrentes e, principalmente, tomar decisões inteligentes sobre seus negócios.

2.2 TAREFAS DE MINERAÇÃO DE DADOS

A escolha das técnicas de mineração de dados que devem ser utilizados nas aplicações do processo de KDD depende do domínio da aplicação e também dos objetivos que se pretende alcançar. A mineração de dados possui uma série de tarefas, onde cada uma pode encontrar um tipo de padrão diferente que são, normalmente, classificadas em duas categorias: preditiva e descritiva (HAN; KAMBER; PEI, 2011; TAN; STEINBACH; KUMAR, 2006).

Uma tarefa preditiva tem por objetivo prever o valor de um atributo baseado nos valores de outros atributos. É aplicada quando se deseja conhecer o comportamento futuro de novas instâncias de dados. Para o atributo cujo valor se pretende prever é dado o nome de variável dependente, enquanto que os atributos usados para fazer a predição são chamados de variáveis independentes (TAN; STEINBACH; KUMAR, 2006).

As tarefas descritivas são utilizadas quando se deseja apenas apresentar os dados de uma forma compreensível. O objetivo é derivar padrões, encontrando relações nos dados analisados (TAN; STEINBACH; KUMAR, 2006).

Dentre as tarefas de mineração de dados, podem ser destacadas a classificação, o agrupamento e as regras de associação. A primeira utiliza uma abordagem preditiva e outras duas são descritivas.

A classificação mapeia, dentro de um conjunto pré-definido de classes, um conjunto de dados de entrada em uma classe de saída, estabelecendo uma função que tem por objetivo determinar a qual classe pertence o conjunto de dados informados. O processo é dividido em duas etapas: treino e teste. Na primeira etapa (treino) um modelo de classificação é construído utilizando um determinado subconjunto de dados, e na segunda etapa (teste) o modelo é utilizado para prever as classes de um subconjunto de dados distinto daquele utilizado na primeira etapa. Um exemplo de classificação é aplicado a bancos que diferenciam seus clientes quanto a possibilidade de fornecer crédito, classificando-os, por exemplo, entre conceder ou não conceder crédito a partir de seu histórico bancário ou capacidade de pagamento (HAN; KAMBER; PEI, 2011).

O agrupamento (ou *clustering*) tem por objetivo encontrar um conjunto de categorias que possam descrever os dados, sendo que em cada grupo os elementos são semelhantes entre si e diferentes dos outros grupos. É, portanto, uma tarefa descritiva. Han, Kamber e Pei (2011) definem o agrupamento como um processo de particionamento de um determinado conjunto de dados em subconjuntos. Uma aplicação típica da tarefa de agrupamento está relacionada a área de marketing, onde empresas buscam agrupar seus consumidores com base em seu perfil, definido pelos

dados de compras disponíveis em sua base de dados. Desta forma, as empresas podem direcionar sua estratégia de marketing para um determinado grupo.

As regras de associação foram inicialmente apresentadas por Agrawal, Imieliński e Swami (1993), consistindo numa tarefa que identifica padrões no próprio conjunto de dados, caracterizando o quanto a ocorrência de um conjunto de atributos determina a ocorrência de outro conjunto distinto. A tarefa foi demonstrada através de um modelo que descrevia o comportamento de clientes ao analisar os itens presentes nas compras de supermercados, com intuito de verificar as relações que possam identificar produtos que influenciam a compra de outros. Por exemplo, uma regra **Pão** \rightarrow **Leite** indica que parte dos clientes que compram pão também compram leite. Assim, uma vez identificado estes padrões, o tomador de decisão pode utilizar estas informações para tentar influenciar a compra de determinados produtos, como por exemplo, dispondo caixas de leite próximas dos pães. Da mesma forma, uma loja virtual pode recomendar aos seus visitantes produtos relacionados com aqueles pelos quais tem interesse ou mesmo que já foram adquiridos. A análise de transações de compras é um exemplo clássico de regras de associação, mas a técnica também pode ser aplicada em diversas áreas.

2.3 MINERAÇÃO DE REGRAS DE ASSOCIAÇÃO

A mineração de regras de associação é uma tarefa fundamental de mineração de dados. Tem por objetivo encontrar todas as relações possíveis de ocorrência simultânea entre itens em uma base de dados. Dentre os modelos criados para extração de padrões em bases de dados, a mineração de regras de associação é um dos mais importantes e mais estudados (LIU, 2011).

Formalmente, uma regra de associação é definida como uma representação da relação entre itens de uma base de dados que ocorrem com uma certa frequência. Seja um conjunto de itens $I = \{i_1, i_2, \dots, i_n\}$ de uma base de dados D , uma regra de associação é uma implicação da forma $X \rightarrow Y$, onde $X \subset I$, $Y \subset I$ e $X \cap Y \neq \emptyset$. O lado esquerdo da regra (X) é denominado antecedente e o lado direito (Y), consequente. A

regra $X \rightarrow Y$ possui em D um suporte, que representa o percentual de transações em D que contém $X \cup Y$, em relação ao total de transações (T) em D , isto é, satisfazem tanto o antecedente quanto consequente da regra, conforme demonstrado na Equação (1).

$$Suporte(X \rightarrow Y) = \frac{T_{X \cup Y}}{T} \quad (1)$$

A regra $X \rightarrow Y$ possui também em D uma confiança (Equação (2)) que representa o percentual de transações em D que contém X e também Y em relação as transações que contém apenas X (AGRAWAL; SRIKANT, 1994).

$$Confiança(X \rightarrow Y) = \frac{Suporte(X \cup Y)}{Suporte(X)} \quad (2)$$

As medidas de suporte e confiança tem finalidades diferentes, mas que se complementam. O suporte corresponde à significância estatística da regra, ou seja, pode indicar sua relevância para o conjunto de dados. A confiança é uma medida que mensura a força da regra, indicando sua validade (AGRAWAL; IMIELNISKI; SWAMI, 1993).

O conceito de regras de associação está originalmente relacionado com padrões obtidos de bases de dados transacionais. Uma regra transacional contém apenas um único predicado, como por exemplo itens de compras. No entanto, supondo que além dos produtos comprados a base de dados também armazena dados de seus clientes, tais como idade, profissão ou estado civil, as regras obtidas poderão conter outros tipos de atributos que, por sua vez, podem assumir diferentes valores ou rótulos, representando novas categorias de dados. Estas são características do conceito de regras de associação multidimensionais (HAN; KAMBER; PEI, 2011).

Por definição, semelhante a formulação original, sendo D uma base de dados relacional, uma regra de associação multidimensional é uma implicação na forma $X_1 \wedge X_2 \wedge \dots \wedge X_n \rightarrow Y_1 \wedge Y_2 \wedge \dots \wedge Y_m$, onde $X_i (i = 1, \dots, n)$ e $Y_j (j = 1, \dots, m)$ representam atributos distintos em D (HAN; KAMBER; PEI, 2011).

Para simplificar o entendimento, as etapas do processo de extração de regras de associação são demonstradas utilizando um exemplo de base de dados transacionais. No entanto, o problema abordado nesse trabalho, bem como a proposta de solução, também considera a geração de regras multidimensionais.

2.3.1 Processo de extração das regras

Liu (2011) destaca que a literatura apresenta um grande número de algoritmos de mineração de regras de associação que resultam em diferentes níveis de eficiência. No entanto, a formulação é baseada na mesma definição. Normalmente, o que diferencia um algoritmo de outro é o seu custo computacional. O algoritmo de mineração mais conhecido é o *Apriori*, proposto por Agrawal e Srikant (1994).

O processo de extração de regras de associação pode ser dividido em duas etapas: identificar os conjuntos frequentes, que atendem ao suporte mínimo, e, para cada conjunto encontrado, gerar regras com a confiança mínima (AGRAWAL; IMIELIŃSKI; SWAMI, 1993; HAN; KAMBER; PEI, 2011). Para exemplificar esse processo considere o conjunto de dados composto por itens de uma transação de compra, conforme **Erro! Fonte de referência não encontrada.** A frequência de ocorrência de um determinado conjunto de itens é o número de transações nas quais consta este conjunto de itens.

Tabela 1 – Exemplo de uma base de transações de compras

#	Itens
1	Café, Leite, Pão
2	Café, Pão
3	Leite, Pão
4	Leite, Pão
5	Café, Leite, Pão

Fonte: Elaboração própria.

A primeira etapa do processo de extração de regras de associação consiste em encontrar todos os conjuntos de itens frequentes a partir dos dados destas transações. O resultado da primeira etapa do processo são conjuntos de itens que ocorrem na

base de dados até o limite mínimo de suporte estabelecido. Supondo que o valor mínimo de suporte é 60%, é possível encontrar 5 conjuntos de itens frequentes na base de dados, conforme demonstrado na Tabela 2.

Tabela 2 – Conjuntos de itens frequentes

#	Conjunto	Suporte
1	Pão	5/5 = 100%
2	Leite	4/5 = 80%
3	Café	3/5 = 60%
4	Pão, Leite	4/5 = 80%
5	Pão, Café	3/5 = 60%

Fonte: Elaboração própria.

Com base nos itens frequentes encontrados, a segunda etapa do processo consiste em gerar as regras de associação, respeitando o limite de confiança escolhido. Uma vez que os itens frequentes utilizados para gerar as regras são aqueles que já satisfazem o suporte mínimo, a regra, por sua vez, também satisfaz esta restrição. Considerando uma confiança mínima de 80%, três regras de associação podem ser geradas, como mostrado na Tabela 3.

Tabela 3 – Regras extraídas de uma base transacional

#	Regra	Suporte	Confiança
1	<i>Leite</i> → <i>Pão</i>	4/5 = 80%	(4/5) / (4/5) = 100%
2	<i>Café</i> → <i>Pão</i>	3/5 = 60%	(3/5) / (3/5) = 100%
4	<i>Pão</i> → <i>Leite</i>	4/5 = 80%	(4/5) / (5/5) = 80%

Fonte: Elaboração própria.

A segunda etapa deste processo é a que exige menos recursos computacionais. Desta forma, é a execução da primeira etapa que determina o desempenho geral do processo de mineração de regras de associação (HAN; KAMBER; PEI, 2011).

2.3.2 Medidas de interesse

A mineração de regras de associação costuma basear-se no modelo onde a regras encontradas possuam valores de suporte e confiança iguais ou superiores aos valores mínimos estabelecidos pelo minerador de dados. Valores elevados de suporte e confiança podem indicar que a regra gerada é relevante para o domínio estudado. Portanto, suporte e confiança são considerados medidas de interesse, uma vez que o especialista de domínio pode identificar regras fortes a partir da observação destes dois índices.

Contudo, mesmo que o suporte e a confiança eliminem um bom número de regras que não são relevantes, o conjunto restante ainda pode conter muitas regras que não são do interesse do minerador de dados (HAN; KAMBER; PEI, 2011). Estas duas medidas de interesse, em muitos casos, não são suficientes para eliminar regras irrelevantes. Neste sentido, existem outras medidas de interesse que podem ajudar a identificar regras mais relevantes.

O *lift* é um exemplo de medida de correlação simples utilizada na mineração de regras de associação, servindo como medida de interesse. Dada uma regra de associação $X \rightarrow Y$, os dois lados da regra podem ser dependentes e estar correlacionados. Quando o *lift* da regra apresenta um valor menor do que 1, indica que a ocorrência de X está negativamente correlacionada com a ocorrência de Y . Por conseguinte, um valor maior que 1 indica que a ocorrência de X implica a ocorrência de Y , isto é, estão positivamente correlacionados. O valor igual a 1 indica que o antecedente e o conseqüente da regra são independentes e não estão correlacionados (HAN; KAMBER; PEI, 2011). Portanto, o *lift* mostra o quanto a ocorrência de X torna mais frequente a ocorrência de Y , sendo que quanto maior o valor da medida, maior é a relevância da regra. A medida é dada dividindo a confiança da regra pelo suporte do conseqüente, como verificado na Equação (3). O resultado é um valor entre 0 e $+\infty$.

$$Lift(X \rightarrow Y) = \frac{Confiança(X \rightarrow Y)}{Suporte(Y)} \quad (3)$$

O *leverage* (ou *Rule Interest*, RI) é outro exemplo de medida de interesse. É calculado pela diferença entre o suporte real e o suporte esperado da regra (Equação (4)). Indica a proporção de transações adicionais na base de dados que abrangem tanto o antecedente como o conseqüente, além do esperado, se estes fossem independentes (WITTEN; FRANK; HALL, 2011). Para este índice, a independência é verificada quando o resultado é igual a 0. Neste caso, há indicação de que ambos os lados da regra ocorrem juntos, mas dentro do esperado. Um resultado positivo denota que antecedente e conseqüente ocorreram juntos um número de vezes maior que o esperado, e, de forma oposta, um valor negativo indica que a ocorrência de ambos os lados da regra juntos é menor que o esperado.

$$Leverage(X \rightarrow Y) = Confiança(X \rightarrow Y) - (Suporte(X) \times Suporte(Y)) \quad (4)$$

As medidas *lift* e *leverage* possuem uma propriedade em comum: são medidas simétricas. Isto é, o valor deste tipo de medida de interesse é igual tanto para a regra $X \rightarrow Y$ como para $Y \rightarrow X$. A razão para que isto ocorra consiste em que medidas simétricas avaliam a dependência entre os itens. De maneira oposta, as medidas assimétricas podem mensurar valores diferentes para $X \rightarrow Y$ e $Y \rightarrow X$, pois avaliam a regra como uma implicação (TAN; KUMAR; SRIVASTAVA, 2002; TAN; STEINBACH; KUMAR, 2006). A confiança é um exemplo de medida de interesse assimétrica.

Outro índice que tem como proposta avaliar a regra como uma implicação é a convicção. É uma medida assimétrica diferente da confiança no sentido que considera tanto o suporte do antecedente como o do conseqüente (BRIN et al., 1997). Se uma dada regra $X \rightarrow Y$ indica, com um determinado grau de certeza, que quando X está presente na transação Y também está, a convicção mede a probabilidade desta regra não estar correta, mensurando a frequência com que X ocorre sem Y . O objetivo é medir o quanto uma regra está distante da independência. Assim como o *lift*, o valor da convicção varia entre 0 e $+\infty$, onde o valor igual 1 indica que o antecedente e o conseqüente da regra são independentes, valor menor que 1 indica relação negativa, e quanto maior o valor, maior a relação entre os dois lados da regra. A Equação (5) demonstra como a medida confiança é obtida.

$$Convicção(X \rightarrow Y) = \frac{Suporte(X) \times Suporte(\neg Y)}{Suporte(X \cup \neg Y)} \quad (5)$$

Verifica-se, portanto, que cada medida de interesse pode trazer uma interpretação diferente para a regra analisada, e que o problema do grande número de regras irrelevantes gerados pelo modelo baseado nas medidas suporte e confiança, pode ser atenuado com a utilização de outras medidas de interesse. Assim, cabe ao minerador de dados decidir quais são os índices adequados para identificar os padrões relevantes na etapa de pós-processamento.

2.3.3 A etapa de pós-processamento

Uma vez descobertos os padrões, é desejável que além de novos e úteis, também sejam compreensíveis, sendo que muitas vezes isto não se dá de forma imediata, mas sim após algum trabalho de análise e interpretação (FAYYAD; PIATETSKY-SHAPIO; SMYTH, 1996b). Esta etapa de análise e interpretação, ou pós-processamento, é de fundamental importância no processo de KDD, uma vez que toda tarefa de mineração de dados demanda um processo de tomada de decisão, incluindo a mineração de regras de associação. Ademais, tarefas descritivas, como as regras de associação, são na maioria das vezes de natureza exploratória, obrigando o minerador de dados a utilizar técnicas de pós-processamento para explicar e validar os padrões extraídos (TAN; STEINBACH; KUMAR, 2006).

Quando o número de regras de associação geradas é pequeno, a análise dos resultados (pós-processamento) pode ser considerada uma tarefa simples. No entanto, o trabalho do minerador de dados pode demandar muito tempo quando necessita analisar um número elevado de regras extraídas. Um dos problemas em relação as regras de associação é que um grande volume de dados, como é comum nos dias atuais, leva a um elevado número de regras. Isto também se verifica quando valores de suporte e confiança estabelecidos não são suficientes para encontrar padrões interessantes, ao passo que reduzir estes valores implica em aumentar o número de regras extraídas. A relação entre tamanho da base de dados e número de

regras é explicada por uma característica das técnicas de mineração de dados que utilizam itens frequentes: se um dado conjunto de itens é frequente, todos os seus subconjuntos também são (HAN; KAMBER; PEI, 2011). Há, portanto, a possibilidade de o número de regras geradas ser suficientemente grande para dificultar ou inviabilizar a análise pelo especialista de domínio.

A literatura propõe vários métodos para realizar o pós-processamento os resultados da mineração de regras de associação. Em geral, existem duas abordagens para o problema: a abordagem objetiva e subjetiva. Analisar regras de associação objetivamente significa considerar suas medidas de interesse como índices estatísticos que medem a força ou relevância de uma dada regra. Desta forma, a abordagem objetiva é orientada aos dados (*data-driven*). No entanto, há o aspecto subjetivo desta análise. Uma mesma regra de associação pode ser considerada relevante para um especialista de domínio, mas não para outro. A abordagem subjetiva leva em consideração o conhecimento do minerador de dados para seleção as melhores regras. Trata-se, portanto, de uma abordagem orientada ao usuário (*user-driven*) (SILBERSCHATZ; TUZHILIN, 1996).

Considerando o elevado número de regras, o minerador irá buscar formas de explorar o conjunto de padrões encontrados, objetivando identificar os mais relevantes. Neste sentido, a combinação de medidas de interesse no processo de análise das regras pode ser mais eficaz do que analisar somente uma medida, pois cada uma delas é capaz de ressaltar uma propriedade diferente da regra analisada (GONÇALVES, 2004). Outrossim, as medidas de interesse possuem limitações, o que não está restrito ao problema do modelo suporte/confiança abordado anteriormente, pois cada medida pode ser eficaz para demonstrar a força de uma determinada regra, mas não para todas (TAN; STEINBACH; KUMAR, 2006). Desta forma, é razoável considerar a necessidade de ordenar as regras de acordo com cada índice para possibilitar a análise do padrão extraído considerando diferentes aspectos.

Outra estratégia que pode ser utilizada pelo minerador de dados é aplicar algum filtro nas regras. Trata-se de uma abordagem que pode diminuir consideravelmente o conjunto de regras analisado. A Tabela 4 apresenta alguns exemplos de regras de associação. Considerando que o especialista tem interesse somente em regras cujo

antecedente contenham apenas pão ou café, as regras 3 e 4 poderiam ser excluídas deste conjunto.

Tabela 4 – Exemplos de regras de associação

#	Regra
1	$P\tilde{a}o \rightarrow Leite$
2	$Caf\acute{e} \wedge P\tilde{a}o \rightarrow Leite$
3	$Caf\acute{e} \wedge Feij\tilde{a}o \rightarrow Arroz$
4	$Feij\tilde{a}o \rightarrow Arroz$

Fonte: Elaboração própria.

Dada uma regra de associação, podem existir dentro do conjunto de padrões extraídos outras regras semelhantes, mas que alguns de seus itens não exercem influência na importância da regra além do que ela já representa sem estes itens. Essas regras são consideradas redundantes, pois os itens da regra original são suficientes para justificar sua relevância, ao passo que adicionar mais itens torna a regra insignificante, uma vez que não acrescenta nenhuma informação ao padrão descoberto (KLEMETTINEN et al., 1994; LIU; HSU; MA, 1999). No exemplo da Tabela 5 a inclusão do leite no antecedente da regra não modificou os valores de suporte e confiança, demonstrando que a regra 2 é redundante.

Tabela 5 – Exemplo de regra de associação redundante

#	Regra	Suporte	Confiança
1	$P\tilde{a}o \rightarrow Caf\acute{e}$	0.2	0.4
2	$P\tilde{a}o \wedge Leite \rightarrow Caf\acute{e}$	0.2	0.4

Fonte: Elaboração própria.

