UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GUSTAVO RAFAEL VALIATI

USO DE TÉCNICAS E FERRAMENTAS DE MINERAÇÃO DE DADOS NA EXTRAÇÃO DE INFORMAÇÕES SOBRE O COMPORTAMENTO DE USO DOS RECURSOS DA INTERNET NA UTFPR - CÂMPUS MEDIANEIRA

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2013

GUSTAVO RAFAEL VALIATI

USO DE TÉCNICAS E FERRAMENTAS DE MINERAÇÃO DE DADOS NA EXTRAÇÃO DE INFORMAÇÕES SOBRE O COMPORTAMENTO DE USO DOS RECURSOS DA INTERNET NA UTFPR - CÂMPUS MEDIANEIRA

Trabalho de Diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – CSTADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Msc. Paulo Lopes de Menezes.

Coorientador: Prof. Msc. Alan Gavioli.

MEDIANEIRA

2013

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

USO DE TÉCNICAS E FERRAMENTAS DE MINERAÇÃO DE DADOS NA EXTRAÇÃO DE INFORMAÇÕES SOBRE O COMPORTAMENTO DE USO DOS RECURSOS DA INTERNET NA UTFPR - CÂMPUS MEDIANEIRA

Por

Gustavo Rafael Valiati

Este Trabalho de Diplomação (TD) foi apresentado às 10h20min do dia 27 de março de 2013, como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, *Câmpus* Medianeira. Os acadêmicos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado com louvor e mérito.

Prof. MSc. Paulo Lopes de Menezes
UTFPR – *Câmpus* Medianeira
(Orientador)

Prof. MSc. Ricardo Sobjak
UTFPR – *Câmpus* Medianeira
(Convidado)

Prof. MSc. Pedro Luiz de Paula Filho
UTFPR – *Câmpus* Medianeira
(Convidado)

Prof. MSc. Juliano Rodrigo Lamb
UTFPR – *Câmpus* Medianeira
(Responsável pelas atividades de TCC)

RESUMO

VALIATI, Gustavo Rafael. Uso de Técnicas e Ferramentas de Mineração de Dados na Extração de Informações Sobre o Comportamento de Uso dos Recursos da Internet na UTFPR - Câmpus Medianeira. Trabalho de Conclusão do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Medianeira, 2013.

O grande crescimento da capacidade de gerar, transmitir e armazenar dados em meios digitais, tem superado a capacidade humana de extração de conhecimento destes dados. A Mineração de Dados é o processo que surgiu nas últimas décadas, justamente com o objetivo de resolver o problema. Este trabalho apresenta a aplicação da Mineração de Dados, como estudo de caso, em uma grande quantidade de dados em *logs*, produzidos pelo Squid, em servidores de compartilhamento de Internet, a fim de extrair conhecimento requerido pelo administrador de redes. O trabalho aborda, de maneira detalhada, a realização das etapas da Mineração de Dados, e também alguns empecilhos que atrapalharam a execução de parte do projeto, como: inviabilidade de construção de uma ferramenta automatizada para o processo de Mineração de Dados; incapacidade de determinado *hardware* suportar processamento de dados requerido; necessidade do uso de novas estratégias na criação de arquivos ARFF grandes, para tornar possível a ferramenta Weka aplicar as tarefas de mineração. Ainda, este trabalho apresenta uma ferramenta de preprocessamento e transformação de dados, criada especificamente para o ambiente encontrado. E como resultado da mineração são apresentados padrões encontrados nos *logs* juntamente com amostras de interpretações possíveis. Por fim, são listadas algumas oportunidades de novos trabalhos.

Palavras-chave: Mineração de Dados; KDD; Logs.

ABSTRACT

VALIATI, Gustavo Rafael. Uso de Técnicas e Ferramentas de Mineração de Dados na Extração de Informações Sobre o Comportamento de Uso dos Recursos da Internet na UTFPR - Câmpus Medianeira. Trabalho de Conclusão do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Medianeira, 2012.

The large increase in capacity to generate, transmit and store data in digital format, has exceeded the human capacity of knowledge extraction of these data. The Data Mining is the process emerged in recent decades, precisely in order to solve the problem. This paper presents the application of Data Mining, as a case study, in a large amount of data in logs, generated by Squid, from Internet sharing servers, to extract knowledge required by the network administrator. The paper discusses, in detail, performing the steps of Data Mining, and also some obstacles that hindered the execution of part of the project, such as inviability of constructing an automated tool to process Data Mining; inability of specific hardware to process the required data; necessity of using new strategies in the creation of large ARFF files, to enable Weka tool to apply mining tasks. Further, this paper presents a tool for preprocessing and data transformation, specifically designed for the environment encountered. And as a result of mining are presented patterns found in the logs along with samples of possible interpretations. At last, a list of some opportunities for new papers is presented.

Keywords: Data Mining; KDD; Logs.

LISTA DE SIGLAS

ACL	*Access Control List*
API	*Application Programming Interface*
ARFF	Attribute-Relation File Format
GPL	*General Public Licence*
HTTP	*Hypertext Transfer Protocol*
KDD	*Knowledge Discovery in Databases*
RAID	*Redundant Array of Independent Disks*
RAM	*Random-access memory*
RPM	Rotações por minuto
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
UTFPR	Universidade Tecnológica Federal do Paraná
XML	*Extensible Markup Language*
XRFF	*Extensible Attribute-Relation File Format*

SUMÁRIO

1 INTRODUÇÃO

Com base na 5ª Pesquisa Sobre Uso das Tecnologias da Informação e da Comunicação no Brasil TIC Empresas 2009, que analisou 3,7 mil empresas com 10 ou mais funcionários em todo o território nacional, é possível observar que 93% destas empresas utilizam o recurso de Internet. E das empresas com mais de 100 funcionários, todas apresentaram o recurso (CGI.BR, 2010).