Outro aspecto que ressalta a importância do pós-processamento de regras de associação é a variação temporal. Em geral, algoritmos para extração de regras de associação consideram implicitamente que os padrões descobertos podem se manter ao longo tempo, o que não é necessariamente verdade (BÖTTCHER; SPOTT; NAUCK, 2005). Ao abordar o exemplo da análise da cesta de compras, é possível identificar o aspecto temporal das bases de dados transacionais. Quando uma compra é registrada, o momento em que foi realizada também faz parte do registro. Desta forma, padrões descobertos na base de compras durante um determinado intervalo

de tempo não necessariamente representam todo o período compreendido na base de dados (ALE; ROSSI, 2000).

Assim, podemos considerar que algumas regras de associação se mantêm válidas para um determinado intervalo de tempo, mas não para outros (LI et al., 2003). Com isso, temos que, no que concerne a etapa de pós-processamento, deve ser uma preocupação do especialista de domínio avaliar se um determinado padrão considerado interessante se manterá confiável no futuro. Uma regra que não se sustenta ao longo do tempo pode perder sua utilidade (LIU; MA; LEE, 2001).

Tan, Steinbach e Kumar (2006) destacam ainda a visualização com uma característica importante para o pós-processamento de regras de associação, que requer um ambiente amigável para interação do especialista de domínio com o sistema de mineração de dados.

2.4 FERRAMENTAS DE MINERAÇÃO DE DADOS

A complexidade das técnicas de mineração de dados pressupõe a utilização de auxílio ferramental como um requisito imprescindível para realizar as tarefas relacionadas ao processo de descoberta de conhecimento em bases de dados. Para apoiar a aplicação destas técnicas, existem diversas ferramentas disponíveis, incluindo softwares de uso comercial, de uso livre e, também, de código aberto. Alguns sistemas gerenciadores de banco de dados também oferecem funcionalidades para a mineração de dados.

Nas subseções seguintes são apresentadas algumas ferramentas de mineração de dados. Além do WEKA, são abordados o RapidMiner, a linguagem de programação R e ainda a ferramenta Rattle, destacando o suporte à mineração de regras de associação e a visualização dos padrões descobertos. Estas ferramentas estão entre as mais utilizadas por acadêmicos e profissionais de mineração de dados (REXER, 2015).

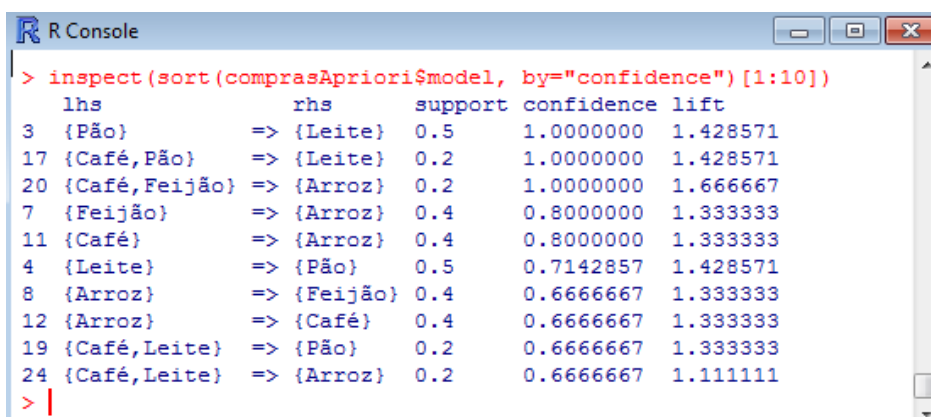
2.4.1 A linguagem R e a ferramenta Rattle

A linguagem de programação R tem como principal finalidade o processamento de rotinas para a realização de cálculos estatísticos (TORGO, 2011). Trata-se de um software livre distribuído não somente como linguagem de programação, mas como um ambiente sofisticado de computação estatística, fornecendo várias abordagens para mineração de dados (WILLIAMS, 2011).

No seu modo de operação básico, a ferramenta pode ser utilizada de forma interativa por meio da linha comando ou através de *scripts*, o que já é suficiente para executar muitos procedimentos úteis. No entanto, o ambiente suporta usos mais avançados, podendo ser utilizado para atividades de maior complexidade tais como escrever novas funções, programas e sistemas ou mesmo estender as funcionalidades da ferramenta, valendo-se da vantagem de ser um software de código aberto (TORGO, 2011; WILLIAMS, 2011).

A linguagem R não possui um recurso nativo para extração de regras de associação. No entanto, é possível fazê-lo por meio da extensão *arules*, que implementa o algoritmo Apriori (HAHSLER et al., 2011). A Figura 3 mostra um exemplo de regras de associação geradas no ambiente do R, ordenadas de forma decrescente pela métrica confiança.

Figura 3 – Regras de associação geradas no ambiente do R



```
> inspect(sort(comprasApriori$model, by="confidence")[1:10])
```

	lhs	rhs	support	confidence	lift
3	{Pão}	=> {Leite}	0.5	1.0000000	1.428571
17	{Café,Pão}	=> {Leite}	0.2	1.0000000	1.428571
20	{Café,Feijão}	=> {Arroz}	0.2	1.0000000	1.666667
7	{Feijão}	=> {Arroz}	0.4	0.8000000	1.333333
11	{Café}	=> {Arroz}	0.4	0.8000000	1.333333
4	{Leite}	=> {Pão}	0.5	0.7142857	1.428571
8	{Arroz}	=> {Feijão}	0.4	0.6666667	1.333333
12	{Arroz}	=> {Café}	0.4	0.6666667	1.333333
19	{Café,Leite}	=> {Pão}	0.2	0.6666667	1.333333
24	{Café,Leite}	=> {Arroz}	0.2	0.6666667	1.111111

```
> |
```

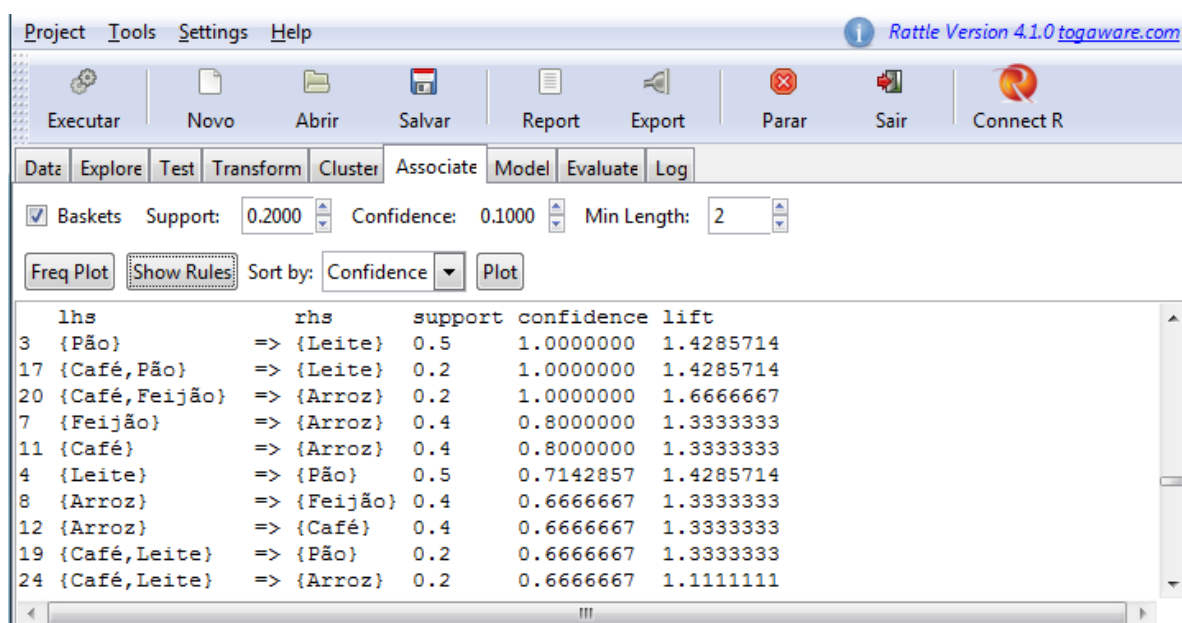
Fonte: Elaboração própria.

A utilização da linguagem R por meio da linha de comando pode se tornar uma tarefa complexa, principalmente para iniciantes na área de mineração de dados.

Soluções que implementam a linguagem R, baseadas em interface gráfica, ajudam a reduzir parte da complexidade (WILLIAMS, 2011). Uma destas soluções é o Rattle (*R Analytical Tool To Learn Easily*), um software de mineração de dados, baseado na linguagem R, desenvolvido com foco em usuários iniciantes (WILLIAMS, 2009).

A ferramenta Rattle é distribuída como um pacote da linguagem R, sendo executada dentro do próprio ambiente do R. Desta forma, o Rattle faz uso de vários recursos já disponíveis na linguagem R, tornando-os menos complexos de se utilizar por meio da interface gráfica. Embora não seja necessário conhecer a linguagem R para começar a trabalhar com o Rattle, a medida em que o usuário ganha mais experiência tende a migrar para o ambiente do R, buscando recursos mais sofisticados e com menos restrições (WILLIAMS, 2009, 2011). O Rattle também possui recurso para extração de regras de associação por meio da mesma extensão utilizada no ambiente do R. A diferença está na facilidade de ajustar os parâmetros utilizando os componentes da interface gráfica. A visualização das regras geradas é semelhante a linguagem R, como demonstrado na Figura 4.

Figura 4 – Visualização de regras de associação no Rattle



	lhs	rhs	support	confidence	lift
3	{Pão}	=> {Leite}	0.5	1.0000000	1.4285714
17	{Café, Pão}	=> {Leite}	0.2	1.0000000	1.4285714
20	{Café, Feijão}	=> {Arroz}	0.2	1.0000000	1.6666667
7	{Feijão}	=> {Arroz}	0.4	0.8000000	1.3333333
11	{Café}	=> {Arroz}	0.4	0.8000000	1.3333333
4	{Leite}	=> {Pão}	0.5	0.7142857	1.4285714
8	{Arroz}	=> {Feijão}	0.4	0.6666667	1.3333333
12	{Arroz}	=> {Café}	0.4	0.6666667	1.3333333
19	{Café, Leite}	=> {Pão}	0.2	0.6666667	1.3333333
24	{Café, Leite}	=> {Arroz}	0.2	0.6666667	1.1111111

Fonte: Elaboração própria.

É possível verificar, tanto no ambiente integrado da linguagem R como na ferramenta Rattle, que a forma de visualização das regras de associação pode dificultar o trabalho do minerador de dados, quando for necessário lidar com o

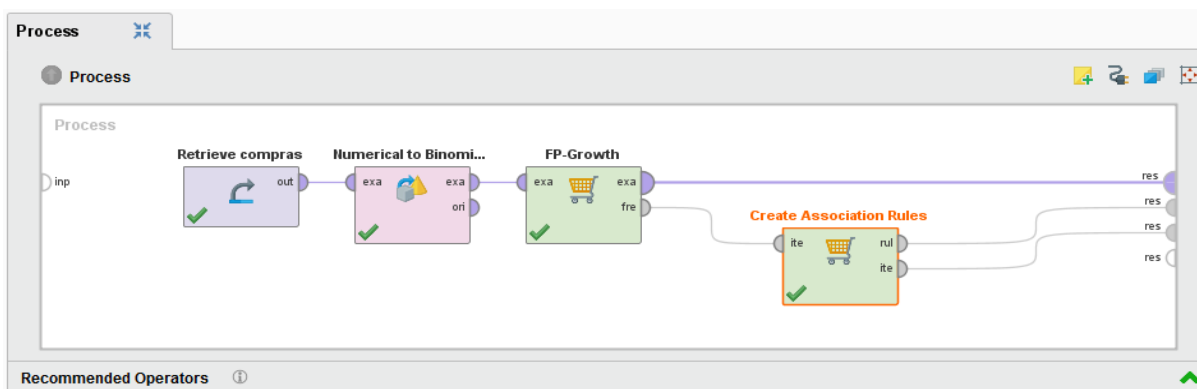
problema de analisar um grande número de padrões extraídos. Neste caso, as regras podem ser exportadas para análise em outras ferramentas, gravando as saídas dos comandos em arquivo, no caso do ambiente de linha de comando, ou exportando as regras no formato PMML (*Predictive Model Markup Language*), no caso do Rattle.

2.4.2 RapidMiner

O RapidMiner é um software de mineração de dados de código aberto e que fornece através de uma interface gráfica amigável recursos para aplicar desde as técnicas já consolidadas de mineração de dados até abordagens mais recentes (KOTU; DESHPANDE, 2015; RAPIDMINER, 2014). O software é mantido e distribuído pela empresa homônima e possui, além da versão gratuita, uma versão paga licenciada para uso comercial, que agrega mais recursos e serviço de suporte.

A ferramenta utiliza a abordagem do controle de fluxo de dados, na qual um processo é construído por meio da combinação de operadores que são graficamente representados na tela. No contexto da ferramenta, um operador é uma estrutura básica que recebe uma entrada, processa os dados e produz uma saída como resultado deste processamento, sendo que tanto a entrada como a saída podem estar ligadas a outros operadores. Internamente, a estrutura é formada por documentos XML (*eXtensible Markup Language*), mas é produzida pelo usuário por meio de operações de arrastar e soltar. Neste sentido, a solução oferece uma boa experiência de uso para o minerador de dados pois, além de verificar constantemente a construção do processo, indicando a ocorrência de erros, também auxilia no controle de cada etapa, permitindo que seja interrompida para análise dos resultados (RAPIDMINER, 2014). Esta abordagem sistemática permite ao minerador de dados maior controle sobre o processo (CHISHOLM, 2013). A Figura 5 mostra a criação de um processo, combinando operadores e estabelecendo um fluxo de trabalho.

Figura 5 – Construindo um processo no RapidMiner



Fonte: Elaboração própria.

Assim como a maioria das ferramentas de mineração de dados disponíveis, o RapidMiner também tem suporte à extração de regras de associação. No entanto, o algoritmo utilizado não é o Apriori. A opção nativa do RapidMiner para mineração de regras de associação é algoritmo FP-growth⁷. O Apriori pode ser utilizado por meio de uma extensão que adiciona recursos da fermenta WEKA ao RapidMiner. Um exemplo de associações geradas pela ferramenta é mostrado na Figura 6.

Figura 6 – Exibição de regras de associação no RapidMiner

The screenshot shows the 'AssociationRules (Create Association Rules)' window. On the left, there are tabs for 'Data', 'Graph', 'Description', and 'Annotations'. The 'Data' tab is active, showing a list of items: Leite, Arroz, Pão, Feijão, Café, and Manteiga. Below this, 'Min. Criterion' is set to 'confidence' and 'Min. Criterion Value' is shown as a slider. The main area displays a table of association rules.

No.	Premises	Conclusion	Support	Confidence ↓	Lift
86	Pão	Leite	0.500	1	1.429
87	Manteiga	Leite	0.100	1	1.429
88	Manteiga	Arroz	0.100	1	1.667
89	Manteiga	Feijão	0.100	1	2
90	Arroz, Pão	Leite	0.100	1	1.429
91	Manteiga	Leite, Arroz	0.100	1	3.333
92	Leite, Manteiga	Arroz	0.100	1	1.667
93	Arroz, Manteiga	Leite	0.100	1	1.429
94	Pão, Feijão	Leite	0.100	1	1.429
95	Pão, Café	Leite	0.200	1	1.429

Fonte: Elaboração própria.

A forma de visualização das regras de associação na ferramenta facilita a análise e a busca pelos padrões relevantes. A regras e suas métricas são mostradas

⁷ O algoritmo FP-growth consiste em uma alternativa ao método empregado no algoritmo Apriori, construindo uma estrutura de dados compactada (FP-tree) para minerar padrões frequentes, e obtendo resultados com maior velocidade em comparação ao Apriori (HAN; KAMBER; PEI, 2011).

em uma tabela, cujos dados podem ser ordenados e filtrados, selecionando os itens presentes no conseqüente das regras ou atribuindo valores mínimos para uma determinada métrica selecionada.

2.4.3 WEKA

Uma das opções de ferramenta disponível é o WEKA (*Waikato Environment for Knowledge Analysis*), que consiste em um software de uso livre, amplamente difundido no meio acadêmico, e mantido pela Universidade de Waikato⁸, Nova Zelândia.

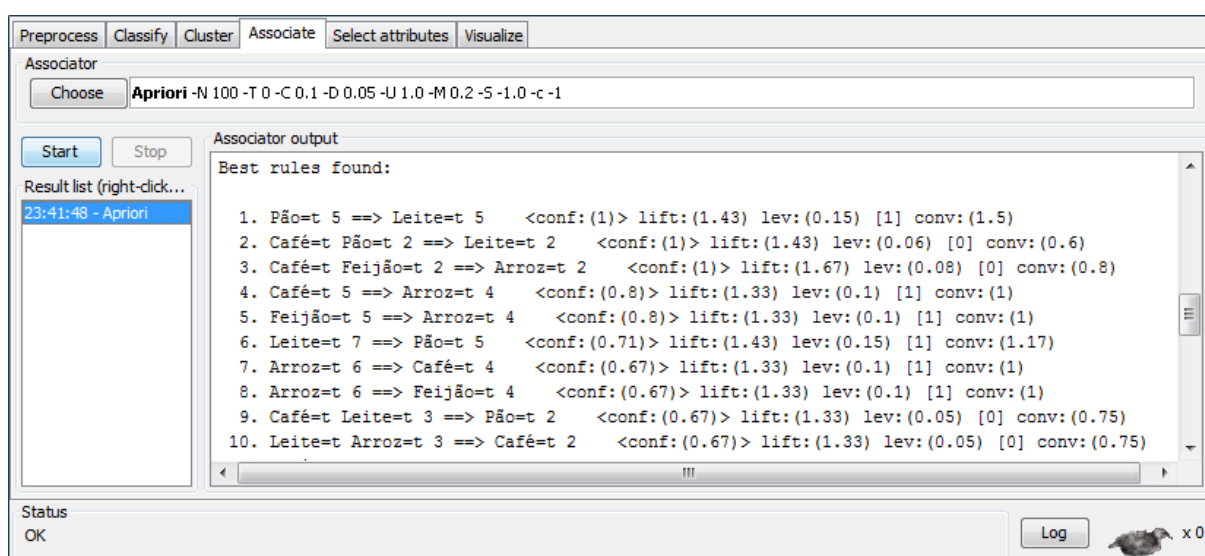
Originalmente, a ferramenta foi desenvolvida para o governo neozelandês, em 1993, para auxiliar a aplicação de técnicas de aprendizagem de máquina em áreas importantes da economia do país. As primeiras versões do software foram implementadas em linguagem C. No entanto, com o passar do tempo o projeto se tornou mais difícil de se manter, sobretudo pela complexidade em dar suporte a diferentes bibliotecas, gerenciar suas dependências, além do processo de configuração de ambiente ser pouco amigável. Foi então que a ferramenta foi reescrita na linguagem Java, incluindo todos os algoritmos que a compõe (HALL et al., 2009).

O WEKA disponibiliza uma coleção de algoritmos de aprendizagem de máquina, além de ferramentas para o pré-processamento dos dados. Oferece um conjunto de funcionalidades que podem ser acessados através de uma interface comum, de modo que o minerador de dados pode comparar diferentes métodos e identificar aqueles que são mais apropriados. Também inclui ferramentas para a transformação dos dados, tais como os algoritmos de discretização. Os principais problemas de mineração de dados são abordados pela ferramenta, dentre eles temos: a regressão, classificação, agrupamento, regras de associação e seleção de atributos (WITTEN; FRANK; HALL, 2011).

⁸ <http://www.waikato.ac.nz/>

No WEKA a tarefa de mineração de regras de associação é executada utilizando implementações de algoritmos como o Apriori e o FP-growth. A saída padrão destes algoritmos é exibida na tela no formato texto de forma semelhante a ferramenta Rattle, conforme exemplo demonstrado na Figura 7. Não há suporte nativo a outras formas de visualização das regras extraídas. A saída do algoritmo pode ser salva em arquivo para ser utilizada em outros softwares para fins análise dos padrões encontrados. No entanto, esta atividade exige que o minerador aplique algum tratamento para saída do algoritmo de forma a permitir que os dados possam ser lidos por outros softwares.