Nota-se que já no ano de 2009 as instituições tomavam o uso da Internet como um recurso essencial, quanto maior fosse a empresa.

A Internet tem vários papéis dentro da instituição. Dentre eles consta: disponibilização de serviços oferecidos pela instituição para usuários externos, disponibilização do acesso dos usuários internos para serviços externos, navegação aleatória na web, *stream* de áudio e vídeo, *download* e *upload* de arquivos, entre outros.

A pesquisa TIC Empresas 2009 mostrou também que 66% das empresas possuem algum tipo de medida de restrição ao acesso de empregados a determinados tipos de páginas da Internet. As principais restrições registradas são aos**:** *sites* pornográficos (62%), *sites* de relacionamento (48%), *sites* de comunicação (41%) e *e-mails* pessoais (30%) (CGI.BR, 2010). Destas instituições, as de porte mais elevado geralmente possuem um administrador de rede, ou seja, o profissional responsável pela configuração e manutenção do servidor de rede, implementando as políticas de uso da Internet definidas pela empresa.

Além da configuração e manutenção do servidor, o administrador de rede, precisa trabalhar para que os recursos sejam aproveitados da melhor maneira possível, já que a banda de Internet é geralmente limitada. Por isso, é importante saber, por exemplo, que em determinado horário do dia, um sistema web empresarial é muito utilizado pelos funcionários, consumindo boa parte da capacidade da rede. Assim, o administrador deve trabalhar para que atividades na rede de menor importância não venham impactar na experiência de uso dos usuários para atividades realmente importantes.

Os administradores tendem a gerar relatórios dos acessos dos usuários, para extrair informações úteis para melhoria das políticas de acesso, para entender problemas e perceber comportamentos.

Na maioria das distribuições Linux atuais, a ferramenta Squid é utilizada como servidor *proxy* padrão, ou seja, servidor de compartilhamento de Internet. Juntamente com o Squid a ferramenta Sarg é geralmente utilizada para gerar relatórios do uso da Internet (MORIMOTO, 2006). O Sarg e ferramentas semelhantes, geram relatórios de baixo nível como: o que determinado IP da rede acessou em determinado período, ou quais IP's acessaram determinado site. Estas informações são importantes e úteis, porém para uma análise mais minuciosa. Para identificar o comportamento, tendências e relatórios mais expressivos dos usuários de uma maneira ágil, estes relatórios não são adequados.

Verifica-se que a análise completa do uso de Internet por parte dos usuários finais, é de grande importância para estas instituições, no âmbito da gerência dos recursos de Internet e das políticas da instituição. Para esta análise mais completa, entende-se que pode ser necessário realizar mineração dos dados dos servidores, de modo a entender o comportamento dos acessos dos usuários, e poder extrair destas informações tomadas de decisão que atendam as políticas da empresa, na forma da melhoria do serviço de gerência do recurso de Internet.

1.1 OBJETIVO GERAL

Possibilitar a análise de arquivos de *log* de servidores *proxy/firewall*, para apresentação do resultado ao administrador de rede em forma de gráficos e/ou tabelas empregando Mineração de Dados.

1.2 OBJETIVOS ESPECÍFICOS

- Identificar dados em arquivos de *log* para gerar informações úteis ao administrador de redes;
- Identificar as ferramentas e tecnologias necessárias para aplicação;
- Identificar os atributos necessários pelo processo de mineração dos dados, e de que forma poderão ser extraídos dos arquivos de *log*;

- Construir um algoritmo para extração de dados dos arquivos de *log*, conforme a identificação realizada;
- Utilizar a ferramenta de mineração de dados Weka para atuar nos dados extraídos, reconhecendo padrões para produção dos relatórios, utilizando para regras de associação o algoritmo Apriori.

1.3 JUSTIFICATIVA

Observando o problema da falta de ferramentas que venham gerar relatórios que permitam a análise ágil do comportamento do usuário na Internet, surge então a possibilidade de desenvolver um sistema (conjunto de métodos e ferramentas) para produção de relatórios com o objetivo de auxiliar o administrador de redes na tomada de decisão em relação à gerência dos serviços e recursos de Internet.

As atividades do sistema seriam semelhantes ao processo de descoberta de conhecimento em bases de dados (do inglês, *Knowledge Discovery in Databases* - KDD) que contém como 5ª etapa o processo de Mineração de Dados, que envolveriam (FAYYAD, PIATETSKY-SHAPIRO, SMYTH; 1996):

- Definição de metas: definir quais os conjuntos de dados serão analisados. O administrador define quais dados devem ser estudados e qual o objetivo final;
- Aquisição dos dados: obter os arquivos de *log* dos servidores de rede correspondentes à necessidade;
- Processamento/Transformação: filtragem dos dados úteis, e preparação para mineração. Um aplicativo/algoritmo analisa e extrai do arquivo de *log* os dados e formato necessários para a mineração;
- Mineração: aplicação de algoritmos para descoberta de padrões nos dados. Uso de uma ferramenta de mineração para realizar este processo;
- Interpretação: Os padrões identificados são apresentados ao administrador para análise e tomada de decisão. Aplicativo com interface web para apresentação dos relatórios ao administrador.

O desenvolvimento do sistema estará voltado para a simplicidade, para que o sistema seja atrativo para os administradores de redes, de maneira que possam instalá-lo em seu ambiente de uma forma facilitada, e de uma maneira descomplicada gerar os relatórios para interpretação.

Como produto final, o projeto trará contribuição para necessidade dos administradores de rede: relatórios oriundos da mineração dos dados de uso da Internet pelos usuários.

1.4 ESTRUTURA DO TRABALHO

O trabalho está organizado em 4 capítulos: introdução, referencial teórico, estudo de caso e considerações finais. Cada um dos capítulos é explicado a seguir.