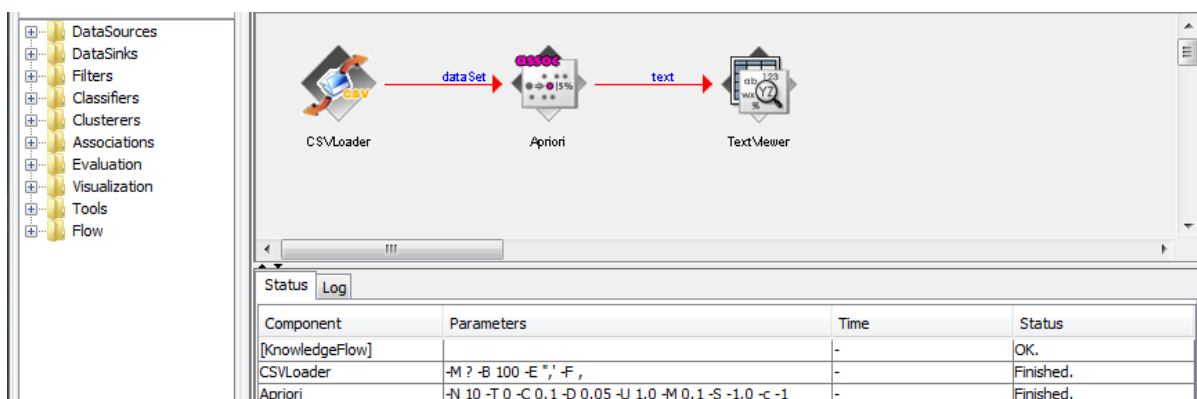
Figura 7 – WEKA exibindo as regras geradas pelo Apriori



Fonte: Elaboração própria.

Como alternativa ao módulo principal da ferramenta, o *Explorer*, o WEKA fornece um recurso para execução das tarefas de mineração de dados por meio da construção de um fluxo de trabalho, o *KnowledgeFlow*, como ocorre no RapidMiner. Todos os algoritmos disponíveis na ferramenta também podem ser utilizados neste módulo, incluindo o algoritmo para gerar regras de associação. No que diz respeito à visualização das regras, a saída do algoritmo é exibida de forma idêntica ao *Explorer*. Na Figura 8 é apresentado um exemplo de processo criado com o *KnowledgeFlow* para extração de regras de associação, dividido em três etapas: carregamento dos dados, processamento dos dados pelo algoritmo Apriori e visualização das regras.

Figura 8 – O módulo *KnowledgeFlow* da ferramenta WEKA



Fonte: Elaboração própria.

Para a maioria dos usuários, as funcionalidades disponíveis no WEKA são suficientes para realização de diversas tarefas de mineração de dados. No entanto, a ferramenta permite que novas funcionalidades sejam adicionadas, sob a forma de extensões, através um mecanismo interno de descoberta de novas classes que podem implementar desde novos algoritmos de classificação e de associação até modificações da interface gráfica do software. A principal tela da ferramenta, o *Explorer*, pode ser modificada, por exemplo, adicionando novas abas ou novas formas de visualização dos dados. A partir da versão 3.7.2, o WEKA permite que estas extensões sejam adicionadas a ferramenta por meio de “pacotes”, que consistem em arquivos com uma estrutura pré-definida, contendo todos os recursos necessários ao funcionamento da extensão (BOUCKAERT et al., 2014).

2.5 CONCLUSÃO

Neste capítulo, foi apresentado um referencial teórico acerca da mineração de dados, com ênfase na tarefa de regras de associação e na etapa de pós-processamento. O referencial apresentado nesta Capítulo inclui também alguns exemplos de ferramentas de mineração de dados e seus recursos para extração de regras de associação, bem como para análise e interpretação das regras. Buscou-se contextualizar a mineração de regras de associação dentro do processo de KDD, e também enfatizar a importância da etapa de pós-processamento, que inclui a interpretação das medidas de interesse.

No capítulo seguinte, dando continuidade ao referencial teórico, são abordados conceitos relacionados à Engenharia de Software, no intuito de fundamentar o desenvolvimento da solução proposta neste trabalho.

3 ENGENHARIA DE SOFTWARE

A Engenharia de Software engloba os vários aspectos da construção de um software, o que compreende algumas etapas e atividades, desde as especificações iniciais até sua manutenção (SOMMERVILLE, 2011). Este processo de construção consiste em empregar princípios de engenharia no intuito de produzir um software que atenda sobretudo aos requisitos de economia, confiabilidade e eficiência (PRESSMAN, 2011).

Sommerville (2011) destaca que normalmente é utilizada uma abordagem sistemática e organizada para o trabalho de produção de um software, objetivando alcançar maior eficiência no processo, bem como obter um produto de qualidade. Para Pressman (2011) esta abordagem engloba um processo, métodos e ferramentas, sendo que este processo se estabelece como a base para a Engenharia de Software.

Assim, este Capítulo aborda conceitos relacionados à Engenharia de Software importantes para a realização deste trabalho, organizados em quatro seções principais: processo de software, o desenvolvimento orientado a reuso, o processo de engenharia de requisitos e a verificação e validação de software.

3.1 PROCESSO DE SOFTWARE

Um processo de software é um conjunto de atividades correlatas, executadas de forma sistemática, cujo objetivo comum é o desenvolvimento ou evolução de um software. As atividades desenvolvidas no processo podem sofrer variações de acordo com o contexto, mas seguem premissas básicas (SOMMERVILLE, 2011).

Embora um processo de software defina todas as etapas do desenvolvimento de software, não podemos entendê-lo como sinônimo de engenharia de software. Isto é, a engenharia de software não pode ser resumida como uma metodologia para realização de atividades, ações ou tarefas, pois deve ser entendida como algo mais abrangente. Desta forma, o processo de software está relacionado com a engenharia de software na medida em que os profissionais, engenheiros de software, adaptam um modelo de processo de software maduro de forma a atender suas demandas. O processo é, portanto, responsável pela abordagem que será utilizada no desenvolvimento do software (PRESSMAN, 2011).

Dentre as atividades desenvolvidas no processo, algumas são consideradas fundamentais: especificação, projeto e implementação, validação e evolução. As especificações de software, também descrita na literatura como engenharia de requisitos, visa compreender e documentar o que o software deve possuir no tocante às funcionalidades, isto é, quais serviços deve executar. O projeto e implementação tem por objetivo efetivar em uma determinada linguagem de programação o que foi documentado durante o processo de engenharia de requisitos. A validação constitui uma etapa na qual se verifica se o que foi implementado está de acordo com os requisitos identificados. Já a evolução do software tem se tornado com passar do tempo cada mais útil, pois além da manutenção do software, novos requisitos podem surgir, o ambiente de negócios pode mudar, dentre outros fatores (SOMMERVILLE, 2011).

A execução destas atividades pode ser representada segundo modelos de processo de software, que retratam de forma abstrata o processo sob uma determinada perspectiva. De forma genérica, três abstrações de um processo de um software são descritas por (SOMMERVILLE, 2011):

- a) **o modelo em cascata:** é uma das abordagens mais antigas e, também, mais utilizadas no processo de software, representando as principais atividades como fases separadas dentro do processo, sendo que cada uma destas fases está encadeada a outra;
- b) **o desenvolvimento incremental:** é uma abordagem cujo o objetivo de desenvolver um sistema adequado às necessidades do cliente é alcançado por meio sucessivas versões do software aprimoradas a partir do retorno dado pelo usuário;
- c) **a engenharia de software orientada a reuso:** tem como premissa a existência de um número grande de componentes de software já desenvolvidos e validados que podem ser reusados em outro software, tendendo a ser um processo menos oneroso e incorrer em menos riscos.

Considerando que o processo de desenvolvimento é constituído por atividades que são repetidas à medida que os resultados ainda não são satisfatórios, os modelos de processo de software podem ter um carácter iterativo. Sommerville (2011) considera que o desenvolvimento iterativo é fundamental, e destaca dois modelos baseados no apoio a iteração do processo: a entrega incremental e o desenvolvimento espiral. A entrega incremental divide o desenvolvimento de forma a permitir que uma versão inicial do sistema seja entregue e depois sejam fornecidos incrementos, desenvolvidos um de cada vez. Segundo Pressman (2011), o desenvolvimento espiral é representado através de um espiral que elenca várias atividades, distribuídas entre as fases do processo, onde cada volta significa uma versão do software que será avaliada pelo cliente. As voltas no espiral continuarão à medida que o software vai sendo melhorado.

3.2 REÚSO DE SOFTWARE

Pressman (2011) considera que softwares de alta qualidade tem como característica importante possuir componentes reutilizáveis, e que o reuso destes componentes proporciona benefícios. Neste sentido, Sommerville (2011) conceitua o

desenvolvimento de software baseado em reuso como uma estratégia da engenharia de software que reutiliza um software já existente e seus componentes, cuja adesão por parte dos profissionais tem sido cada vez maior, principalmente em virtude da provável redução de custos que esta abordagem pode proporcionar.

Um outro aspecto que corrobora com o crescimento do emprego de reuso em projetos de software, é a grande quantidade de soluções de software reusáveis atualmente disponíveis. Muito deste crescimento se deve ao movimento *open source* que promoveu uma vasta coleção de código reusável na forma de bibliotecas de software ou aplicações completas. Outrossim, novos padrões que surgem favorecem a reutilização de software, tais como os *web services* (SOMMERVILLE, 2011).

Mesmo considerando a reusabilidade como um fator de qualidade de software, Pressman (2011) destaca dois aspectos que devem ser analisados no que diz respeito à decisão de reusar software: se um software já existente atende aos requisitos pode ser menos oneroso adquiri-lo do que desenvolver; e se este software exige que sejam realizadas modificações, o custo deve ser considerado com cautela pois pode incorrer e um valor além do que seria necessário para um novo projeto.

Neste sentido, Sommerville (2011) observa que há benefícios do reuso de software, como a redução dos custos de desenvolvimento, mas também pode haver problemas associados ao reuso, na medida em que esta redução de custos no desenvolvimento pode não compensar os custos relacionados a adequação do software para reuso. O Quadro 1 lista algumas vantagens e desvantagens do desenvolvimento de software baseado em reuso.

Quadro 1 – Vantagens e desvantagens do reuso de software

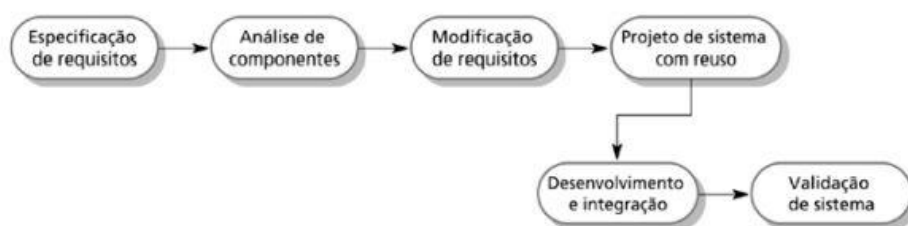
Vantagens	Desvantagens
<ul style="list-style-type: none"> • Softwares reusados já foram testados sob vários aspectos e provavelmente são mais confiáveis; • Risco de processo reduzido, na medida em que o custo do software existente já é conhecido, reduzindo a margem de erro de estimativa de custos; 	<ul style="list-style-type: none"> • Possibilidade de maiores custos de manutenção, pois os componentes reusáveis podem se tornar incompatíveis com futuras modificações; • Algumas ferramentas de software não suportam o desenvolvimento com reuso;

Vantagens	Desvantagens
<ul style="list-style-type: none"> • Evita repetir o mesmo trabalho, pois muitas funcionalidades já estão encapsuladas nos softwares reusáveis; • Conformidade com padrões, sobretudo de interface gráfica, pois a utilização de componentes reusáveis permite ao usuário reconhecer o mesmo padrão em todas as aplicações; • Redução do tempo de desenvolvimento e validação. 	<ul style="list-style-type: none"> • Alguns profissionais acreditam que escrever seu próprio código é mais confiável; • A criação, manutenção e uso de uma biblioteca de componente reusáveis pode demandar que processos de desenvolvimento sejam adaptados de forma a garantir que estes componentes sejam realmente utilizados; • O emprego de reuso envolve encontrar, compreender e, muitas vezes, adaptar os componentes reusáveis.

Fonte: Adaptado de Sommerville (2011, p. 297-298)

O modelo baseado em reuso se assemelha aos demais padrões de desenvolvimento no que diz respeito aos estágios inicial e final do processo, com a especificação de requisitos e o processo de validação. No entanto, o desenvolvimento de software baseado em reuso pressupõe atividades específicas como a análise dos componentes a serem reutilizados, modificações de requisitos para refletir os componentes disponíveis, além da integração destes componentes. A Figura 9 mostra uma representação do modelo baseado em reuso.

Figura 9 – Engenharia de Software orientada a reuso



Fonte: Sommerville (2011, p. 23).

O reuso de software também se aplica aos produtos COTS⁹ (*comercial-off-the-shelf*), seja pela configuração e customização do sistema, ou ainda pela integração

⁹ COTS é um tipo de sistema de software que possui recursos prontos para uso com aplicação em domínios específicos, tais como negócios, engenharia, medicina, dentre outros. Em geral, a maioria dos softwares do tipo *desktop* pode ser considerado um produto COTS (SOMMERVILLE, 2011).

com aplicações existentes. Em muitos casos a solução permite que as funcionalidades sejam estendidas por meio da criação de *plug-ins* (SOMMERVILLE, 2011).

Softwares de código aberto são também utilizados como softwares COTS. Mesmo que o código fonte esteja disponível, em muitos casos estes softwares são utilizados sem que sejam realizadas qualquer tipo de modificações, prevalecendo as funcionalidades disponíveis prontas para uso (TORCHIANO; MORISIO, 2004). Neste sentido, estas características também se aplicam ao WEKA, pois a ferramenta pode ser usada como um produto de prateleira com diversas funcionalidades já implementadas, mas também pode ser integrada a outros projetos ou receber novos recursos por meio de extensões.

3.3 ENGENHARIA DE REQUISITOS

Os requisitos de software consistem em um mapeamento de todos os serviços que devem ser fornecidos pelo software em questão, transformando as necessidades e objetivos segundo o entendimento do cliente ou usuário em funcionalidades do sistema. Os requisitos também podem representar limitações de caráter operacional do sistema. Este processo de identificar e documentar todos estes aspectos é denominado engenharia de requisitos (SOMMERVILLE, 2011).

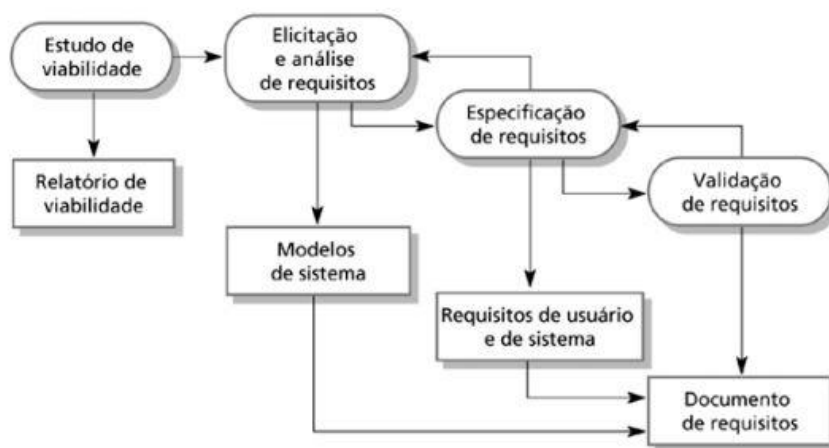
Pressman (2011) destaca que a definição dos requisitos de software é uma atividade que deve ser executada junto com o cliente, e que o resultado desta atividade é documento que descreve a especificação dos requisitos de software.

O documento de requisitos de software, ou a especificação de requisitos, é um documento oficial que descreve tudo que deve ser implementado pelos desenvolvedores, cujo conteúdo foi previamente acordado com o cliente. É recomendado que o documento seja rico em detalhes, principalmente se o cliente for externo e seu modelo de negócio seja desconhecido pela equipe de desenvolvedores (SOMMERVILLE, 2011).

Segundo Sommerville (2011), estes requisitos podem ser classificados em funcionais e não funcionais. Os funcionais são aqueles que representam o que o sistema deve fazer, como deve se comportar e, em muitas vezes, o que o sistema não pode fazer. Os não funcionais, por sua vez, não estão diretamente ligados às funcionalidades do sistema, estando, portanto, relacionados a outras propriedades do sistema como, por exemplo, espaço de armazenamento em disco, facilidade de uso, segurança, aspectos legais e de privacidade, dentre outros.

Da mesma forma que ocorre com os processos de software, onde os engenheiros adaptam modelos genéricos à realidade de seus projetos, um processo genérico de engenharia de requisitos também pode ser moldado para ser aplicado a um determinado projeto de software. Sommerville (2011) destaca que o processo de engenharia de requisitos deve ser composto por tarefas que incluem um estudo de viabilidade, a eliciação e análise de requisitos, a validação de requisitos, bem como seu gerenciamento. As atividades se relacionam dentro do processo, e cada uma traz um resultado produzindo seu próprio documento, conforme demonstrado na Figura 10.

Figura 10 – Processo de engenharia de requisitos



Fonte: Sommerville (2011, p. 24).

A primeira etapa consiste em determinar se vale a pena ou não dar continuidade ao projeto de desenvolvimento do software, através do estudo de viabilidade. Uma vez identificada a viabilidade do projeto, os requisitos serão obtidos por meio do processo de eliciação e análise, que pressupõe o envolvimento de todos os interessados no projeto, especificados em um documento, e depois validados para

verificar se realmente definem as demandas do usuário. A Figura 10 demonstra também o aspecto iterativo da elicitação e análise de requisitos, cujo processo irá se repetir quantas vezes for necessário para que se produza um documento de requisitos que reflete a demanda dos usuários (SOMMERVILLE, 2011).

A engenharia de requisitos é, portanto, uma etapa fundamental do processo de engenharia de software. Uma vez que esta fase seja bem executada, o planejamento nela estabelecido irá representar uma vantagem durante a execução do projeto. Considerando que todas funcionalidades e restrições do sistema foram bem discutidas, a implementação destes requisitos será um processo menos custoso se comparado a um projeto mal planejado, mesmo que ocorram alterações nos requisitos durante o processo.

3.4 VERIFICAÇÃO E VALIDAÇÃO DE SOFTWARE

No decorrer do processo de desenvolvimento de um software é comum a utilização de testes para verificar se as funcionalidades implementadas atendem ao especificado. No entanto, os testes podem ser entendidos como técnicas que compõem um processo mais abrangente denominado verificação e validação. Este processo pode ser aplicado desde a fase inicial do projeto até a avaliação final do software. É importante destacar a distinção entre verificação e validação, uma vez que a verificação consiste em avaliar se o software está de acordo com as especificações, enquanto que a validação tem por objetivo certificar que as expectativas dos usuário foram atendidas (SOMMERVILLE, 2011).

Desta forma, para que um software seja considerado adequado à sua proposta inicial, é necessário considerar aspectos que dizem respeito tanto às funcionalidades implementadas quanto às expectativas dos usuários. Para verificar estes aspectos, duas abordagens podem ser adotadas: a inspeção e o teste de software. A inspeção incide em examinar os artefatos de software do projeto, que pode incluir documentos de requisitos, diagramas e código fonte, buscando por inconsistências que podem incorrer em defeitos no sistema. O teste de software consiste em executar o software,

ou uma parte dele, afim de verificar se a saída corresponde ao esperado (SOMMERVILLE, 2011).

Paula Filho (2009) destaca que a atividade de revisão dos artefatos produz maior eficácia ao processo de verificação e validação se comparado aos testes, pois evidencia os defeitos de forma direta. Os testes mostram apenas sinais de algum problema no software, sendo necessário valer-se da atividade de inspeção do código para determinar a causa do problema. No entanto, os testes são importantes para o processo de desenvolvimento de um software.

A validação do software deve combinar tanto a realização de testes de sistema quanto o atendimento às expectativas dos usuários. Os testes incluem a avaliação do funcionamento do software em seu ambiente operacional, o qual deve simular o ambiente definitivo, e testes de aceitação preliminares. A aceitação definitiva depende da atividade de avaliação de uso, onde o usuário pode validar o produto desenvolvido sob sua própria perspectiva (PAULA FILHO, 2009).

3.5 CONCLUSÃO

Neste capítulo, foram apresentados conceitos relacionados à Engenharia de Software que compõem o referencial teórico deste trabalho. Estes conceitos foram relevantes no desenvolvimento da solução para o problema de pesquisa estabelecido, que envolveu o processo de engenharia de requisitos, o reúso de software, uma vez que se trata de uma extensão da ferramenta WEKA, bem como a verificação e validação da solução. A extensão desenvolvida, portanto, é resultado também da aplicação destes e conceitos, sendo apresentada no próximo Capítulo.

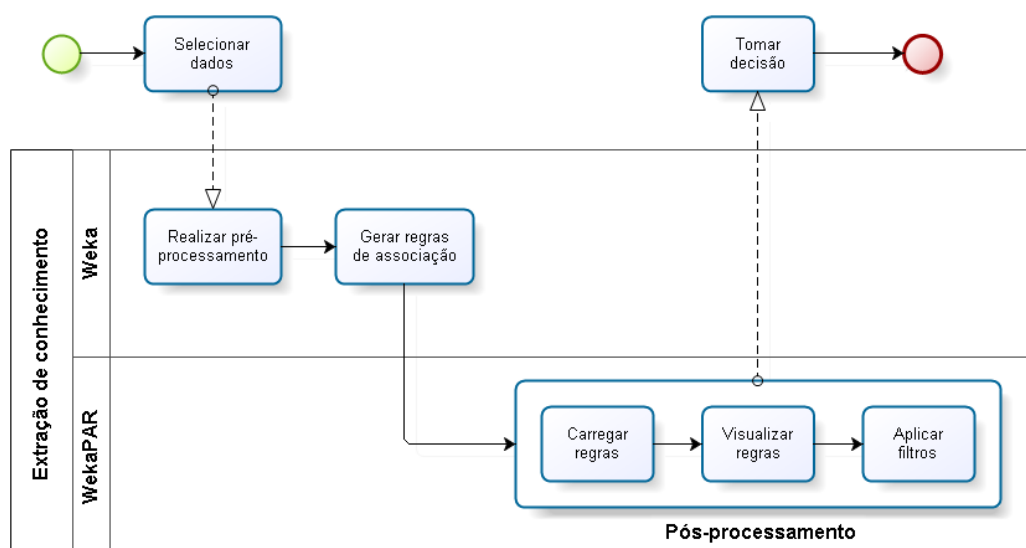
4 WEKAPAR: POST-PROCESSING OF ASSOCIATION RULES FOR WEKA

O problema investigado por esta pesquisa diz respeito às dificuldades que podem ser enfrentadas na etapa de pós-processamento de regras de associação. Neste trabalho, o problema estabelecido tem seu escopo definido para os usuários da ferramenta WEKA. Desta forma, duas abordagens poderiam ser consideradas para desenvolvimento da solução. A primeira consiste em desenvolver um software externo ao WEKA capaz de ler a saída do algoritmo de mineração de regras de associação executado na ferramenta, que pode ser salva em arquivo. A segunda abordagem dispõe sobre aprimorar as funcionalidades da WEKA, por meio de uma extensão que acrescentasse recursos de visualização e análise das regras geradas, de forma integrada à ferramenta, sem a obrigatoriedade de realizar um processo de exportação e importação dos dados. O segundo enfoque figurou como melhor opção, principalmente pelos recursos disponíveis na WEKA para o desenvolvimento de extensões e pelo benefício que a integração com a ferramenta confere aos usuários.

A solução proposta não modifica a forma como as regras de associação são geradas no WEKA. O propósito desta solução é prover ao usuário do WEKA um ambiente onde as regras mais interessantes possam ser selecionadas dado um conjunto de parâmetros estabelecidos pelo usuário. E, ao mesmo tempo que a solução se integra ao WEKA, garante a independência entre a extração de regras de associação pelos algoritmos do WEKA e o pós-processamento dos padrões obtidos.

A Figura 11 demonstra o processo de extração de padrões representados pelas regras de associação utilizando a ferramenta WEKA e a extensão proposta neste trabalho. O processo tem início com a seleção dos dados, uma atividade que neste modelo independe de ferramenta. Por meio do WEKA são executadas as tarefas de pré-processamento e execução do algoritmo de mineração de regras de associação. A partir deste ponto, o minerador de dados pode contar com os recursos da extensão desenvolvida para auxiliar o pós-processamento dos padrões extraídos. Os dados de entrada são as regras obtidas por meio dos recursos da ferramenta WEKA. Os padrões mais relevantes identificados após análise das regras subsidiam a tomada de decisão que encerra o processo.

Figura 11 – Processo de extração de conhecimento



Fonte: Elaboração própria.

Desta forma, a extensão desenvolvida implementa funcionalidades que visam prover ao minerador de dados que utiliza a ferramenta WEKA maior facilidade na etapa de pós-processamento de regras de associação. A implementação destas funcionalidades está baseada na especificação da ferramenta proposta que, por sua vez, é resultado da aplicação de técnicas da engenharia de requisitos. No Quadro 2 são apresentados os requisitos funcionais elicitados para este trabalho.

Quadro 2 – Requisitos funcionais

ID	Funcionalidade	Necessidades	Prioridade
RF1	Visualizar regras	Visualizar as regras extraídas numa tabela, exibindo antecedente e consequente da regra,	Alta

ID	Funcionalidade	Necessidades	Prioridade
		além das medidas de interesse, em colunas separadas.	
RF2	Filtrar regras	Aplicar um filtro no conjunto de regras tendo como entrada um termo de busca determinado pelo usuário.	Alta
RF3	Ordenar as regras	O software deve permitir que as regras possam ser ordenadas com base nos valores das medidas de interesse.	Alta
RF4	Aplicar filtro em um lado específico da regra	Possibilitar que um filtro seja aplicado a um determinado lado da regra (antecedente ou consequente), e que estes possam ser combinados em um mesmo filtro.	Alta
RF5	Explorar subconjunto de regras	Filtrar regras com base no subconjunto de atributos do antecedente de uma dada regra selecionada, mantendo o mesmo consequente.	Alta
RF6	Salvar as regras geradas	Salvar no disco o conjunto de regras exibidos na tabela.	Média
RF7	Exportar as regras	Exportar as regras visualizadas na tabela utilizando o formato CSV.	Baixa
RF8	Regra inversa	Exibir a regra inversa de uma dada regra selecionada, isto é, invertendo as posições do antecedente e consequente.	Alta
RF9	Filtrar regras com base nas métricas	A solução deve permitir estabelecer valores mínimos para as métricas disponíveis na WEKA e utilizar estes parâmetros como filtro para as regras.	Alta
RF10	Selecionar atributos	O usuário poderá filtrar as regras selecionando os atributos de seu interesse e seus respectivos rótulos para o antecedente e consequente da regra.	Alta
RF11	Combinar filtros	A solução deve permitir que os filtros aplicados ao antecedente e consequente das regras possam ser combinados com os valores mínimos das métricas de interesse determinados pelo usuário.	Alta

Fonte: Elaboração própria.

As demais especificações da solução proposta constam no documento de requisitos elaborado, cuja versão completa pode ser consultada no Apêndice A deste trabalho.

Nas seções seguintes são discutidos aspectos de implementação da solução proposta, a criação do pacote para WEKA, e ainda uma visão geral acerca das funcionalidades desenvolvidas.

4.1 IMPLEMENTAÇÃO

Uma vez definida a abordagem para desenvolver a solução na forma de uma extensão para a ferramenta WEKA, a linguagem de programação utilizada na implementação deveria ser necessariamente aquela utilizada no projeto do WEKA, isto é, a linguagem Java.

Inicialmente foi necessário estabelecer como as regras geradas poderiam estar disponíveis para visualização e análise. A maneira encontrada foi a implementação de um *plug-in* de visualização de regras de associação. O WEKA permite implementar uma interface específica do projeto que oferece acesso as regras de associação geradas. No trecho de código, exibido na Figura 12, o método *getVisualizeMenuItem* recebe como parâmetro um objeto do tipo *AssociationRules*, que armazena as regras extraídas.

Figura 12 – Código que implementa o *plug-in* de visualização

```
public JMenuItem getVisualizeMenuItem(final AssociationRules rules, final String name) {
    final JMenuItem menuItem = new JMenuItem("View in postprocess tab");
    menuItem.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            PostprocessAssociationsPanel.loadRules(rules);
        }
    });
    return menuItem;
}
```

Fonte: Elaboração própria.

A outra etapa da implementação preocupou-se em estabelecer de que forma as regras seriam visualizadas na ferramenta. Uma das opções que o WEKA fornece é estender o *Explorer* para adicionar novas guias. No entanto, é necessário satisfazer alguns requisitos.

Um dos requisitos é que a classe que cria a nova guia deve necessariamente ser derivada da classe Java *javax.swing.JPanel*, ou seja, a classe deve estender um *JPanel*. O outro requisito obrigatório é a implementação da interface *weka.gui.explorer.Explorer.ExplorerPanel*, por meio da qual é possível definir o título da guia e também ter acesso ao conjunto de instâncias carregado no WEKA. No caso da solução apresentada neste trabalho, não é necessário ter acesso as instâncias,

uma vez que os dados que devem ser considerados, que são as regras de associação, são fornecidas pela interface do *plug-in* de visualização.

Além destes requisitos, podem ser implementadas interfaces que, apesar de não ser obrigatórias, apresentam funcionalidades importantes. A interface *weka.gui.explorer.Explorer.LogHandler*, por exemplo, permite gerenciar mensagens de log na ferramenta WEKA. Por meio dos métodos desta interface, é possível exibir mensagens sobre o andamento das tarefas executadas, da mesma forma como ocorre com as demais funcionalidades nativas da ferramenta.

Os filtros implementados para seleção de atributos são aplicados nos dados da tabela onde são exibidas as regras de associação, por meio de expressões regulares¹⁰. Inicialmente, qualquer expressão regular suportada pelo Java pode ser aplicada como filtro¹¹. O padrão será comparado tanto com a coluna onde é exibido o antecedente da regra quanto com a coluna do consequente. No entanto, é possível determinar uma expressão a ser aplicada especificamente para o antecedente ou consequente da regra. Ademais, a solução permite ainda que termos de busca sejam combinados utilizando operadores lógicos, formando expressões cuja precedência pode ser definida com o uso de parênteses. Neste caso, as expressões dentro dos parênteses mais internos são processadas primeiro. Outra possibilidade é que os resultados incluam um termo específico, ou seja, composto exclusivamente por um determinado termo de busca.

O Quadro 3 elenca alguns tipos de filtro que podem ser aplicados aos padrões extraídos. Para aplicar um filtro a um lado específico da regra é necessário determinar se é o antecedente (X) ou o consequente (Y), seguido do termo de busca entre colchetes. Neste formato, os caracteres entre os colchetes são processados como expressões regulares. Quando o lado da regra não é indicado todo o termo de busca é tratado como uma expressão regular.

¹⁰ Uma expressão regular é composta por uma sequência de caracteres que representam um padrão utilizado para buscar correspondências em dados textuais (DEITEL, 2010).

¹¹ O tipo de processamento de expressões regulares utilizado na linguagem Java é baseado no padrão NFA (*Nondeterministic Finite Automaton*) tradicional (FRIEDL, 2006).

Quadro 3 – Exemplos de termos utilizados nos filtros

Termo de busca	Descrição
.*atributo1.*	Busca regras onde o termo “atributo1” aparece no antecedente ou no consequente.
X[atributo1]	Seleciona regras cujo antecedente contém o termo “atributo1”.
Y[atributo1]	Seleciona regras cujo consequente contém o termo “atributo1”.
Y[atributo1=rotulo1]	Seleciona regras cujo consequente contém o termo “atributo1=rotulo1”.
Y[^atributo1=rotulo1\$]	Seleciona regras cujo consequente é composto apenas pelo termo “atributo1=rotulo1”.
Y[atributo2] AND Y[atributo1]	Seleciona regras cujo consequente contém o termo “atributo2” e o termo “atributo1”, ocorrendo juntos.
Y[atributo2] OR Y[atributo1]	Seleciona regras cujo consequente contém o termo “atributo2” ou “atributo1”.
(Y[atributo3] OR (Y[atributo2] AND Y[atributo1]))	Seleciona regras cujo consequente contém o termo “atributo3” ou contém o termo “atributo2” e “atributo1”, ocorrendo juntos.

Fonte: Elaboração própria.

No que diz respeito aos valores mínimos das medidas de interesse, os filtros foram implementados de forma a selecionar regras cujos valores das métricas sejam iguais ou superiores ao limiar estabelecido. Desta forma, independente da métrica utilizada para extrair as regras de associação, com a extensão WekaPAR é possível filtrar as regras obtidas combinando qualquer uma das medidas de interesse disponíveis na ferramenta WEKA.

A visão geral das funcionalidades implementadas é apresentada na Seção 4.3, e no Capítulo 5 são demonstrados exemplos de aplicação dos filtros.

4.2 CRIAÇÃO DO PACOTE PARA WEKA

Desde a versão 3.7.2 do WEKA, novas extensões podem ser adicionadas à ferramenta, por meio de pacotes. É importante fazer a distinção em relação aos

pacotes do Java, que está relacionado a forma de organização de suas classes. Um pacote, no contexto do WEKA, consiste em um conjunto de funcionalidades adicionais disponibilizadas aos usuários, por meio de um arquivo que reúne todos os recursos necessários ao seu funcionamento, tais como executáveis Java, documentação, configurações, código-fonte, dentre outros. O pacote é um arquivo comprimido no formato ZIP, que possui uma estrutura definida. A Figura 13 mostra a estrutura do pacote da extensão desenvolvida nesta pesquisa. Alguns subdiretórios e arquivos foram omitidos para permitir uma melhor compreensão da estrutura principal.

Figura 13 – Estrutura do pacote

```

WekaPAR
|   WekaPAR.jar
|   build_package.xml
|   Description.props
|   Explorer.props
+---doc
+---lib
|   combinatoricslib-2.1.jar
\---src
    \---main
        \---java
            +---arpp
            |   \---table
            \---weka
                \---gui
                    +---explorer
                    |   PostprocessAssociationsPanel.java
                    \---visualize
                        \---plugins
                            PostprocessAssociationsPlugin.java

```

Fonte: Elaboração própria.

O diretório *src* contém o código-fonte do projeto. Algumas classes têm que, necessariamente, constar no pacote Java correspondente do projeto do WEKA. Estas classes ficam sob o pacote *weka*. No caso da extensão desenvolvida nesta pesquisa, a classe *PostprocessAssociationsPanel*, que adiciona uma nova guia ao *Explorer*, foi colocada no pacote *weka.gui.explorer*, e a classe *PostprocessAssociationsPlugin*, que permite acesso as regras geradas pela WEKA, no pacote *weka.gui.visualize.plugins*. Contudo, os arquivos do diretório *src* não são utilizados em tempo de execução pelo WEKA. A ferramenta irá buscar pelo arquivo JAR que contém as classes do projeto representadas pelos *bytecodes* Java.

No arquivo *Explorer.props* deve constar a referência para a classe que implementa a nova guia criada. Esta configuração é um requisito necessário para que a guia esteja disponível no WEKA. Além disso, ao iniciar o *Explorer* no WEKA, apenas

a guia de pré-processamento está disponível, até que um conjunto de dados seja carregado no ambiente para que então as demais guias estejam acessíveis. Neste projeto, optou-se por não adotar este padrão, sendo necessário adicionar a opção *standalone* na propriedade *Tabs*, conforme demonstrado na Figura 14. Isto permite que o trabalho salvo por meio da opção disponível na guia de pós-processamento possa ser acessado sem a necessidade de carregar novamente a base de dados.

Figura 14 – O arquivo *Explorer.props*

```
# Explorer.props file.
# Adds the PostprocessPanel to the Tabs key.
# The standalone option makes the tab available without
# requiring the preprocess panel to load a dataset first.

Tabs=weka.gui.explorer.PostprocessAssociationsPanel:standalone
```

Fonte: Elaboração própria.

Um outro arquivo de propriedades necessário é o *Description.props*, nele estão contidas informações sobre a extensão desenvolvida, que são utilizadas pelo gerenciador de pacotes do WEKA. Dentre as informações contidas no arquivo estão o nome do pacote, nome autor e mantenedor, dependências, versão e licença. Algumas das propriedades disponíveis são mostradas na Figura 15. Apesar de não ser obrigatória, a propriedade *depends* é relevante e consta na maioria dos pacotes, pois permite, além de indicar a dependência com outros pacotes, especificar a compatibilidade com o própria WEKA. Assim, no caso de tentar instalar o pacote em versões conflitantes, o usuário será avisado sobre a incompatibilidade.

Figura 15 – O arquivo *Description.props*

```
# Package name (required)
PackageName=WekaPAR

# Version (required)
Version=0.6.3

# Author (required)
Author=Daniel Silva

# Maintainer (required)
Maintainer=Daniel Silva <danielnsilva@gmail.com>

# License (required)
License=GPL 3.0

# Dependencies
Depends=weka (>=3.7.8)
```

Fonte: Elaboração própria.

No diretório *lib*, devem ser disponibilizadas as bibliotecas de terceiros necessárias ao funcionamento da extensão. A exemplo do arquivo JAR, o diretório

principal do pacote, todos os arquivos JAR do diretório *lib* também serão carregados pelo gerenciador de pacotes do WEKA.

A documentação do projeto está disponível no diretório *doc*. Trata-se de um diretório opcional na estrutura do pacote, mas configura uma boa prática, principalmente por se tratar de software de código aberto. Normalmente, neste diretório é encontrada ao menos a documentação do código-fonte. No caso deste projeto, o código-fonte foi documentado por meio da ferramenta Javadoc.

Parte do processo de criação do pacote pode ser automatizado com a utilização da ferramenta Ant¹². Os desenvolvedores do WEKA disponibilizam um *template* do arquivo de configuração que deve ser adaptado em cada projeto. Várias etapas do processo podem ser especificadas neste arquivo, como a compilação do código, a criação do arquivo JAR executável, o pacote ZIP, além da documentação gerada pelo Javadoc. A Figura 16 mostra um trecho do arquivo de configuração do Ant. Nesta parte do arquivo, são feitas as especificações para gerar a documentação do código-fonte.

Figura 16 – Trecho do arquivo de configuração do Ant

```
<!-- Make the javadocs -->
<target name="docs"
    depends="init_all"
    description="Make javadocs into ./doc">
    <mkdir dir="${doc}"/>
    <javadoc sourcepath="${src}"
        classpathref="project.class.path"
        destdir="${doc}"
        Author="yes"
        Public="yes"
        link="http://weka.sourceforge.net/doc.dev/"
        maxmemory="256m"/>
</target>
```

Fonte: Elaboração própria.