O primeiro capítulo dá a introdução do trabalho, explicando os objetivos geral e específicos, e a justificativa de sua realização.

O segundo capítulo trata das referências teóricas envolvidas no trabalho. O capítulo se subdivide em dois outros: descoberta de conhecimento em bases de dados, e serviços da Internet na rede local. O primeiro do conceito do KDD, a descrição do processo, descrição de algumas tarefas de mineração, descrição de alguns algoritmos das tarefas e fala sobre a ferramenta Weka. O segundo dos principais serviços de Internet encontrados em redes locais, como *proxy*, *firewall* e servidor de páginas.

O terceiro capítulo aborda o estudo de caso realizado no trabalho. Em seus subcapítulos são descritas detalhadamente as etapas do processo realizado, desde o início até os resultados obtidos com a mineração. Também descreve problemas encontrados que impossibilitaram a execução de alguns objetivos.

No quarto capítulo é apresentada a conclusão do trabalho e também ideias para trabalhos futuros.

2. REFERENCIAL TEÓRICO

Este capítulo apresentará o referencial teórico referente à elaboração deste trabalho. É dividido em dois principais assuntos: descoberta de conhecimento em banco de dados e servidores de Internet em redes locais.

2.1 DESCOBERTA DE CONHECIMENTO EM BANCO DE DADOS

2.1.1 Conceito

Na última década, houve grande avanço em diversas áreas tecnológicas, que proporcionaram um crescimento elevado nas capacidades de gerar, coletar, armazenar e transmitir dados digitais. O baixo custo de equipamentos, como computadores e demais dispositivos, trouxeram um aumento significativo no número de usuários conectados em rede. Com isso, a Internet passou a conter informações de diversos tipos, origens, formatos e fins, sendo geradas e/ou acessas a todo momento, em quantidades imensuráveis (SANTOS, 2009).

Com a grande quantidade de informações disponíveis na Internet, apesar de facilmente acessíveis, torna-se difícil fazer a localização do conteúdo alvo. Esta é uma preocupação forte, sendo possível notar que algumas grandes empresas trabalham arduamente para justamente realizar a organização de imensas bases de dados, a fim de apresentar de uma maneira útil a informação para seus usuários. Ferramentas web como o Google Search, Bing e SourceForge, trabalham neste aspecto (SANTOS, 2009).

Além das informações livres na Internet, organizações do mundo todo possuem em seus sistemas de informação, grandes bases de dados particulares que crescem diariamente. E para estas instituições, tomar decisões inteligentes, seguras e confiáveis é parte fundamental de seu negócio. Para isso elas tendem a investir em extração de conhecimento de suas próprias bases de dados, conhecimentos estes que serão diretamente utilizados para tomada de decisões que influenciam em seus processos gerencias (MACEDO; MATOS, 2010).

Este crescimento do volume de dados, passou a gerar a necessidade imediata de técnicas e ferramentas capazes de transformar, de forma inteligente e automática, indefinidas quantidades de dados em informações úteis, ou seja, em conhecimento. Essas informações, estão geralmente implícitas sob enormes quantias de dados, e não podem ser descobertas, ou no mínimo, facilmente identificadas utilizando-se sistemas convencionais (SFERRA; CORRÊA, 2003).

Segundo Macedo e Matos (2010), o processo que visa atender a esta necessidade, ao fato de se obter conhecimento a partir de bases de dados, dá significado ao termo Descoberta de Conhecimento em Banco de Dados (do inglês, *Knowledge Discovery in Databases* – KDD). Este termo é sinônimo da expressão Mineração de Dados (em inglês, *Data Mining*), que coincide com o nome de uma das etapas do próprio KDD (GALVÃO; MARIN, 2009).

Segundo Fayyad, Piatetsky-Shapiro e Smyth (1996), KDD é um processo não-trivial da identificação de padrões válidos, novos, potencialmente úteis e entendíveis em dados.

> A mineração de dados tenta transformar muita desinformação (na forma de dados espalhados) em informações úteis, criando modelos e regras. Sua meta é usar os modelos e regras para prever um comportamento futuro, melhorar seu negócio, ou apenas explicar coisas que caso contrário não seria possível explicar. Estes modelos podem confirmar o que já pensávamos, ou ainda melhor, podem achar coisas novas em nossos dados que nem sabíamos que existiam (ABERNETHY, 2010).

Para melhor entender o processo KDD é importante entender a diferença entre dado, informação e conhecimento. Observa-se a pirâmide na Figura 1 (MACEDO; MATOS, 2010).

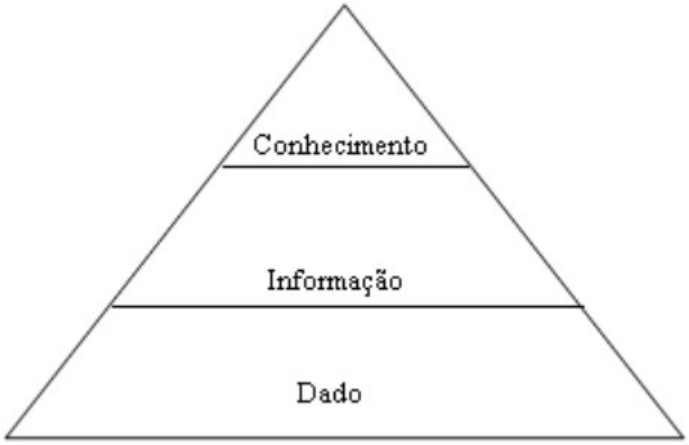

Figura 1: Pirâmide do conhecimento.
Fonte: MACEDO; MATOS, 2010.

Os dados que constituem a base da pirâmide são interpretados como o conjunto de dados quaisquer coletados e armazenado pelos recursos das tecnologias da informação. A

informação, no segundo nível da pirâmide, se refere aos dados organizados, selecionados, com significado e contexto. E o topo da pirâmide, onde encontra-se o conhecimento, condiz ao conjunto de padrões que podem definir a relação entre dados e informações (MACEDO; MATOS, 2010).

2.1.2 O processo KDD

Segundo Fayyad, Piatetsky-Shapiro e Smyth (1996), o processo KDD é interativo e iterativo, pois envolve várias etapas com muitas decisões realizadas pelo usuário. O processo é dividido em cinco principais etapas: seleção, preprocessamento, transformação, mineração e interpretação. A Figura 2 ilustra as fases do processo.

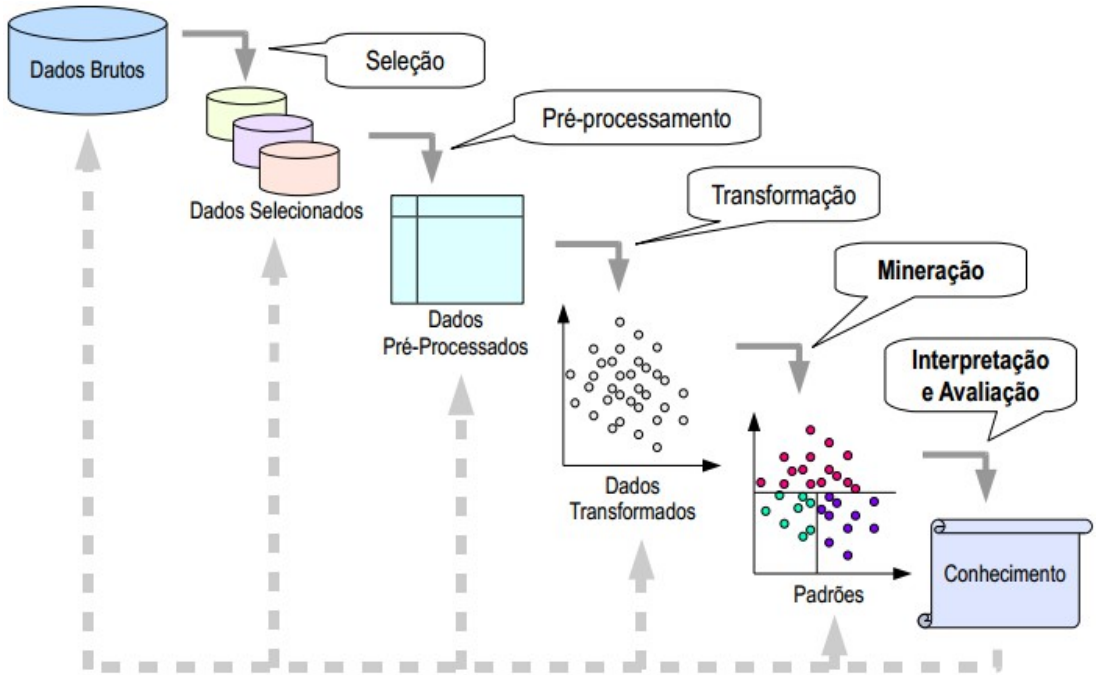

Figura 2: Etapas KDD.
Fonte: SANTOS, 2009.

Antes da primeira etapa (seleção), é necessário ter-se obtido o entendimento do problema alvo, o entendimento do domínio da aplicação, a relação dos conhecimentos prioritários e os objetivos na visão quem solicitou a mineração (FAYYAD, PIATETSKY-SHAPIRO, SMYTH; 1996).

A seleção consiste na escolha do conjunto de dados, do foco em um subconjunto de variáveis ou amostra de dados, aos quais a descoberta de conhecimento deverá ser aplicada (FAYYAD, PIATETSKY-SHAPIRO, SMYTH; 1996).

A segunda etapa (preprocessamento), consiste no preprocessamento dos dados coletados na primeira etapa. Envolve a realização da limpeza, ou seja, remoção dos dados desnecessários, aplicação de estratégias para dados faltantes, normalizações, indexação de sequências temporais, com a finalidade de reduzir os possíveis ruídos (FAYYAD, PIATETSKY-SHAPIRO, SMYTH; 1996).

Na terceira etapa, a transformação, contempla a redução e projeção a fim de encontrar características úteis para a representação dos dados em função do objetivo. Utilizando da redução da dimensionalidade ou transformação de métodos, o número efetivo de variáveis consideráveis pode ser reduzido, ou representações invariáveis podem ser determinadas para os dados (FAYYAD, PIATETSKY-SHAPIRO, SMYTH; 1996).

A mineração de dados, referente à quarta etapa, estabelece a escolha do método de mineração referente ao objetivo estabelecido no início do processo. Os métodos podem ser associação, classificação, agrupamento, regressão, entre outros. Esta etapa ainda condiz com a escolha de algoritmos para as tarefas, e a busca por padrões utilizando-se do algoritmo e dos dados (FAYYAD, PIATETSKY-SHAPIRO, SMYTH; 1996).

E por último, a interpretação dos resultados obtidos, ou seja, a análise dos padrões encontrados pelas tarefas de mineração aplicadas, a avaliação do atendimento aos objetivos propostos, e a aplicação do conhecimento obtido (FAYYAD, PIATETSKY-SHAPIRO, SMYTH; 1996).

2.1.2.1 Etapa de Mineração de Dados do processo KDD

A etapa de Mineração de Dados, segundo Santos (2009), refere-se ao conjunto de técnicas que quando aplicadas viabilizam o aprendizado de padrões a partir de um conjunto de dados, permitindo explicações sobre a natureza dos dados, e previsões a partir dos padrões encontrados. A mineração também envolve em definir-se modelos apropriados ou a

determinação de padrões através dos dados observados (FAYYAD, PIATETSKY-SHAPIRO, SMYTH; 1996).