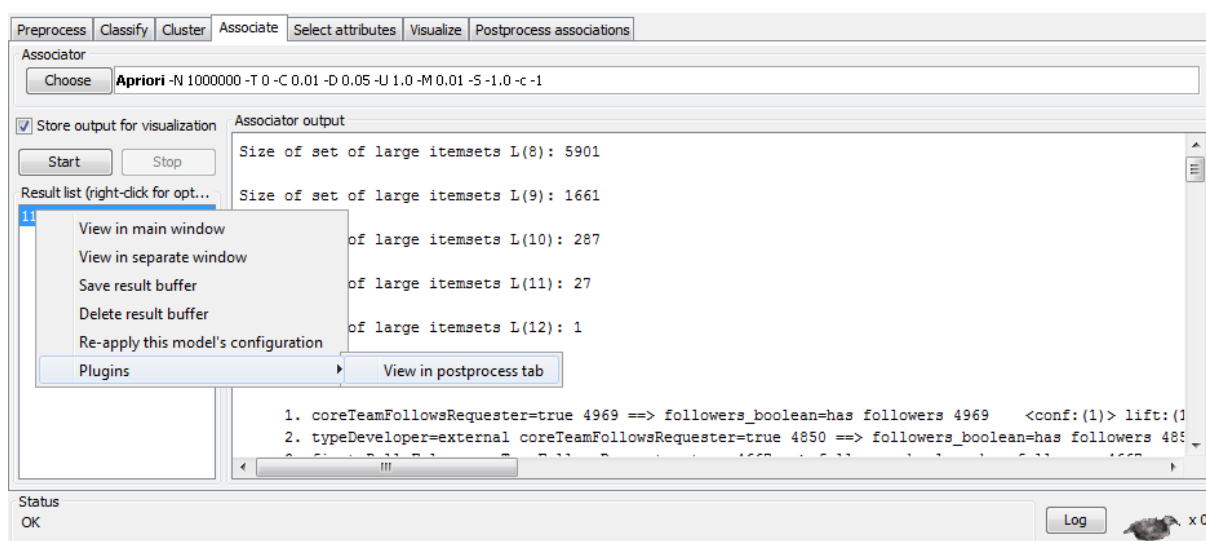
Há muito para ser explorado em relação aos pacotes do WEKA, além do que foi discutido nesta seção. A utilização de outros recursos depende do que a extensão pretende implementar como funcionalidade. No entanto, os aspectos abordados foram suficientes para criação do pacote deste projeto.

¹² Apache Ant é uma ferramenta que automatiza a execução algumas tarefas do processo de criação de software, tais como compilação de código-fonte, testes, gerenciamento de dependências, criação de pacotes, dentre outros (LOUGHRAN; HATCHER, 2007).

4.3 VISÃO GERAL

O processo de extrair regras de associação na ferramenta WEKA permanece o mesmo, tendo como único requisito para utilizar a guia de pós-processamento a seleção da opção “*store output for visualization*”, que irá habilitar uma nova opção no menu de contexto da lista de resultados, conforme demonstrado na Figura 17. Esta nova opção carrega os resultados do algoritmo na guia de pós-processamento.

Figura 17 – Opção de visualização das regras

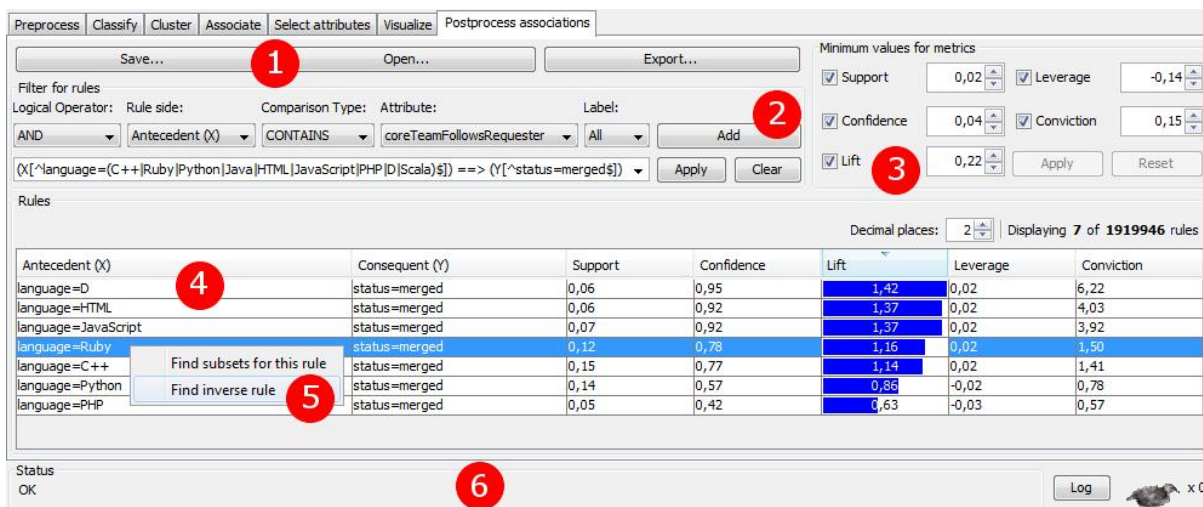


Fonte: Elaboração própria.

Todas as funcionalidades implementadas pela solução apresentada neste trabalho, são acessadas por meio da guia de pós-processamento de regras de associação. A Figura 18 mostra uma visão geral da tela, que é dividida em seis componentes principais: (1) uma barra de comandos com opções para salvar em arquivo e abrir o conjunto de regras, além exportá-los no formato CSV; (2) filtro para ser aplicado nas regras por meio da seleção de atributos e rótulos; (3) filtro para selecionar regras com base nos valores mínimos estabelecidos para as medidas de interesse; (4) uma tabela para exibição das regras; (5) um menu de contexto com opções para encontrar subconjuntos de regras e a regra inversa para uma determinada regra alvo selecionada¹³; (6) e a barra de status.

¹³ O subconjunto de regras e a regra inversa são filtros aplicados aos atributos da regra, construídos dinamicamente utilizando expressões regulares. A utilização de destes filtros é discutida na seção 5.1.2.

Figura 18 – Visão geral da guia de pós-processamento



Fonte: Elaboração própria.

Muitas vezes, o processo de mineração das regras de associação é demorado devido a relação direta entre as características da base de dados, como número de instâncias e atributos, e o custo computacional. A ferramenta permite que as regras extraídas sejam salvas em arquivo e possam ser carregadas novamente sem a necessidade de executar outra vez o algoritmo de mineração de regras de associação. No entanto, a contribuição desta funcionalidade limita-se à etapa de pós-processamento, uma vez que, havendo a necessidade de refazer atividades relacionadas a outras etapas, como a seleção de dados, o algoritmo deverá ser, necessariamente, executado para gerar novas regras.

Para ajudar na seleção das melhores regras, o usuário conta com um filtro interativo que pode ser aplicado de duas formas: a primeira com foco nas regras, por meio da seleção de atributos e rótulos; e a segunda através da definição de valores mínimos para as medidas de interesse. Os filtros constituem o principal recurso da solução para ajudar o minerador de dados a lidar com os problemas de pós-processamento de regra de associação, tais como o elevado número de regras, regras redundantes, dentre outros.

Diferente da saída padrão do WEKA, que mostra as regras de forma não estruturada, na guia de pós-processamento os padrões extraídos são dispostos numa tabela onde cada regra é exibida em uma linha, e seu antecedente e consequente, além de suas medidas de interesse, são separados em colunas. Esta abordagem

facilita sua visualização dos padrões, tornando a interação do minerador de dados com a ferramenta mais amigável.

Há ainda a barra de status, que permite manter o usuário informado sobre o que está sendo executado pela ferramenta. Este é um recurso nativo do WEKA, sendo útil, por exemplo, para indicar o status da execução do algoritmo de mineração de regras de associação, bem como informar o fim do processo. É também utilizado na guia de pós-processamento, servindo como uma forma de integração com a ferramenta WEKA.

4.3.1 Integração com a WEKA

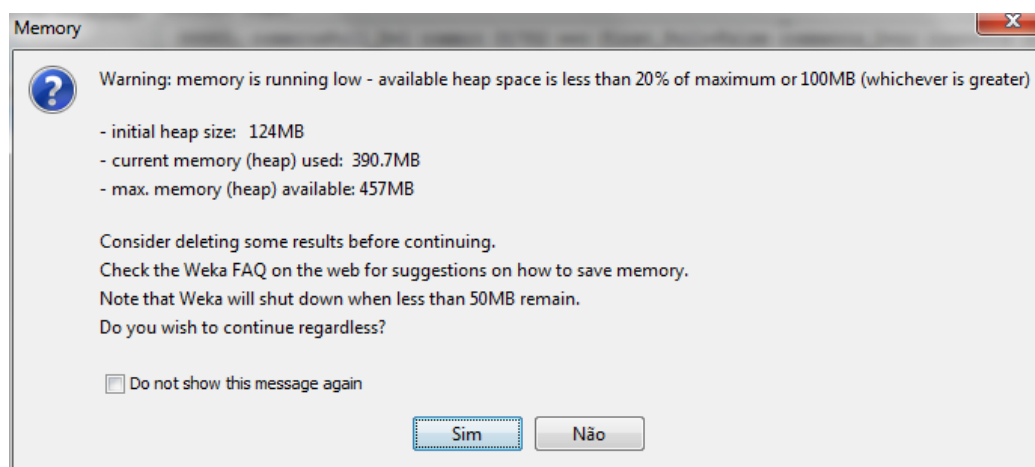
Além de fornecer meios de facilitar a seleção das melhores regras, a solução proposta preocupa-se em manter a integração com a ferramenta WEKA. A forma de implementação por meio de uma extensão da ferramenta já configura como um fator de integração. No entanto, há outros aspectos que são levados em consideração para garantir ao usuário do WEKA uma solução que se aproxima da forma de utilização dos recursos nativos da ferramenta.

Um destes fatores é o layout da tela de visualização das regras. Por tratar-se de um recurso novo que engloba aspectos de visualização, não há no WEKA um modelo que possa ser seguido, como no caso do desenvolvimento de um novo algoritmo de classificação cujos parâmetros de entrada e saída são bem definidos pela ferramenta. Porém, são utilizados na solução componentes comuns de interfaces gráficas como botões, campos de texto, caixas de marcação e tabelas, não sendo aplicadas a estes componentes nenhum tipo de modificação visual. Desta forma, os componentes exibidos na guia de pós-processamento acompanham o padrão visual da ferramenta.

Outro fator considerado é o gerenciamento do consumo de memória. A extração de um grande número de regras de associação exige também uma grande quantidade de memória disponível. No WEKA, o consumo de memória é verificado antes da execução de cada tarefa no intuito de evitar que o processamento resulte em

falha devido à falta de memória disponível. Caso a quantidade de memória disponível seja igual ou menor a 20% do total, é exibida uma mensagem informando ao usuário que o software poderá ser encerrado em caso de não haver memória disponível suficiente para o seu funcionamento. Este recurso de verificação da memória disponível também é utilizado antes da execução de cada funcionalidade da guia de pós-processamento de regras de associação. Um exemplo de mensagem de alerta sobre a baixa quantidade de memória disponível é exibido na Figura 19.

Figura 19 – Mensagem de alerta sobre memória disponível




Fonte: Elaboração própria.

A exibição de informações sobre o andamento da execução das funcionalidades da guia de pós-processamento é outro fator considerado no desenvolvimento com o objetivo de enriquecer a integração com o WEKA. Por meio deste recurso o usuário é informado sobre o carregamento das regras na tabela, aplicação dos filtros, gravação e leitura de arquivos, ou mesmo sobre a ocorrência de erros durante a execução. No entanto, o espaço da barra de status é limitado, sendo útil apenas para exibição de mensagens curtas. Para registrar informações mais completas, como no caso de longas mensagens de erro, é utilizado o recurso de *log* do WEKA, cuja visualização está acessível por meio do botão ao lado da mensagem de status. Além destas informações, uma imagem animada no canto inferior direito da tela indica que alguma tarefa está em execução.

Figura 20 – Barra de status e logs

requester_experience=very experienced li...	followers_boolean=h...	0,13	1,00	1,06	0,01	23,33
requester_experience=very experienced li...	followers_boolean=h...	0,15	1,00	1,06	0,01	23,52
requester_experience=very experienced li...	followers_boolean=h...	0,12	1,00	1,06	0,01	23,19
requester_experience=very experienced li...	followers_boolean=h...	0,18	1,00	1,06	0,01	23,41
requester_experience=very experienced li...	followers_boolean=h...	0,18	1,00	1,06	0,01	23,17
requester_experience=very experienced f...	followers_boolean=h...	0,15	1,00	1,06	0,01	22,85
requester_experience=very experienced f...	followers_boolean=h...	0,12	1,00	1,06	0,01	22,49
requester_experience=very experienced f...	followers_boolean=h...	0,15	1,00	1,06	0,01	22,78

Status
Loading rules...

Log
 x 1

Fonte: Elaboração própria.

A implementação destes recursos como forma de integração com a WEKA, complementa as funcionalidades disponíveis na guia de pós-processamento. Mesmo não sendo um requisito obrigatório, esta integração melhora a experiência do usuário com a ferramenta.

4.4 CONCLUSÃO

Neste capítulo, foi apresentada a solução proposta para o problema endereçado por esta pesquisa. Foram discutidos a implementação, a criação do pacote para WEKA, além da apresentação de uma visão geral da extensão desenvolvida, incluindo aspectos de integração com a ferramenta WEKA.

No próximo Capítulo, é apresentado os resultados da validação das funcionalidades desenvolvidas e da avaliação realizada com usuários.

5 AVALIAÇÃO DA SOLUÇÃO PROPOSTA

A avaliação da solução proposta neste trabalho consiste em demonstrar as funcionalidades implementadas e como estas permitem mais facilidade ao minerador de dados na etapa de pós-processamento de regras de associação. Também faz parte da avaliação analisar o impacto da solução proposta sob a perspectiva de usuários da ferramenta WEKA.

Desta forma, foram realizados experimentos cuja tarefa básica era a exploração do conjunto de regras de associação extraídas utilizando a ferramenta WEKA, e analisadas com extensão desenvolvida por meio da aplicação de filtros, objetivando identificar padrões relevantes em bases de dados de repositórios de software.

Para avaliação com usuários foi aplicado um questionário com o objetivo de validar a solução proposta em relação a satisfação dos avaliadores após utilização do WEKA para analisar os padrões extraídos com os recursos nativos e, posteriormente, utilizando a extensão apresentada neste trabalho.

Desta forma, o Capítulo foi dividido em duas seções principais, além da conclusão. Na primeira seção é apresentado o resultado da validação das funcionalidades desenvolvidas, e na segunda seção o resultado da avaliação realizada com usuários.

5.1 VALIDAÇÃO DAS FUNCIONALIDADES IMPLEMENTADAS

A forma de visualização das regras de associação proposta neste trabalho, em contraste com o padrão da ferramenta WEKA, representa uma vantagem ao minerador de dados na etapa de análise e interpretação das regras extraídas. No entanto, são os filtros disponíveis que oferecem ao especialista de domínio mais recursos para lidar com os problemas relacionados ao pós-processamento de regras de associação.

5.1.1 Base de dados

Nos exemplos de aplicação dos filtros demonstrados nas seções seguintes, foram utilizadas duas bases de dados com informações de *pull requests*¹⁴ extraídos de repositórios de software do GitHub. A primeira base de dados reúne informações de 19 projetos. Esta base conta 46.910 instâncias. A outra possui apenas dados do projeto TrinityCore. É uma base menor que a anterior, com 2.173 instâncias.

As duas bases de dados possuem os mesmos atributos. No entanto, nem todos os atributos foram considerados. Alguns atributos são resultado de um trabalho de pré-processamento dos dados originais do GitHub, com a utilização de algoritmos de discretização de dados. O Quadro 4 mostra os 14 atributos selecionados dentre os 61 disponíveis.

Quadro 4 – Atributos utilizados na extração das regras

Atributo	Descrição
language	Linguagem de programação utilizada no projeto
followers_boolean	Indica se desenvolvedor (requester) possui seguidores no GitHub

¹⁴ *Pull request* consiste em um recurso do GitHub que permite que sejam submetidas solicitações de alterações de código para serem avaliadas pelos desenvolvedores principais de um projeto de software (BIRD; MENZIES; ZIMMERMANN, 2015).

Atributo	Descrição
requester_experience	Grau de experiência do desenvolvedor (requester)
first_Pull	Indica se é o primeiro pull request do desenvolvedor
life_time	Tempo de vida do pull request
closedBy	Desenvolvedor que avaliou e encerrou o pull request
status	Indica se o pull request foi aceito ou rejeitado
comments_D	Indica se foram feitos comentários ao pull request durante seu tempo de vida
commitsPull_D	Quantidade de commits presentes no pull request
total_lines_D	Quantidade de linhas de código modificadas nos arquivos que fazem parte do pull request
changedFiles_D	Quantidade de arquivos modificados no pull request
typeDeveloper	Indica se o desenvolvedor faz parte do time principal de desenvolvimento ou é externo em relação ao projeto
coreTeamFollowsRequester	Indica se o avaliador segue o desenvolvedor (requester) no GitHub
requester_experience_project	Quantidade de contribuições ao projeto já realizadas pelo desenvolvedor (requester)

Fonte: Elaboração própria.

5.1.2 Aplicação de filtros

Um dos principais problemas da etapa de pós-processamento de regras de associação é o elevado número de padrões gerados. Considerando o modelo baseado nas medidas de suporte e confiança para gerar regras de associação, em muitos casos os padrões que interessam ao especialista de domínio possuem valores muito baixos para estas medidas, o que pode levar a um grande volume de regras para serem analisadas.

A Figura 21 mostra o resultado da aplicação de um filtro para selecionar regras cujo antecedente é formado pelos atributos *first_Pull* e *life_time*, considerando qualquer um de seus rótulos, no intuito de analisar a influência destes fatores na

aceitação de *pull requests*. O filtro foi aplicado em um conjunto de regras mineradas com valores de suporte e confiança de 7,5% e 50%, respectivamente, resultando em um total de 70.718 regras geradas.

Figura 21 – Exemplo de resultado da aplicação do filtro

Rules						
			Decimal places: <input type="text" value="2"/>		Displaying 4 of 70718 rules	
Antecedent (X)	Consequent (Y)	Support	Confidence	Lift	Leverage	Conviction
first_Pull=False life_time=Very Fast	status=merged	0,25	0,82	1,23	0,05	1,86
first_Pull=False life_time=Fast	status=merged	0,12	0,78	1,17	0,02	1,51
first_Pull=False life_time=Medium	status=merged	0,10	0,75	1,12	0,01	1,30
first_Pull=False life_time=Slow	status=merged	0,08	0,59	0,89	-0,01	0,81

Fonte: Elaboração própria.

As três primeiras regras obtidas por meio da aplicação do filtro indicam, com base no valor do *lift* que quando não se trata do primeiro *pull request* do desenvolvedor e que seu tempo de vida é médio, rápido ou muito rápido, as chances de serem aceitos aumentam na ordem de 12%, 17% e 23%, respectivamente. No entanto, o filtro não recuperou nenhuma regra para o caso de ser o primeiro *pull request* do desenvolvedor (*first_Pull = True*), o que indica que não há uma regra com esta informação que satisfaça as condições do filtro dentro do conjunto de regras extraídas.

Desta forma, o padrão pode ser evidenciado através de uma nova extração de regras, reduzindo o valor de suporte para 2% e confiança para 25%. Para este novo conjunto de padrões o mesmo filtro recupera mais duas regras onde o atributo *first_Pull = True* está presente. Estas regras, mostradas na Figura 22, indicam que o primeiro *pull request* de um desenvolvedor tem menos chances de ser aceito, seja o tempo de avaliação muito rápido (26% menos chances) ou demorado (61% menos chances). Assim, tem-se uma informação que pode ser útil para o especialista de domínio, mas que para ser obtida foram geradas 946.498 regras, um número muito elevado se comparado ao conjunto anterior.

Observa-se que estes padrões presentes no segundo conjunto de regras gerado têm valores de suporte mais baixos, indicando que ocorrem com menos frequência na base de dados, o que explica a necessidade de se obter um maior número de regras para que possam ser encontrados. Esta é uma característica presente em bases de dados de muitos domínios, onde a descoberta de padrões

interessantes pode exigir valores muito baixos de suporte, o que também evidencia a utilidade da aplicação de filtros nos conjuntos de regras gerados.