Segundo Fayyad, Piatetsky-Shapiro e Smyth (1996), esta etapa do processo envolve a aplicação de repetidas iterações de métodos particulares da Mineração de Dados.

De acordo com Santos (2009), existem duas principais categorias da mineração de dados:

- A preditiva, que trata do uso de atributos do conjunto de dados para prever valores desconhecidos ou futuros para um conjunto de dados relacionado;
- A descritiva, que se detém na descoberta de padrões que descrevem os dados e que podem ser interpretados por humanos.

Para encontrar os padrões, podem ser usadas tarefas de mineração como: classificação, regressão, sumarização, agrupamento, modelagem de dependência, e detecção de mudanças ou desvios (SANTOS, 2009).

2.1.2.1.1 Regras de Associação (Modelagem de Dependência)

Segundo Fayyad, Piatetsky-Shapiro e Smyth (1996), a associação é uma técnica de modelagem de dependência. Com ela é possível identificar um modelo que apresenta dependências (regras) significativas entre valores de um atributo de um conjunto de dados.

A tarefa de associação consiste em identificar e descrever associações entre variáveis no mesmo item ou associações entre itens diferentes que ocorram simultaneamente, de forma frequente em banco de dados (GALVÃO; MARIN, 2009).

Um dos algoritmos mais utilizados para a tarefa de associação, é o Apriori. Quando este algoritmo é aplicado em um conjunto de dados, possibilita encontrar regras do tipo “se X está presente então Y também tende a estar”, ou seja, transações da base de dados analisada que contém “X” tendem a conter “Y” (CORREA, 2004).

Segundo Amo (2004), o Apriori foi proposto em 1994 pela equipe de pesquisa do Projeto QUEST da IBM que deu origem ao *Intelligent Miner*.

Conforme Agrawal (1994), o algoritmo precisa ser alimentado com dois parâmetros essenciais: suporte e confiança. O suporte define o número de ocorrências mínimas que uma

regra encontrada deve possuir para se tornar válida, e a confiança define o valor mínimo de precisão da regra. Os dois valores são obtidos da seguinte forma:

- Suporte: número de ocorrências com X e Y, dividido pelo número total de registros;
- Confiança: número de ocorrências com X e Y, dividido pelo número de ocorrências de X.

O algoritmo é executado em 3 etapas: geração dos candidatos, poda dos candidatos e validação dos candidatos. Na primeira são vasculhadas por todas as regras possíveis, na segunda são eliminados os candidatos que não atingiram a confiança mínima e a terceira valida os candidatos em relação ao valor de suporte (AMO, 2004).

2.1.2.1.2 Classificação

A classificação é a tarefa da mineração mais utilizada, já que é comum dos seres humanos o ato de classificar (AMORIM, 2006).

Segundo Galvão e Marin (2009), a tarefa de classificação se refere a predição de uma variável categórica. Tem o objetivo de descobrir uma função que mapeie um conjunto de registros em um conjunto de variáveis predefinidas, denominadas classes. E esta função descoberta, é aplicada em novos registros, com a tentativa de descobrir a classe em que tais registros se enquadram. Os algoritmos mais comuns para classificações são: redes neurais, *Back-Propagation*, classificadores Bayesianos e algoritmos genéticos.

Segundo Santos (2009), a classificação é realizada utilizando valores de atributos como exemplos e de suas classes já informadas (supervisionado). O algoritmo aprenderá que testes e valores devem ser aplicados aos novos atributos para decidir por uma classe.

Um dos algoritmos mais conhecidos para a tarefa de classificação é o ID3. Este é um algoritmo que utiliza árvore de decisão. Sua elaboração, baseia-se em sistemas de inferência e nos conceitos de aprendizagem. O ID3 escolherá um dos atributos da melhor maneira possível, para se tornar a raiz da árvore, a fim de separar o conjunto inicial em conjuntos menores e mais homogêneos. O algoritmo utiliza de uma equação de ganho para decidir quais atributos serão raiz e nós da árvore (SOUZA, 2008).

2.1.2.1.3 Regressão (Estimativa)

A regressão tem como meta a descoberta de uma função preditiva, semelhante à classificação, porém com o objetivo de calcular um valor numérico real ao invés de uma classe descritiva. Assim como na classificação, exemplos de transações do banco são usadas para criar a função capaz de predizer novos valores numéricos desconhecidos (SANTOS, 2009).

Segundo Macedo e Matos (2010), a estatística, redes neurais, dentre outras áreas são responsáveis por possibilitar a implementação da tarefa de regressão.

2.1.2.1.4 Agrupamento (Clusterização)

A tarefa de agrupamento baseia-se na segmentação dos dados. Não existem classes pré-definidas, cabendo ao algoritmo descobrir as classes (grupos/*clusters*), realizando uma classificação não supervisionada com a utilização de medidas de similaridade (SOUZA, 2008).

Segundo Bogorny (2003), o agrupamento é uma das técnicas de mineração mais utilizadas devido a sua habilidade de identificar estruturas diretamente através dos dados, sem haver o conhecimento prévio dos mesmos.

O K-means é um algoritmo para análise dos dados e a criação dos grupos. Este algoritmo pode receber como parâmetro a quantidade de grupos desejáveis. O conjunto de dados é entregue ao algoritmo na forma de matriz de dissimilaridade entre objetos. Nesta matriz, o objetivo é calcular a distância entre os objetos através de um objeto de referência (centróide), durante várias vezes até não ocorram alterações significativas (SOUZA, 2008).

Segundo Souza (2008), a escolha do algoritmo para o agrupamento deve observar alguns pontos importantes:

- O objetivo da aplicação: sempre que a aplicação requer menor distância entre o centróide em relação aos demais objetos é sugerido que sejam utilizados os algoritmos k-means ou k-medoids. Caso a aplicação se refira a dados que são imagens ou mapas, é aconselhado o uso de algoritmos baseados em densidade.

- Qualidade e velocidade: para qualquer aplicação, a velocidade de execução e a qualidade dos *clusters* gerados, são os critérios mais importantes na escolha do algoritmo. Geralmente algoritmos que geram grupos de boa qualidade, não são capazes de analisar bases de dados grandes. Neste caso é aconselhável que seja executado o algoritmo sobre a base de dados já condensada, mas sem provocar a perda de informações importantes para a geração dos *clusters*.
- Características dos dados: as características dos dados a serem agrupados devem ser consideradas.
- Domínio dos atributos: a maioria dos algoritmos de agrupamento tem mais facilidade de execução com valores numéricos para os atributos dos objetos.
- Ruído: os algoritmos são muito sensíveis a ruído nos dados, que afeta na qualidade dos *clusters* gerados.

2.1.3 Ferramenta de Mineração de Dados: Weka

Weka é abreviação para *Waikato Environment for Knowledge Learning* (em português, ambiente Waikato para aprendizagem/obtenção de conhecimento). É um software desenvolvido na Universidade de Waikato na Nova Zelândia, que em seu começo tinha como propósito na identificação de informação em dados gerados no ramo da agricultura (WEKA, 2013b).

> A mineração de dados não é o domínio exclusivo das grandes empresas e do software caro. Na realidade, há um software que faz quase todas as mesmas coisas que estes programas caros este software se chama WEKA (ABERNETHY, 2010).

A Weka é uma coleção de algoritmos de aprendizado para tarefas de mineração de dados. Os algoritmos podem ser tanto aplicados diretamente através da interface da Weka, ou também utilizados em um código Java particular. A Weka contém ferramentas preprocessamento, classificação, regressão, agrupamento, regras de associação e visualização (WEKA, 2013a).

A Weka é uma aplicação de código aberto, que está gratuitamente disponível através da licença GPL (*General Public Licence*). Inicialmente foi desenvolvida com a linguagem C,

e posteriormente foi reescrita na linguagem Java, com intuito de ser compatível com a maioria das plataformas computacionais (WEKA, 2013b).

- A ferramenta é amigável devido à sua interface gráfica, que proporciona uma configuração e operação rápidas. Permite que novos usuários consigam encontrar informações escondidas em suas bases de dados, devido as simples opções e interfaces visuais (WEKA, 2013b).

As suas principais características são (WEKA, 2013c):

- 49 ferramentas de preprocessamento de dados;
- 76 algoritmos de classificação e regressão;
- 8 algoritmos de agrupamento;
- 15 avaliadores de atributos/subconjuntos, e 10 algoritmos de seleção;
- 3 algoritmos para regras de associação;
- 3 interfaces gráficas:
  - Explorer: para análise exploratória de dados;
  - Experimenter: ambiente experimental;
  - KnowledgeFlow: interface com novo modelo de processo.

2.2 SERVIÇOS DE INTERNET NA REDE LOCAL

Esta seção tem como objetivo apresentar um breve conceito sobre alguns serviços relacionados com a Internet, que estão presentes na rede local.

2.2.1 Servidor Proxy

Um servidor *proxy* serve como intermediário entre a Internet e computadores da rede local (MORIMOTO, 2006).

> É um computador que funciona como intermediário entre um navegador da web (como o Internet Explorer) e a Internet. Os servidores *proxy* ajudam a melhorar o desempenho na web armazenando uma cópia das páginas da web utilizadas com mais frequência. Quando um navegador solicita uma página que está armazenada na coleção do servidor *proxy* (o cache), ela é disponibilizada pelo servidor *proxy*, o que é mais rápido do que acessar a web. Os servidores *proxy* também ajudam a melhorar a segurança porque filtram alguns tipos de conteúdo da web e softwares mal-intencionados. Eles são mais usados por redes de organizações e empresas. Normalmente, as pessoas que se conectam à Internet de suas casas não usam um servidor *proxy* (MICROSOFT, 2013).

Segundo Morimoto (2006), os principais objetivos de um *proxy*, em relação à rede local, são: compartilhar a conexão de Internet, melhorar o desempenho do uso da Internet, bloquear o acesso de determinadas páginas, e registrar todo o uso de Internet realizado.

Quando um servidor *proxy* compartilha a Internet com os computadores da rede local, ele passa a funcionar como uma roteador, encaminhando pacotes das requisições da rede local para a Internet e da Internet para a rede local. Assim a única fonte de contato dos computadores da rede interna com os serviços da web é através do servidor *proxy* (MORIMOTO, 2006).

O registro de todos os acessos realizados também é uma característica do servidor *proxy*. Conforme tenha processado qualquer requisição do cliente, o *proxy* armazena os registros os dados do acesso em arquivos de *log*, com informações como data, endereço IP, URL (*Uniform Resource Locator*) acessada, se o acesso foi liberado ou bloqueado, entre outras (MORIMOTO, 2006).

O servidor *proxy* tem a possibilidade de manter um *cache*, que se refere a um sistema de armazenamento temporário de páginas e arquivos recentemente acessados. Isso agiliza o acesso para os clientes da rede local, pois não é necessária que a requisição seja encaminhada para o servidor web na Internet, já que o conteúdo está armazenado no *cache* do servidor. Assim o servidor, ao verificar que possui os dados em memória, de maneira ágil, entrega-os ao cliente (MORIMOTO, 2006).