Figura 22 – Aplicação do filtro após redução de métricas

Rules						
			Decimal places: 2		Displaying 6 of 946498 rules	
Antecedent (X)	Consequent (Y)	Support	Confidence	Lift	Leverage	Conviction
first_Pull=False life_time=Very Fast	status=merged	0,25	0,82	1,23	0,05	1,86
first_Pull=False life_time=Fast	status=merged	0,12	0,78	1,17	0,02	1,51
first_Pull=False life_time=Medium	status=merged	0,10	0,75	1,12	0,01	1,30
first_Pull=False life_time=Slow	status=merged	0,08	0,59	0,89	-0,01	0,81
first_Pull=True life_time=Very Fast	status=merged	0,04	0,49	0,74	-0,02	0,65
first_Pull=True life_time=Slow	status=merged	0,02	0,26	0,39	-0,04	0,45

Fonte: Elaboração própria.

Para ajudar o especialista de domínio no processo de análise das regras de associação geradas, a solução proposta neste trabalho conta com 4 formas diferentes de filtrar os resultados obtidos: selecionando os atributos tanto do antecedente quanto do consequente da regra, estabelecendo valores mínimos para as medidas de interesse, buscando regras baseadas nos subconjuntos de atributos do antecedente de uma dada regra, e comparando regras inversas. O filtro aplicado as regras pode ainda ser combinado com valores mínimos das métricas, de forma a obter melhores resultados.

5.1.2.1 Seleção de atributos

O problema do elevado número de regras extraídas não está relacionado tão somente à quantidade de instâncias na base de dados e aos valores de suporte e confiança, mas também à quantidade de atributos, bem como ao número de rótulos distintos que cada atributo possui.

Para demonstrar como a seleção de atributos pode contribuir para diminuição do número de regras na etapa de pós-processamento, foram extraídas um total de 1.919.946 regras tendo com parâmetro um suporte de 2% e a confiança de 4%. O resultado da aplicação dos filtros para selecionar regras com base nos atributos presentes no consequente é mostrado na Tabela 6.

Os três primeiros filtros mostrados na tabela selecionam regras cujos itens do consequente incluem um atributo específico ou a combinação de dois ou mais atributos, independente de rótulo. A partir do filtro 4 são selecionadas regras onde o consequente é composto exclusivamente pelos atributos escolhidos, ou um conjunto deles, que podem ainda possuir um rótulo específico. Observa-se que os mesmos filtros também podem ser aplicados no antecedente da regra. Conforme demonstrado, cada um destes filtros contribui para diminuição do número de regras.

Tabela 6 – Seleção de regras com base em atributos

#	Comparação	Consequente	Número de regras	Redução (%)
1	Contém	<i>status</i>	563.809	70,63
2	Contém	<i>status</i> \wedge <i>life_time</i>	113.146	94,11
3	Contém	<i>status</i> \wedge <i>life_time</i> \wedge <i>comments_D</i>	31.391	98,37
4	Igual	<i>status</i>	14.782	99,23
5	Igual	<i>status</i> = <i>merged</i>	10.914	99,43
6	Igual	<i>status</i> \wedge <i>life_time</i>	5.329	99,72
7	Igual	<i>status</i> = <i>merged</i> \wedge <i>life_time</i> = <i>Very Fast</i>	2.554	99,87
8	Igual	<i>status</i> \wedge <i>life_time</i> \wedge <i>comments_D</i>	2.193	99,89
9	Igual	<i>status</i> = <i>merged</i> \wedge <i>life_time</i> = <i>Very Fast</i> \wedge <i>comments_D</i> = <i>has comments</i>	36	99,99

Fonte: Elaboração própria.

Uma das opções disponíveis na implementação do *Apriori* para o WEKA é a determinação de um atributo único para o consequente, de forma que somente sejam exibidas na saída do algoritmo regras que satisfazem essa condição (WITTEN; FRANK; HALL, 2011). Esta técnica, denominada CAR (*Class Association Rules*), também pode ser reproduzida utilizando um filtro. O item 4 da Tabela 6 é um exemplo de aplicação filtro para selecionar regras cujo consequente possui um único atributo. Adicionalmente, os recursos de filtragem demonstrados neste trabalho complementam a técnica na medida em que permitem selecionar um rótulo específico (item 5) ou ainda utilizar a composição de dois ou mais atributos¹⁵ (item 6).

¹⁵ A utilização da técnica CAR para determinar dois ou mais atributos no consequente é possível no WEKA desde que na etapa de pré-processamento seja criado um novo atributo a partir da composição de outros.

Cabe aqui retomar aquilo que foi abordado no Capítulo 2, sobre a possibilidade de realizar esta seleção de atributos ainda na etapa de pré-processamento. Observa-se que a mineração de regras de associação é uma atividade exploratória e, desta forma, pode ser útil ao minerador de dados utilizar recursos de visualização para analisar os padrões gerados, mesmo que decida posteriormente realizar uma nova extração de regras.

Assim, a seleção de atributos na etapa de pós-processamento permite que o especialista de domínio encontre as melhores regras com base nos atributos de seu interesse, considerando o conjunto de padrões já extraídos. Neste sentido, a utilização de filtros para seleção de atributos facilita a execução da atividade de análise e interpretação dos padrões descobertos.

5.1.2.2 Medidas de interesse

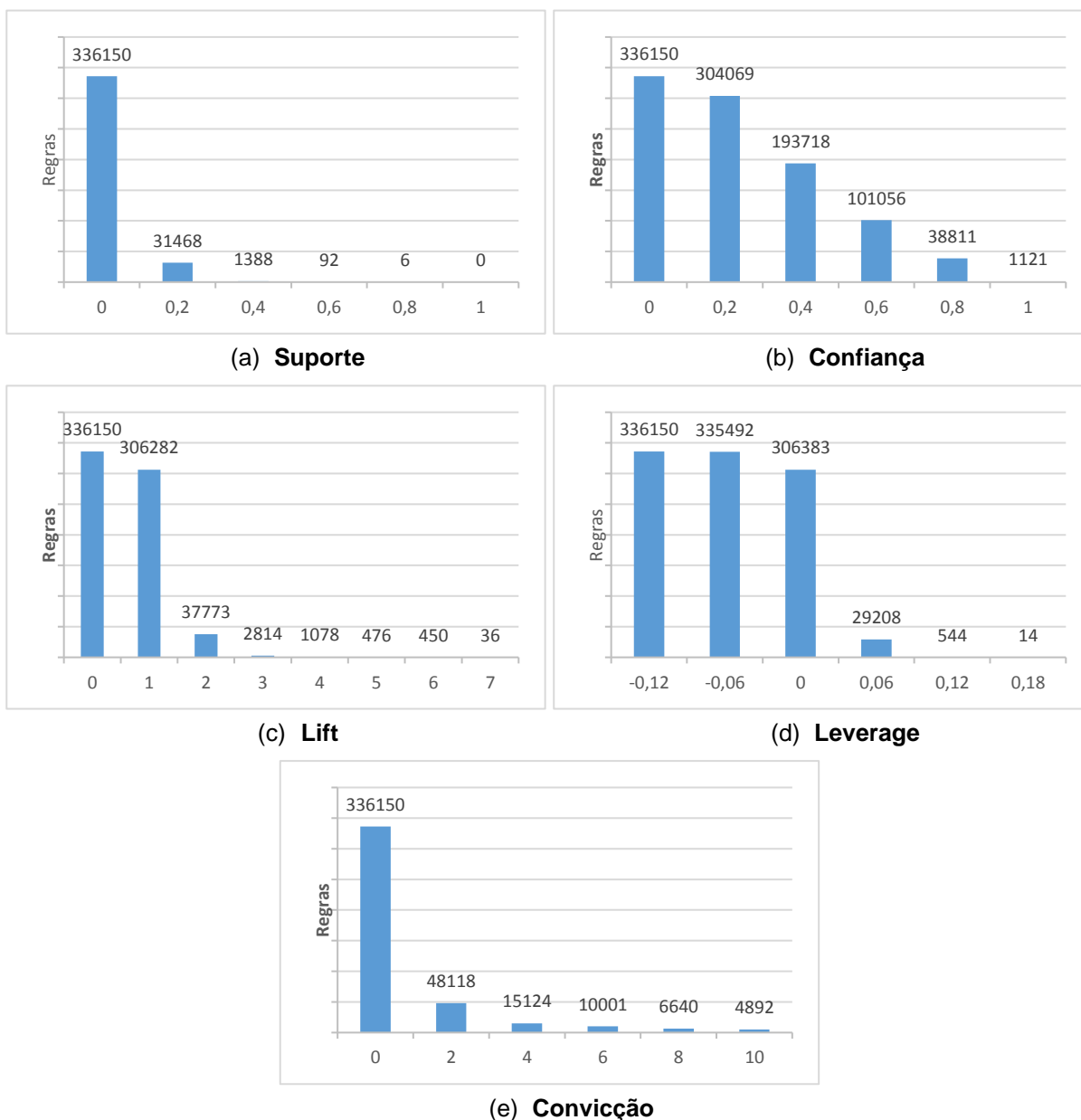
As medidas de interesse são as principais referências utilizadas pelo minerador de dados para identificar padrões relevantes. Por meio da solução proposta neste trabalho, é possível filtrar os padrões estabelecendo valores mínimos para cada uma das métricas disponíveis no WEKA, e não somente aquela utilizada como parâmetro na extração das regras de associação. Em uma análise mais simples, quanto maior o valor destas medidas, mais relevante é a regra. Neste sentido, a Figura 23 mostra a relação entre os valores mínimos das medidas de interesse (representados no eixo horizontal) e o número de regras de associação resultante da aplicação do filtro (representado no eixo vertical).

Observa-se que para as medidas suporte (a) e confiança (b) os valores considerados na escala são todos os valores possíveis para estas medidas, que estão entre 0 e 1. No caso do *lift* (c) e *leverage* (d) os valores considerados são aqueles observados no conjunto de regras geradas.

Para a medida convicção não foram considerados todos os valores medidos, uma vez que esta métrica pode auferir valores muito altos que normalmente

representam regras óbvias, como por exemplo se o desenvolvedor é muito experiente implica que ele fez muitas contribuições ao projeto, ou se o desenvolvedor não possui nenhuma experiência também nunca contribuiu com o projeto. Valores altos para esta medida podem indicar também regras que possuem muitos atributos, evidenciando padrões difíceis de serem explicados.

Figura 23 – Número de regras e valores de métricas



Fonte: Elaboração própria.

É importante notar a relação entre os valores mínimos das medidas e o número de regras. O que se observa é que todas elas podem contribuir individualmente para diminuir o número de padrões. No entanto, pode ser interessante ao minerador de

dados não somente estabelecer um valor mínimo para uma determinada métrica, mas também combinar duas ou mais medidas, reduzindo o número de regras no intuito de identificar padrões mais relevantes.

Na Tabela 7 são mostrados exemplos de diferentes combinações de valores mínimos de medidas de interesse e o número de regras obtido em cada uma. Ressalte-se que o minerador de dados não precisa necessariamente observar todas as medidas. A combinação pode envolver apenas as métricas de seu interesse.

Tabela 7 – Filtros com valores combinados de métricas

#	Valores mínimos das métricas					Número de regras
	Suporte	Confiança	Lift	Leverage	Convicção	
1	0,10	0,20	1,00	0,05	1,50	11.873
2	0,15	0,25	1,50	0,08	1,75	3872
3	0,20	0,30	1,75	0,10	2,00	194

Fonte: Elaboração própria.

Além do problema dos valores elevados da convicção, também há possibilidade de que cada uma das outras medidas não seja eficaz na identificação de padrões relevantes. A determinação do valor mínimo para uma dada métrica deve ser feita com cautela. Como já demonstrado neste capítulo, para determinados domínios, padrões relevantes possuem suporte muito baixo, ao passo que elevar o valor mínimo desta medida pode ocultar regras que interessam ao minerador. O mesmo pode ocorrer também com a métrica confiança. E quanto ao *lift*, a exemplo do que acontece com a medida convicção, os valores elevados podem significar regras que não representam informações úteis.

As medidas de interesse podem apresentar uma interpretação diferente para cada regra. Neste sentido, é útil ao minerador de dados ordenar os padrões extraídos com base nas medidas de interesse.

5.1.2.3 Combinando filtros de atributos e métricas

A seleção de atributos presentes no antecedente ou consequente da regra e a determinação de valores mínimos para as métricas de interesse são filtros que podem ser utilizados de forma individual ou em conjunto. As composições dos dois tipos de filtro podem levar a um número ainda menor de regras.

Na Tabela 8 é mostrada a variação dos valores das métricas suporte e *lift* ao longo do tempo, para padrões identificados a partir da aplicação de filtros. Para este exemplo foram realizadas três extrações na base de dados do projeto TrinityCore para criar novas bases distintas, tendo como referência o ano do *pull request*. Utilizando os dados do ano de 2012 foram extraídas 278.181 regras, com suporte e confiança igual ou superior a 1% e 70% respectivamente. Por meio da aplicação dos filtros, foram selecionadas regras que indicam a relação entre a experiência do desenvolvedor (antecedente) e a aceitação do *pull request* (consequente), cujo *lift* é maior que 1. Com base nos padrões identificados foram obtidas regras para os anos de 2013 e 2014.

Tabela 8 – Variação das métricas ao longo do tempo

#	Antecedente da regra (Consequente <i>status = merged</i>)	2012		2013		2014	
		Sup	Lift	Sup	Lift	Sup	Lift
1	<i>requester_experience_project = many contributions</i>	0,42	1,11	0,31	1,25	0,27	1,33
2	<i>requester_experience = very experienced</i>	0,25	1,11	0,16	1,32	0,16	1,47

Fonte: Elaboração própria.

As regras indicam, com base no *lift*, que a experiência do desenvolvedor no projeto aumenta as chances de aceitação do *pull request* na ordem de 11%, 25% e 33% respectivamente de 2012 a 2014, e o grau de experiência geral do desenvolvedor também aumenta as chances de aceitação em percentuais ainda maiores. É um padrão que não apenas é confirmado ao longo do tempo, mas também se torna cada vez mais forte.

Este tipo de análise é um dos aspectos que demonstram a importância da etapa de pós-processamento de regras de associação. Neste exemplo a combinação dos filtros para selecionar atributos e definir valores mínimos de métricas foi importante principalmente para análise do primeiro conjunto de regras extraídos, pois foi possível

estabelecer tanto os atributos e rótulos de ambos os lados da regra como um valor mínimo para o *lift*. Outro aspecto importante nesta análise é a integração com o WEKA, que permitiu obter as regras dos outros períodos, uma vez identificado um padrão de interesse usando a própria ferramenta.

5.1.2.4 Subconjunto de regras

Conforme discutido no Capítulo 2, Han, Kamber e Pei (2011) destacam que umas das características da mineração de regras de associação diz respeito ao fato de que se um dado conjunto de itens é frequente, todos os seus subconjuntos também são. Esta é uma característica que pode levar a extração de muitas regras redundantes. Uma das formas de identificar padrões redundantes no conjunto de regras extraídas é analisando os subconjuntos de regras gerados a partir da combinação dos atributos presentes no antecedente da regra. Isto é, selecionar no conjunto de regras aquelas cujo antecedente é formado por qualquer uma das combinações possíveis dos itens que compõe o antecedente de uma dada regra alvo.

Com a aplicação deste filtro, o especialista de domínio pode avaliar se a subtração de um ou mais itens no antecedente da regra altera o valor das medidas de interesse. Esta análise pode indicar tanto que a presença de um item contribui para aumentar a força de uma regra ou que a ocorrência de um item diminui a importância da regra, ou ainda pode indicar que este item interfere pouco ou nada nas medidas analisadas, sendo, portanto, uma regra redundante.

Um exemplo de resultado da aplicação do filtro para encontrar os subconjuntos de uma regra é mostrado na Tabela 9, que contém padrões obtidos de dados do projeto TrinityCore. A medida *lift* da regra 2 indica que quando o arquivo *objectmgr.h* é modificado, e os arquivos *dbcstructure.h* e *item.cpp* não, as chances do *pull request* ser rejeitado aumentam na ordem de 60%. Todas as demais regras foram selecionadas a partir da aplicação do filtro, tendo a regra 2 como alvo. Nota-se que a regra 1 possui um *lift* maior que o da regra 2. Tal constatação demonstra que a inclusão dos atributos que representam os arquivos não modificados diminui a

importância da regra. De fato, para o domínio abordado nesta avaliação, há muitos casos onde há apenas 1 arquivo modificado no *pull request*, e, por conseguinte, todos os demais arquivos do projeto não foram modificados. Neste sentido, é o atributo que representa o arquivo modificado que evidencia o padrão, ao passo que quando não está presente, como nas regras 5, 6 e 7, diminui as chances de ocorrer o consequente.

Tabela 9 – Subconjuntos de regras que indicam a rejeição de um *pull request*

#	Antecedente da regra (Consequente <i>status = closed</i>)	Sup	Conf	Lift
1	<i>objectmgr.h</i> = 1	0,02	0,66	1,70
2	<i>dbcstructure.h</i> = 0 \wedge <i>objectmgr.h</i> = 1 \wedge <i>item.cpp</i> = 0	0,01	0,62	1,60
3	<i>objectmgr.h</i> = 1 \wedge <i>item.cpp</i> = 0	0,01	0,61	1,57
4	<i>dbcstructure.h</i> = 0 \wedge <i>objectmgr.h</i> = 1	0,01	0,59	1,51
5	<i>item.cpp</i> = 0	0,38	0,38	0,98
6	<i>dbcstructure.h</i> = 0	0,37	0,38	0,98
7	<i>dbcstructure.h</i> = 0 \wedge <i>item.cpp</i> = 0	0,37	0,38	0,98

Fonte: Elaboração própria.

Em outros casos, a análise pode não ser tão simples quanto o exemplo da Tabela 9. A Tabela 10 mostra outro resultado da aplicação do filtro para encontrar os subconjuntos de uma regra. Neste caso, os subconjuntos encontrados têm como origem a regra 1, cuja medida *lift* indica que quando não se trata do primeiro *pull request* do desenvolvedor, e que o tempo de avaliação é muito rápido e há apenas 1 *commit*, as chances do *pull request* ser aceito são 23% maiores. No entanto, a regra 2 possui o mesmo *lift* da regra 1, porém com um atributo suprimido. Isto denota que as características *first_Pull = False* \wedge *life_time = Very Fast* são suficientes para explicar o padrão. Desta forma, a presença do atributo *commitsPull_D = 1* torna a regra redundante, pois não acrescenta informação útil ao padrão. A menor importância deste atributo pode ainda ser verificada na regra 7, onde o *lift* é de apenas 1,02.

Tabela 10 – Subconjuntos de regras que indicam a aceitação de um *pull request*

#	Antecedente da regra (Consequente <i>status = merged</i>)	Sup	Conf	Lift
1	$first_Pull = False \wedge life_time = Very\ Fast \wedge commitsPull_D = 1$	0,20	0,82	1,23
2	$first_Pull = False \wedge life_time = Very\ Fast$	0,25	0,82	1,23
3	$life_time = Very\ Fast \wedge commitsPull_D = 1$	0,24	0,77	1,14
4	$first_Pull = False \wedge commitsPull_D = 1$	0,38	0,76	1,14
5	$first_Pull = False$	0,55	0,76	1,13
6	$life_time = Very\ Fast$	0,30	0,75	1,13
7	$commitsPull_D = 1$	0,46	0,69	1,02

Fonte: Elaboração própria.

De acordo com os exemplos apresentados nesta seção, fica evidente a utilidade do filtro para visualizar os subconjuntos de uma dada regra. Com a aplicação deste filtro é possível identificar regras redundantes, um dos problemas relacionados as regras de associação, e, assim, ressaltar padrões que realmente representam informações úteis.

5.1.2.5 Regra inversa

Ao analisar uma dada regra de associação, pode ser útil ao minerador de dados verificar se a inversão da posição do antecedente e consequente na regra evidenciará padrões com diferenças nas métricas. Este tipo de análise está fundamentada na existência de medidas de interesse assimétricas, ou seja, métricas cujos valores são diferentes para as regras inversas. No WEKA, estão disponíveis duas medidas deste tipo: a confiança e a convicção. A aplicação do filtro para buscar regras inversas permite ao especialista de domínio comparar os valores das medidas de interesse assimétricas para o par de regras selecionadas. Desta forma, a análise sempre estará baseada em duas regras: a regra original e a regra cujo antecedente e consequente tem suas posições invertidas em relação a regra original.