Com o uso do servidor *proxy* compartilhando a Internet, em muitos casos torna-se necessário realizar o controle do uso de Internet. Por isso, o *proxy* oferece a possibilidade de bloquear acessos indesejados, com configurações simples no servidor. O bloqueio é feito através do uso de ACL's (do inglês, *Access Control List*), e pode ser feito com base no horário, login, endereço IP, entre outras maneiras (MORIMOTO, 2006).

Segundo Morimoto (2006), o servidor *proxy* pode ser configurado para interagir com o computador da rede local de duas maneiras. Na primeira, é necessário que o computador seja

configurado manualmente com os dados do servidor *proxy*, e com isso surge a possibilidade de criar logins e senhas para os usuários. A outra maneira e também a mais utilizada, é o uso do "*proxy* transparente", em que não são necessárias configurações no cliente, e logins e senhas não podem ser utilizados.

2.2.1.1 Squid

Segundo Wessels (2004), o Squid é software de "*Web caching*" mais popular em uso.

Squid é um *proxy* totalmente caracterizado no HTTP/1.0 (*Hypertext Transfer Protocol*). Oferece um rico ambiente de controle de acessos, autorizações e de registros (*logs*) para criação de um *proxy* web, e aplicações servidoras de conteúdo. O Squid também oferece um alto desempenho de tráfego, por meio de um rico controle de otimizações (SQUID-CACHE.ORG, 2013).

O Squid é baseado no Harvest Cache Daemon, desenvolvido nos anos de 1990, sendo que tornou-se uma bifurcação do código base, juntamente com o Netapp's Netcache. Atualmente o projeto é desenvolvido de maneira voluntária, a fim de desenvolver as atuais e próximas tecnologias de *cache* de conteúdo (SQUID-CACHE.ORG, 2013).

2.2.2 Servidor Firewall

Um servidor *firewall* filtra o tráfego realizado entre redes distintas, forçando cada uma das redes a passar por políticas de controle de acesso. Em relação a questões de segurança, um *firewall* frequentemente defende uma rede interna de ataques oriundos de redes externas não confiáveis. O *firewall* garante que apenas tráfegos de dados autorizados atravessem de uma lado para o outro, entre as redes conectadas (ROECKL, 2004).

Segundo Grennan (2000), o *firewall* funciona com um processo de filtragem de pacotes, agindo no nível da rede. Conforme os pacotes chegam ao *firewall*, eles são filtrados conforme seu tipo, endereço de origem, endereço de destino, e porta, contidos em cada

pacote. Os *firewalls* são bastante transparentes ao usuário, sendo que este não precisa realizar configurações em seu computador ou aplicativos, para que seja possível acessar a Internet (GRENNAN, 2000).

Segundo Morimoto (2006), para se manter um servidor seguro é importante manter um *firewall* ativo na rede, com propósito de permitir apenas acessos aos serviços que realmente se deseja disponibilizar na rede interna.

Para Morimoto (2006), a ideia mais comum de *firewall* é de um dispositivo com que está ligado à Internet e também a rede local, mantendo-as inicialmente isoladas. Assim o *firewall*, aceita as conexões oriundas da rede local, e faz o roteamento para a Internet. Porém, as tentativas de conexão vindas da Internet são bloqueadas antes mesmo de chegar aos clientes.

2.2.3 Servidor de Páginas

Os servidores de páginas (ou servidores web) são a parte mais importante da Internet. São estes servidores que hospedam todas as páginas, mecanismos de buscas, e servem de base para diversos tipos de aplicativos web (MORIMOTO, 2006).

Segundo Souza (2013), um servidor web é o programa responsável pela publicação de documentos, imagens ou qualquer outro objeto que venha a ser acessado por um cliente através de um navegador, podendo estar na rede local ou na Internet.

2.2.3.1 Apache

O Apache é um dos servidores web mais antigos, seguro e com inúmeros módulos, que adicionam suporte aos mais exóticos recursos. Em 2006, quase 70% dos servidores web do mundo utilizavam o Apache (MORIMOTO, 2006).

O Apache é oriundo do Apache HTTP Server Project. Este projeto faz parte de um esforço colaborativo de desenvolvimento de software, de nível comercial, gratuitamente

disponível. O projeto é administrado em conjunto por um grupo de voluntários localizados em todo o mundo, que utilizam a Internet como meio de comunicação, planejamento, e desenvolvimento do servidor e sua documentação. Além disso, recebe contribuição de muitos usuários com ideias, documentação e codificação. O projeto faz parte do Apache Software Fundation, e teve início em fevereiro de 1995 (APACHE, 2013).

3. ESTUDO DE CASO

Neste capítulo é realizado o detalhamento do estudo de caso aplicado do processo de descoberta de conhecimento em bases de dados. As principais fases são destacadas em subcapítulos.

3.1 DEFINIÇÃO DE METAS

O primeiro passo realizado no processo, foi a entrevista com o administrador de rede da UTFPR Câmpus Medianeira. Neste primeiro momento, foi necessário saber quais eram as informações almejadas.

Após a entrevista, verificou-se que o ambiente/situação a ser analisada referia-se aos dados coletados e armazenados do uso da Internet de todo o Câmpus da instituição. Estes dados ficavam armazenados em meios digitais por longos períodos em máquinas servidoras, e eram gerados por meio de softwares de compartilhamento de Internet.

Segundo o administrador, o que lhe faltava era conseguir obter certeza de que as ferramentas que ele já usava, lhe traziam informações verdadeiras sobre o uso de Internet na instituição. Além desta confirmação, tinha como objetivo descobrir novas situações que tais ferramentas não lhe apresentava.

Assim, os seguintes questionamentos foram propostos pelo administrador:

- Qual a relação entre dias da semana, faixas de horários e os downloads realizados?
- Qual a relação entre dias da semana, faixas de horários e os acessos à páginas em geral?
- Qual a relação entre dias da semana, faixas de horários e os grupos de usuários?
- Qual a relação entre dias da semana, faixas de horários, grupos de usuários e os downloads realizados?
- Qual a relação entre dias da semana, faixas de horários, grupos de usuários e os acessos à páginas em geral?