Um dos resultados possíveis desta análise é a confirmação de que o padrão selecionado tem ou não a mesma força de sua respectiva regra invertida. Isto pode ser verificado não somente para valores exatamente iguais nas duas regras, mas

também quando os valores das medidas de interesse confiança e convicção se aproximam dos valores da regra original, indicando forte correlação entre os dois lados da regra qualquer que seja a posição.

Na Tabela 11, o padrão original, representado pela regra 1, indica com 39% de confiança que quando o tempo de avaliação é demorado, trata-se do primeiro *pull request* do desenvolvedor. Sua regra inversa, representada pela regra 2, possui confiança de 35%, um percentual muito próximo ao da regra original, o que também pode ser verificado na medida convicção, demonstrando a forte correlação entre antecedente e consequente. O valor 1,53 para medida *lift* das duas regras já evidencia a forte correlação, mas a análise das medidas assimétricas confirma a relevância do padrão independentemente da direção da implicação na regra.

Tabela 11 – Regra inversa ratificando a correlação entre os lados da regra

#	Regra	Confiança	Convicção	Lift
1	<i>life_time = Slow</i> \rightarrow <i>first_Pull = True</i>	39%	1,22	1,53
2	<i>first_Pull = True</i> \rightarrow <i>life_time = Slow</i>	35%	1,18	1,53

Fonte: Elaboração própria.

Por outro lado, este tipo de análise também pode levar a uma conclusão oposta. Por meio da avaliação das medidas assimétricas, o minerador pode verificar que estas possuem valores muito diferentes quando comparados o padrão original e sua regra invertida. Desta forma, uma das regras pode ser considerada irrelevante, mesmo que os valores das medidas simétricas (como o *lift*) indique que o padrão é relevante.

A Tabela 12 mostra um exemplo de análise da regra inversa que evidencia um padrão irrelevante. Com um *lift* valorado em 1,28 para as duas regras, pode-se inferir na regra 1 que quando a contribuição do *pull request* é baseada na linguagem HTML aumenta em 28% as chances da avaliação ser muito rápida. No entanto, para a regra 2 a mesma interpretação não pode ser confirmada, uma vez que os valores das medidas confiança e convicção estão muito abaixo dos valores da regra original.

Tabela 12 – Regra inversa evidenciando regras irrelevantes

#	Regra	Confiança	Convicção	Lift
1	<i>language = HTML</i> → <i>life_time = Very Fast</i>	52%	1,23	1,28
2	<i>life_time = Very Fast</i> → <i>language = HTML</i>	08%	1,02	1,28

Fonte: Elaboração própria.

Além de contribuir para ratificar a correlação entre antecedente e consequente da regra, este filtro representa uma vantagem ao especialista de domínio também no aspecto da visualização, pois permite analisar as regras em pares, facilitando a observação das medidas de interesse assimétricas. Conforme demonstrado no exemplo, este filtro contribui também para identificação de regras irrelevantes, que figura como um dos problemas relacionados ao pós-processamento de regras de associação.

5.2 AVALIAÇÃO COM USUÁRIOS

A avaliação com usuários teve por objetivo validar a solução proposta neste trabalho no tocante a percepção dos usuários em relação as funcionalidades implementadas. Os avaliadores eram alunos do curso de Pós-graduação em Desenvolvimento de Software e Infraestrutura para Internet, da UFAC. A avaliação foi realizada durante a oferta da disciplina Governo Eletrônico, na qual foram abordados tópicos relacionados a mineração de dados governamentais. Ao fim da disciplina, após realizarem algumas atividades utilizando apenas os recursos nativos da ferramenta WEKA, os alunos foram instruídos a minerar regras de associação e realizar a análise das regras obtidas utilizando o *plug-in* WekaPAR. Posteriormente, foi aplicado um questionário para os 40 participantes.

As respostas possíveis cada uma das 12 perguntas formuladas seguem uma escala do tipo *Likert*¹⁶, onde os avaliadores tinham que escolher entre as opções:

¹⁶ A escala *Likert* é um tipo de escala simétrica de intensidade representada normalmente por cinco expressões verbais que vão desde a máxima discordância até a máxima concordância, ou quatro expressões quando o ponto de neutralidade é omitido (BECKER, 2015).

discordo totalmente, discordo, neutro, concordo ou concordo totalmente. A Tabela 13 mostra um resumo das respostas do questionário. As avaliações foram consideradas positivas quando recebiam a resposta concordo ou concordo totalmente, e consideradas negativas quando eram avaliadas em discordo ou discordo totalmente. O resultado completo da aplicação do questionário de avaliação é encontrado no Apêndice B.

Tabela 13 – Resultado do questionário de avaliação

#	Questão	Avaliação (%)		
		Positiva	Neutro	Negativa
1	A solução fornece informações suficientes para análise dos dados?	92,5	7,5	0
2	Você considera que os dados são apresentados com exatidão?	82,5	17,5	0
3	Você acha que os dados são apresentados em um formato útil que permite fácil compreensão e localização das informações na tela?	77,5	15	7,5
4	O tempo de resposta dos comandos é satisfatório?	90	5	5
5	Os nomes que identificam cada opção facilitam a compreensão do que cada ação irá executar?	87,5	7,5	5
6	Você considera que a solução representa alguma vantagem em relação a outros métodos de análise?	72,5	27,5	0
7	A integração com a ferramenta WEKA é satisfatória?	92,5	7,5	0
8	A terminologia utilizada está de acordo com as demais funcionalidades do WEKA e com as tarefas de mineração de dados?	85	15	0
9	A possibilidade de estabelecer valores mínimos para as métricas facilita a busca por regras relevantes?	90	7,5	2,5
10	O filtro para as regras, bem como seu formato, contribui positivamente para a busca por informações relevantes?	92,5	5	2,5
11	Você considera que o uso das caixas de seleção de atributos e rótulos das regras facilita a construção dos filtros?	90	10	0
12	De maneira geral, você considera que a solução proposta cumpre o objetivo de permitir maior facilidade no trabalho de análise de padrões descobertos representados pelas regras de associação?	92,5	7,5	0
Média		87,08	11,04	1,88

Fonte: Elaboração própria.

Numa análise geral verifica-se que a solução foi avaliada positivamente pela maioria predominante dos usuários. Todas as questões obtiveram mais avaliações

positivas do que negativas, sendo que na maioria o percentual de aprovação superou os 90%.

Dentre as questões melhor avaliadas, a questão 12 obteve 92,5% de avaliações positivas e nenhuma negativa. E dentre as questões com pior avaliação está questão 3, que obteve o maior número de avaliações negativas, na ordem de 7,5%, no entanto sendo avaliada positivamente por 77,5% dos usuários.

Um aspecto relevante dos resultados da aplicação do questionário é a elevada quantidade de avaliadores que nem concordavam e nem discordavam do item avaliado. Na questão 6, por exemplo, 27,5% dos avaliadores não se consideram capazes de opinar sobre o item, o que corresponde a 11 usuários. Nas demais questões, ao menos 5% dos avaliadores responderam da mesma forma. Um fator que pode explicar estes números é a heterogeneidade do grupo de avaliadores, uma vez que não foi exigido dos participantes experiência em mineração de dados além do que foi ministrado na disciplina. Assim, é possível que determinados avaliadores não tivessem o conhecimento necessário para julgar algumas questões.

No entanto, é razoável considerar a solução como validada sob a perspectiva do usuário, dado o elevado percentual de avaliações positivas para o questionário aplicado, com média aproximada de 87,08%. Soma-se a este aspecto o fato de que em 7 das 12 questões a maioria absoluta dos avaliadores concedeu a avaliação máxima (concordo totalmente), o que reforça ainda mais a conclusão de que a extensão desenvolvida é uma ferramenta útil na etapa de pós-processamento de regras de associação.

5.3 CONCLUSÃO

Neste capítulo, foi apresentada uma análise das funcionalidades desenvolvidas e como estas auxiliam a avaliação das regras de associação, no intuito de validar a solução proposta. Foi apresentada também uma avaliação realizada com usuários.

Os resultados demonstram que as funcionalidades implementadas permitem maior facilidade no processo de análise das regras de associação, sobretudo por acrescentar à ferramenta WEKA recursos que auxiliam a visualização e seleção das regras mais interessantes. A utilidade destes recursos também foi evidenciada na avaliação realizado com usuários, que registrou um elevado índice avaliações positivas.

No Capítulo que segue, são apresentadas as considerações finais sobre este trabalho de conclusão de curso, além das contribuições dos resultados da pesquisa, e também recomendações para trabalhos futuros.

6 CONSIDERAÇÕES FINAIS

Os problemas relacionados a etapa de pós-processamento de regras de associação decorrem sobretudo do grande volume das bases de dados que, por sua vez, levam a um elevado número de padrões obtidos. Não há, dentre as funcionalidades nativas da ferramenta WEKA, recursos para lidar com esses problemas, principalmente no tocante a visualização das regras extraídas, obrigando o usuário a buscar outras ferramentas para auxiliar o trabalho interpretação das regras geradas.

Este trabalho de conclusão de curso apresentou uma proposta de solução para o problema de analisar padrões representados pelas regras de associação extraídos utilizando a ferramenta WEKA, fornecendo recursos de visualização e filtragem das regras obtidas. Assim, o principal objetivo deste trabalho foi desenvolver uma ferramenta que permitisse ao minerador de dados selecionar as regras mais relevantes por intermédio de parâmetros, na forma de uma extensão integrada a ferramenta WEKA.

Para tal fim, foi realizada uma revisão da literatura acerca do processo de descoberta de conhecimento em bases de dados com ênfase na tarefa de mineração de regras de associação e na etapa de pós-processamento. Neste ponto da pesquisa, foi possível tratar de problemas relacionados a interpretação e análise dos padrões. Posteriormente, a solução foi implementada em linguagem de programação, observando os requisitos constantes na documentação da ferramenta WEKA. A

solução proposta foi então avaliada do ponto de vista de seus recursos para auxiliar o pós-processamento de regras de associação na WEKA, demonstrando como as funcionalidades implementadas poderiam ajudar o minerador de dados a identificar os padrões mais relevantes. Foi realizada ainda uma avaliação sob a perspectiva de usuários da ferramenta WEKA por meio da aplicação de questionário.

Uma das dificuldades enfrentadas no decorrer deste trabalho de pesquisa foi a impossibilidade de realizar uma avaliação com participantes experientes na utilização da WEKA como ferramenta para mineração de regras de associação. No entanto, considera-se útil a avaliação realizada, pois os participantes puderam verificar as dificuldades de analisar as regras de associação utilizando apenas os recursos nativos da WEKA, e posteriormente avaliar se o *plug-in* WekaPAR contribui para facilitar esta atividade.

Em relação aos objetivos deste trabalho, pode-se concluir que estes foram alcançados, uma vez que a solução foi proposta, implementada e validada seguindo todas as etapas e atividades descritas neste relato, obtendo resultados satisfatórios. Conclui-se também que o problema de pesquisa estabelecido foi resolvido, pois, conforme demonstrado neste documento, a extensão desenvolvida fornece recursos que auxiliam o minerador de dados na etapa de pós-processamento de regras de associação, permitindo maior facilidade para analisar os padrões extraídos.

6.1 CONTRIBUIÇÕES

Umas das contribuições deste trabalho é fornecer uma ferramenta para análise de regras de associação de forma integrada ao WEKA. Esta integração permite que o minerador de dados conte com recursos para pós-processamento das regras na própria ferramenta utilizada para extrair os padrões, servindo de alternativa à utilização de outro software para este fim.

Neste sentido, a solução permite lidar com alguns dos problemas mais comuns relacionados ao pós-processamento de regras de associação, tais como o elevado número de regras geradas, a existência de padrões irrelevantes ou ainda regras

redundantes, nas quais alguns atributos não acrescentam informações úteis ao padrão analisado.

Outro aspecto relevante em relação as contribuições deste trabalho diz respeito aos recursos de visualização dos padrões extraídos em contraste com a saída padrão dos algoritmos de mineração de regras de associação do WEKA. A possibilidade de ordenação, de acordo com as medidas de interesse, representa uma vantagem ao especialista de domínio, no sentido que permite analisar as diferentes interpretações que cada medida de interesse pode apresentar.

No entanto, o benefício mais importante consiste nos recursos disponíveis para filtragem de regras tanto pela seleção de atributos e rótulos como pelos valores mínimos de métricas de interesse, incluindo também os filtros para buscar subconjuntos de regras e analisar medidas assimétricas por meio da regra inversa. Ademais, a solução apresentada neste trabalho permite ainda que os diferentes filtros sejam combinados, contribuindo para facilitar o trabalho do minerador de dados.

Considerando as contribuições relatadas pode-se concluir que a ferramenta proposta auxilia o processo de análise e interpretação das regras de associação no WEKA, tornando-o menos complexo para o usuário. Nesta perspectiva, tendo em vista que boa parte dos usuários da ferramenta WEKA a utiliza para fins acadêmicos, é razoável afirmar que o *plug-in* WekaPAR contribui também para auxiliar alunos iniciantes que utilizam a ferramenta WEKA para minerar regras de associação.

6.2 RECOMENDAÇÕES

Mesmo considerando que os objetivos estabelecidos foram plenamente atendidos pela solução proposta, é possível listar algumas recomendações para trabalhos futuros que poderão incrementar as funcionalidades desenvolvidas ou ainda obter melhores resultados, dando continuidade a esta pesquisa. Desta forma, recomenda-se:

- a) Identificar na literatura diferentes abordagens propostas para o pós-processamento de regras de associação e implementá-las na ferramenta;
- b) Realizar um estudo comparativo dos recursos de análise de regras de associação das principais ferramentas de mineração de dados disponíveis no mercado, identificando aspectos negativos e positivos, bem como recursos que poderiam adicionados ao *plug-in* WekaPAR;
- c) Realizar uma avaliação das funcionalidades da ferramenta por meio da elaboração de estudos de casos com especialistas de diferentes domínios que possuam experiência em mineração de dados;
- d) Avaliar a interface gráfica da solução proposta neste trabalho com base nos métodos relacionados a IHC (interação humano-computador), principalmente no tocante ao aspecto de usabilidade.

REFERÊNCIAS

AGRAWAL, R.; IMIELIŃSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. **ACM SIGMOD Record**, v. 22, n. 2, p. 207–216, 1993.

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules. **Proceedings of the 20th International Conference on Very Large Data Bases**, v. 1215, p. 487–499, 1994.

ALE, J. M.; ROSSI, G. H. **An approach to discovering temporal association rules**. Proceedings ACM Symposium on Applied Computing. **Anais...** In: SAC 2000. Villa Olmo, Como, Italy: ACM, 2000.

BECKER, J. L. **Estatística Básica: Transformando Dados em Informação**. Porto Alegre: Bookman Editora, 2015.

BIRD, C.; MENZIES, T.; ZIMMERMANN, T. **The art and science of analyzing software data**. Waltham, MA: Morgan Kaufmann, 2015.

BÖTTCHER, M.; SPOTT, M.; NAUCK, D. **Detecting temporally redundant association rules**. Proceedings Fourth International Conference on Machine Learning and Applications. **Anais...** In: ICMLA 2005. Los Angeles: IEEE, 2005.

BOUCKAERT, R. R.; FRANK, E.; HALL, M.; KIRKBY, R.; REUTEMANN, P.; SEEWALD, A.; SCUSE, D. **WEKA Manual for Version 3-7-12**. Hamilton: University of Waikato, 2014.

BRIN, S.; MOTWANI, R.; ULLMAN, J. D.; TSUR, S. **Dynamic itemset counting and implication rules for market basket data**. ACM SIGMOD Record. **Anais...** In: SIGMOD '97. New York: ACM, 1997.

CHISHOLM, A. **Exploring data with RapidMiner**. Birmingham: Packt Publishing, 2013.

DEITEL, P. **Java como programar**. 8. ed. São Paulo: Pearson Prentice Hall, 2010.

FAYYAD, U.; PIATETSKY-SHAPIO, G.; SMYTH, P. From data mining to knowledge discovery: an overview. In: FAYYAD, U.; PIATETSKY-SHAPIO, G.; SMYTH, P.; UTHURUSAMY, R. (Eds.). . **Advances in knowledge discovery and data mining**. Menlo Park: AAAI Press, 1996a. p. 1–34.

FAYYAD, U.; PIATETSKY-SHAPIO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI magazine**, v. 17, n. 3, p. 37–54, 1996b.

FRIEDL, J. **Mastering Regular Expressions**. 3. ed. Sebastopol: O'Reilly Media, 2006.

GONÇALVES, E. C. Regras de Associações e suas Medidas de Interesse Objetivas e Subjetivas. **INFOCOMP Journal of Computer Science**, v. 4, n. 1, p. 26–35, 2004.

HAHSLER, M.; CHELLUBOINA, S.; HORNIK, K.; BUCHTA, C. The arules R-package ecosystem: analyzing interesting patterns from large transaction data sets. **The Journal of Machine Learning Research**, v. 12, p. 2021–2025, 2011.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The WEKA data mining software: an update. **ACM SIGKDD explorations newsletter**, v. 11, n. 1, p. 10–18, 2009.

HAN, J.; KAMBER, M.; PEI, J. **Data mining: concepts and techniques**. 3. ed. San Francisco: Morgan Kaufmann, 2011.

INMON, W. H. **Building the data warehouse**. 3. ed. New York: J. Wiley, 2002.

KLEMETTINEN, M.; MANNILA, H.; RONKAINEN, P.; TOIVONEN, H.; VERKAMO, A. I. **Finding Interesting Rules from Large Sets of Discovered Association Rules**. Proceedings of the Third International Conference on Information and Knowledge Management. **Anais...** In: CIKM '94. New York, NY, USA: ACM, 1994.

KOTU, V.; DESHPANDE, B. **Predictive analytics and data mining: concepts and practice with RapidMiner**. 1. ed. Waltham, MA: Morgan Kaufmann, 2015.

LI, Y.; NING, P.; WANG, X. S.; JAJODIA, S. Discovering calendar-based temporal association rules. **Data & Knowledge Engineering**, v. 44, n. 2, p. 193–218, 2003.

LIU, B. **Web data mining: exploring hyperlinks, contents, and usage data**. 2. ed. New York: Springer, 2011.

LIU, B.; HSU, W.; MA, Y. **Pruning and Summarizing the Discovered Associations**. Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. **Anais...** In: KDD '99. New York, NY, USA: ACM, 1999.

LIU, B.; MA, Y.; LEE, R. **Analyzing the interestingness of association rules from the temporal dimension**. Proceedings IEEE International Conference on Data Mining. **Anais...** In: ICDM 2001. San Jose: IEEE, 2001.

LOUGHRAN, S.; HATCHER, E. **ANT in action**. 2. ed. Greenwich: Manning, 2007.

MACCUE, C. **Data mining and predictive analysis: intelligence gathering and crime analysis**. Amsterdam: Butterworth-Heinemann, 2007.

PAULA FILHO, W. DE P. **Engenharia de software: fundamentos, métodos e padrões**. 3. ed. Rio de Janeiro: LTC, 2009.

PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional**. 7. ed. Porto Alegre: McGraw-Hill, 2011.

RAPIDMINER. **RapidMiner Studio Manual**, 2014. Disponível em: <<http://docs.rapidminer.com/downloads/RapidMiner-v6-user-manual.pdf>>. Acesso em: 18 mar. 2016

REXER, K. **2013 Data Mining Survey: Highlights**. In: PREDICTIVE ANALYTICS WORLD. Boston, 30 set. 2013.

REXER, K. **2015 Data Science Survey: Highlights**. In: PREDICTIVE ANALYTICS WORLD. Boston, 29 set. 2015.

SILBERSCHATZ, A.; TUZHILIN, A. What makes patterns interesting in knowledge discovery systems. **Knowledge and Data Engineering, IEEE Transactions on**, v. 8, n. 6, p. 970–974, 1996.

SILVA, E. L. DA; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 4. ed. Florianópolis: UFSC, 2005.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TAN, P.; KUMAR, V.; SRIVASTAVA, J. **Selecting the right interestingness measure for association patterns**. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. **Anais...** In: KDD '02. New York: ACM Press, 2002.

TAN, P.; STEINBACH, M.; KUMAR, V. **Introduction to data mining**. Boston: Pearson Addison Wesley, 2006.

TORCHIANO, M.; MORISIO, M. Overlooked aspects of COTS-based development. **IEEE Software**, v. 21, n. 2, p. 88–93, 2004.

TORGO, L. **Data mining with R: learning with case studies**. Boca Raton, FL: Chapman & Hall/CRC, 2011.

WILLIAMS, G. J. Rattle: a data mining GUI for R. **The R Journal**, v. 1, n. 2, p. 45–55, 2009.

WILLIAMS, G. J. **Data Mining with Rattle and R**. New York, NY: Springer, 2011.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data mining: practical machine learning tools and techniques**. 3. ed. Burlington, MA: Morgan Kaufmann, 2011.

APÊNDICES

APÊNDICE A – DOCUMENTO DE REQUISITOS

Documento de Requisitos de Software

Extensão da WEKA para pós-processamento de regras de associação

Revisão 1

Daniel Augusto Nunes da Silva

Rio Branco / AC

2015

1. ANÁLISE DO PROBLEMA

Ao analisar os resultados da extração de regras de associação, o trabalho do minerador de dados pode demandar muito tempo quando necessita analisar um número elevado de regras extraídas. O problema decorre, principalmente, do grande volume de dados disponível, de onde pretende-se obter conhecimento.

O minerador de dados busca, portanto, a utilização de ferramentas no intuito de auxiliar a filtragem destas regras, por intermédio de parâmetros, de forma a obter um conjunto de regras relevantes para o domínio estudado com base em valores de métricas de interesse, tais como suporte, confiança e *lift*. Esta é uma etapa importante do processo de descoberta de conhecimento, uma vez que é comum a existência de redundâncias ou relações irrelevantes nos padrões descobertos.

Dentre as ferramentas disponíveis para mineração de dados, o WEKA é uma das mais difundidas. No WEKA, considerando a tarefa de extração de regras de associação, os recursos para análise dos padrões descobertos são limitados. Isto obriga o usuário a buscar outras ferramentas para complementar o trabalho de mineração de dados iniciado no WEKA.

2. NECESSIDADES BÁSICAS DO CLIENTE

A necessidade básica é permitir ao minerador de dados mais facilidade no pós-processamento de regras de associação, fornecendo recursos para análise dos padrões descobertos de forma integrada à ferramenta WEKA.

Uma das funções da ferramenta WEKA é a extração de conhecimento utilizando algoritmos para obtenção de regras de associação. O WEKA tem suporte à criação de extensões que complementam estas funções da ferramenta.

Assim, o software especificado neste documento será desenvolvido na forma de uma extensão da ferramenta WEKA, para atender a necessidade de integração. Para auxiliar o minerador de dados na etapa de pós-processamento é necessário ainda que o software permita a visualização de regras extraídas, por meio da ferramenta WEKA, e forneça recursos para filtrar os padrões, de acordo com os

atributos e seus rótulos, bem como por meio das métricas de interesse disponíveis na WEKA.

3. ESTUDO DE VIABILIDADE

O projeto é considerado viável considerando os aspectos técnicos, econômicos e legais, descritos a seguir.

3.1. Viabilidade Técnica

Do ponto de vista técnico o software é viável, pois seu desenvolvimento será baseado na linguagem de programação Java, que é bastante difundida e possui boa documentação, o que facilita uma busca por informações quando necessário. A viabilidade técnica também se justifica no suporte ao desenvolvimento de *plug-ins* que a ferramenta WEKA possui, bem como a documentação relacionada disponível.

3.2. Viabilidade Econômica

O desenvolvimento da solução pode ser considerado economicamente viável pois não incorre em custos adicionais decorrentes do processo de desenvolvimento, uma vez que consiste em um projeto acadêmico. Também não será necessário investir em aquisição de equipamentos ou licenciamento de softwares para desenvolvimento.

3.3. Viabilidade Legal

A solução proposta não possui qualquer impedimento legal, pois não está em desacordo com nenhuma lei, seja na esfera municipal, estadual ou federal. Devido ao WEKA estar licenciado sob a GPL (*General Public License*), o software desenvolvido também deverá utilizar o mesmo licenciamento, bem como atenderá a exigência da GPL de disponibilização do código-fonte quando for distribuído publicamente, o que será viabilizado utilizando o GitHub.

4. MISSÃO DO SOFTWARE

Permitir ao minerador de dados a seleção de regras de associação, de forma integrada a ferramenta WEKA, por intermédio de parâmetros na etapa de pós-processamento.

5. LIMITES DO SISTEMA

ID	Funcionalidade	Justificativa
L1	Realizar pré-processamento de dados e gerar regras de associação	Não faz parte dos objetivos do software substituir funcionalidades já atendidas pela ferramenta WEKA.

6. BENEFÍCIOS GERAIS

ID	Benefício
B1	Maior facilidade para identificar padrões relevantes
B2	Integração com a ferramenta WEKA

7. RESTRIÇÕES

ID	Restrição	Descrição
R1	Compatibilidade	O software não será compatível com versões da ferramenta WEKA que não suportam o uso de pacotes para adicionar novas funcionalidades.

8. ATORES

ID	Ator	Descrição
A1	Minerador de dados	Usuário que já utiliza a ferramenta WEKA para extrair padrões na forma de regras de associação.

9. REQUISITOS FUNCIONAIS

ID	Funcionalidade	Necessidades	Prioridade
RF1	Visualizar regras	Visualizar as regras extraídas numa tabela, exibindo antecedente e consequente da regra, além das medidas de interesse, em colunas separadas.	Alta
RF2	Filtrar regras	Aplicar um filtro no conjunto de regras tendo como entrada um termo de busca determinado pelo usuário.	Alta
RF3	Ordenar as regras	O software deve permitir que as regras possam ser ordenadas com base nos valores das medidas de interesse.	Alta
RF4	Aplicar filtro em um lado específico da regra	Possibilitar que um filtro seja aplicado a um determinado lado da regra (antecedente ou consequente), e que estes possam ser combinados em um mesmo filtro.	Alta
RF5	Explorar subconjunto de regras	Filtrar regras com base no subconjunto de atributos do antecedente de uma dada regra selecionada, mantendo o mesmo consequente.	Alta
RF6	Salvar as regras geradas	Salvar no disco o conjunto de regras exibidos na tabela.	Média
RF7	Exportar as regras	Exportar as regras visualizadas na tabela utilizando o formato CSV.	Baixa
RF8	Regra inversa	Exibir a regra inversa de uma dada regra selecionada, isto é, invertendo as posições do antecedente e consequente.	Alta
RF9	Filtrar regras com base nas métricas	A solução deve permitir estabelecer valores mínimos para as métricas disponíveis na WEKA e utilizar estes parâmetros como filtro para as regras.	Alta
RF10	Selecionar atributos	O usuário poderá filtrar as regras selecionando os atributos de seu interesse e seus respectivos rótulos para o antecedente e consequente da regra.	Alta
RF11	Combinar filtros	A solução deve permitir que os filtros aplicados ao antecedente e consequente das regras possam ser combinados com	Alta

ID	Funcionalidade	Necessidades	Prioridade
		os valores mínimos das métricas de interesse determinados pelo usuário.	

10. REQUISITOS NÃO-FUNCIONAIS

ID	Requisito	Categoria
RNF1	A extensão deve implementar os recursos de pós-processamento na janela principal do Explorer da WEKA por meio de uma nova guia.	Integração
RNF2	O usuário deverá ser informado sobre o andamento da execução das rotinas por meio da barra de status da WEKA.	Usabilidade
RNF3	O software deverá ser distribuído no formato de um pacote para WEKA.	Integração
RNF4	Os filtros aplicados as regras deve suportar o uso de expressões regulares.	Implementação
RNF5	A interface gráfica deve acompanhar o padrão visual da ferramenta.	Usabilidade
RNF6	A solução deve ser desenvolvida utilizando a linguagem de programação Java, a mesma utilizada no projeto da WEKA.	Implementação
RNF7	O software deve ser compatível com o requisito mínimo de versão do Java exigido pela ferramenta WEKA. Versões a partir da 3.7.1 exigem no mínimo o Java 6.	Implementação

11. REQUISITOS DE HARDWARE

O desempenho da execução da tarefa de minerar regras de associação na ferramenta WEKA está diretamente relacionado a quantidade de memória RAM disponível. Em geral, para casos onde um grande número de regras é gerado, o software exige muita memória. A exigência será ainda maior quando for utilizada a solução especificada neste documento. No entanto, não há documentação dos requisitos mínimos de hardware para o WEKA.

Desta forma, as configurações apresentadas a seguir estão baseadas nos recursos necessários para funcionamento do ambiente Java, tecnologia na qual o WEKA é baseado, bem como na experiência do uso da ferramenta WEKA.

11.1. Configuração mínima

- 512 MB de memória RAM;
- 256 MB de espaço em disco.

11.2. Configuração recomendada

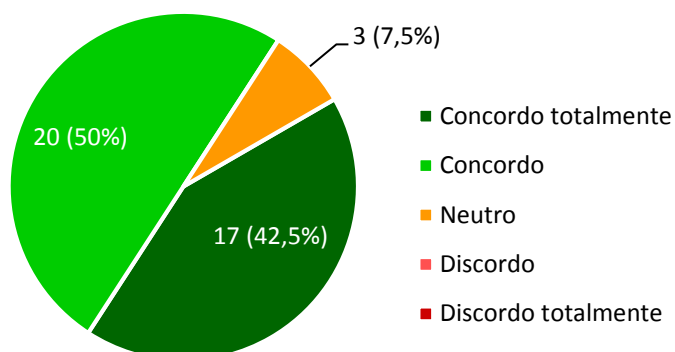
- 8GB de memória RAM;
- 10 GB de espaço em disco.

12.FERRAMENTAS DE DESENVOLVIMENTO E LICENÇA DE USO

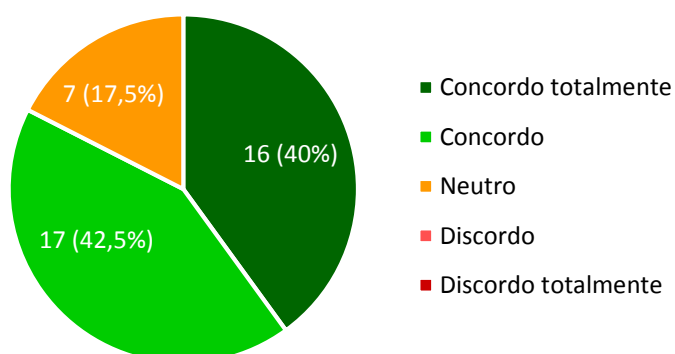
- a) Eclipse Luna 4.4.1 (Eclipse Public License - v1.0);
- b) WEKA 3.7.12 (General Public License – v3.0);
- c) Java SE Development Kit 7 (Oracle Binary Code License);
- d) Apache Ant (Apache License - v2.0).

APÊNDICE B – RESULTADOS DO QUESTIONÁRIO DE AVALIAÇÃO

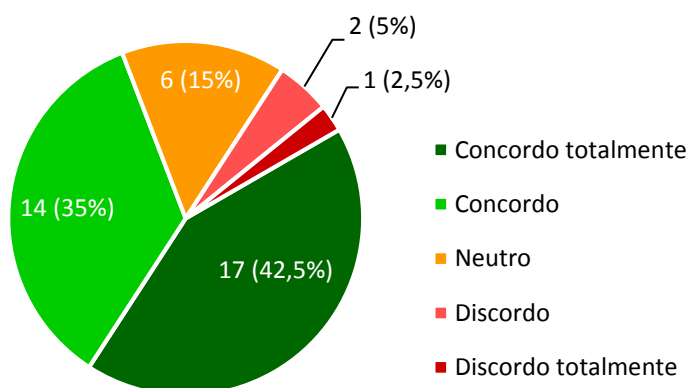
1. A solução fornece informações suficientes para análise dos dados?



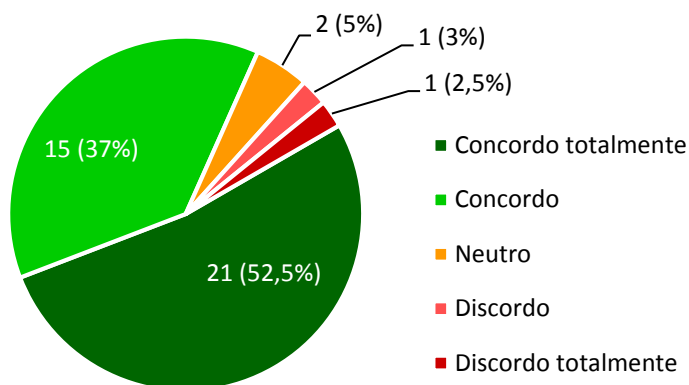
2. Você considera que os dados são apresentados com exatidão?



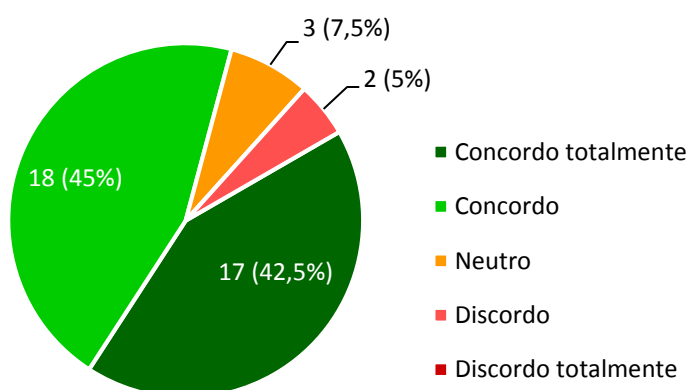
3. Você acha que os dados são apresentados em um formato útil que permite fácil compreensão e localização das informações na tela?



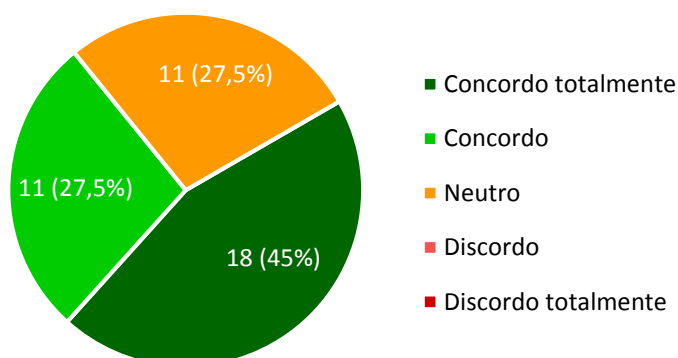
4. O tempo de reposta dos comandos é satisfatório?



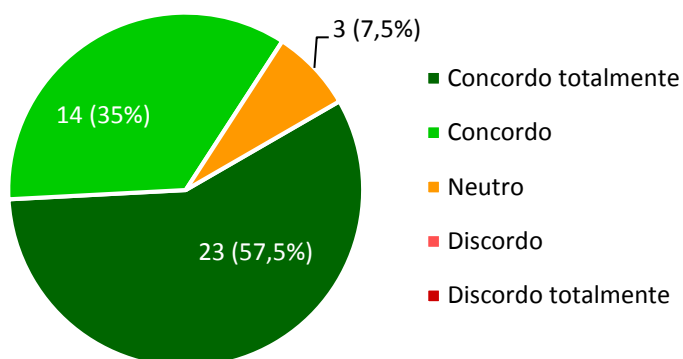
5. Os nomes que identificam cada opção facilitam a compreensão do que cada ação irá executar?



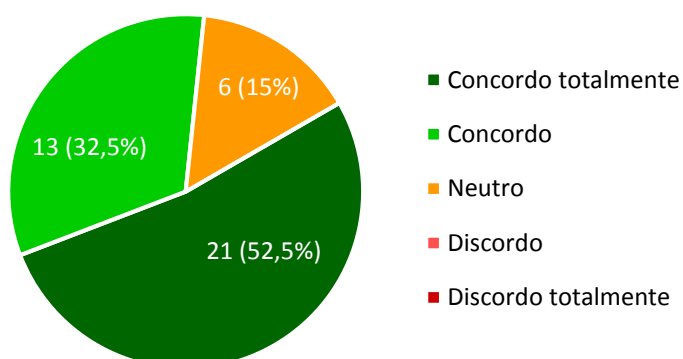
6. Você considera que a solução representa alguma vantagem em relação a outros métodos de análise?



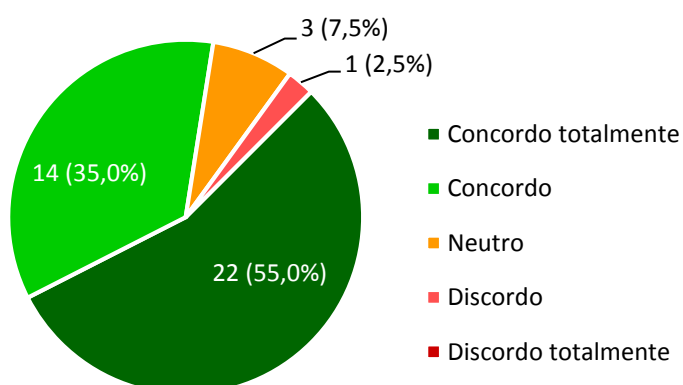
7. A integração com a ferramenta WEKA é satisfatória?



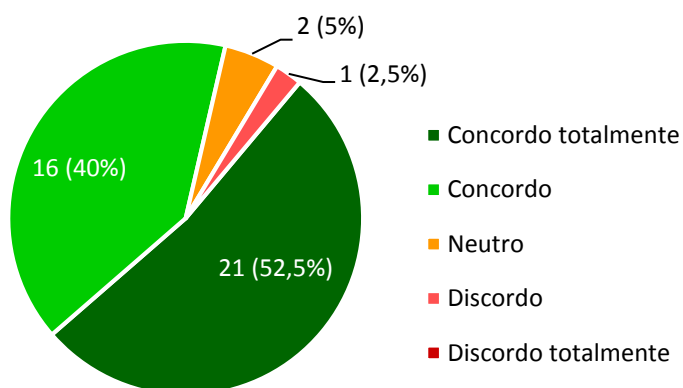
8. A terminologia utilizada está de acordo com as demais funcionalidades da WEKA e com as tarefas de mineração de dados?



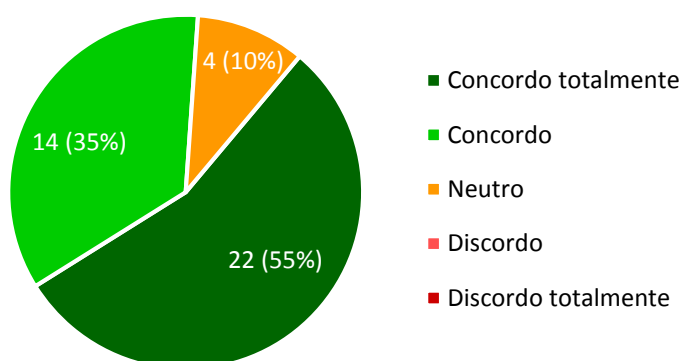
9. A possibilidade de estabelecer valores mínimos para as métricas facilita a busca por regras relevantes?



10. O filtro para as regras, bem como seu formato, contribui positivamente para a busca por informações relevantes?



11. Você considera que o uso das caixas de seleção de atributos e rótulos das regras facilita a construção dos filtros?



12. De maneira geral, você considera que a solução proposta cumpre o objetivo de permitir maior facilidade no trabalho de análise de padrões descobertos representados pelas regras de associação?