- Qual a relação entre dias da semana, faixas de horários, grupos de usuários e o acesso aos sistemas da UTFPR?
- Qual a relação entre os grupos de usuários e as URL's acessadas?
- Quais os 20 IP's que respondem pelo maior percentual de uso da rede?
- Qual o protocolo de uso mais intenso para as redes 192.168.13.0/24 à 192.168.18/24?

3.1.1 Análise das questões

Analisando cada uma das questões, foi possível perceber que algumas delas não possuíam características que correspondessem à uma solução através de tarefas e algoritmos de mineração de dados, como: Associação, Agrupamento, Classificação e Regressão.

Os algoritmos das tarefas de mineração de dados são destinados a perceber padrões complexos em um conjunto de dados pouco simples. Por isso, as seguintes questões foram eliminadas das metas do projeto:

- Questão 1: quais os 20 IP's que respondem pelo maior percentual de uso da rede?
- Questão 2: qual o protocolo de uso mais intenso para as redes 192.168.13.0/24 à 192.168.18.0/24?

Para se entender melhor o motivo de cada uma das 3 questões terem sido removidas, observa-se a explicação a seguir.

- Questão 1: esta requer uma totalização do número de acessos juntamente com o tamanho de cada acesso, para cada IP da rede, restringindo os 20 mais utilizados. Nenhuma das tarefas de mineração faz somas em atributos de instâncias diferentes, e já neste ponto não torna-se viável minerar através da Weka em busca de sanar este questionamento.
- Questão 2: também com a mesma situação da questão 1, sendo necessário totalização por protocolo descobrindo a intensidade através da quantidade de acessos e do tamanho de cada acesso, e ainda uma restrição por faixas de IP.

Apesar destas questões terem sido removidas deste escopo, elas ainda podem ser respondidas, ou seja, pode-se descobrir seus padrões através da mineração de dados por outros

meios, como comandos SQL (*Structured Query Language)*. Isto envolveria a inserção dos dados extraídos e transformados para dentro de uma instância de uma banco de dados, e com simples comandos SQL seria possível responder plenamente os questionamentos. Os comandos SQL possuem instruções de soma, agrupamento e restrição que são suficientes para obter resultados satisfatórios quanto ao questionamento do administrador de redes.

3.1.2 Conclusão das questões alvo

Com as questões analisadas, foi necessário remover as que não atendiam os objetivos dos algoritmos das tarefas de mineração fornecidas pela Weka.

Quanto as demais questões, pode-se observar que existiam condições de serem respondidas através da aplicação do processo KDD no sistema automatizado a ser criado, sendo que este apresentaria os padrões encontrados, respondendo aos questionamentos do administrador de redes da UTFPR.

3.2 CRIAÇÃO DO SISTEMA AUTOMATIZADO

Nesta fase, foram iniciadas as pesquisas para o estabelecimento de bases para a construção do sistema automatizado que seria capaz de minerar os dados dos servidores, com finalidade de responder os questionamentos do administrador.

Logo no início das pesquisas foram encontradas algumas situações que viriam a tornar impossível a criação de tal sistema.

Inicialmente o sistema havia sido planejado para funcionar no seguinte âmbito de quatro partes:

- Primeira parte: o administrador de rede acessaria uma interface gráfica (web) e faria o *upload* dos arquivos de *log* dos servidores. Correspondendo à primeira etapa do KDD, a coleta dos dados.

- Segunda parte: o sistema deveria ser capaz de preprocessar e transformar os dados obtidos dos arquivos enviados pelo administrador, gerenciando automaticamente o processamento. Correspondendo à segunda e terceira etapas do KDD.
- Terceira parte: o sistema deveria aplicar os dados transformados, aos algoritmos de mineração de dados, na busca por padrões. Referente à quarta etapa do KDD.
- Quarta parte: sistema deveria apresentar ao administrador, relatórios com informações claras sobre os padrões encontrados, permitindo-lhe tomadas de decisão. Que corresponde à quinta e última etapa do KDD.

Apesar das tarefas do sistema haverem sido bem definidas, elas por si só, não foram capazes de expressar a viabilidade de suas construções. Ou seja, não foi possível entender a dificuldade de se realizar cada uma das etapas do sistema proposto, na fase do projeto deste trabalho.

A criação de um sistema web não é extremamente custosa em relação ao tempo. Mas o que demanda muito tempo, é a integração do processo KDD à um sistema web. E isso foi possível de perceber, durante esta análise logo após a definição das metas do projeto. A seguir serão explicadas as dificuldades encontradas em cada uma das partes do sistema que viria a ser desenvolvido.

Referente à primeira parte, esta contempla basicamente o *upload* dos arquivos brutos necessários durante todo o processo KDD. Um mecanismo de *upload* de arquivos para o sistema web não representa grandes dificuldades de implementação podendo afirmar-se que esta parte poderia ser construída dentro dos prazos.

Já referente à segunda parte, o preprocessamento e transformação, foi identificado como um procedimento difícil de ser colocado em prática num curto período de tempo. Primeiramente, seria necessário fazer com que o sistema pudesse reconhecer os vários padrões de *logs* de servidores, como: servidor *firewall*, servidor *proxy* e servidor de páginas. Além disso, ele deveria ser capaz de interpretar em cada registro dados incompletos, dados compostos, dados ilegíveis e dados inesperados, sendo responsável por fazer a transformação necessária para legibilidade do dado, reagindo automaticamente à estes eventos.

Outra opção para esta segunda parte, seria dar ao administrador de rede, a responsabilidade de reagir à estes dados conforme o sistema não soubesse agir. Porém isso foge da ideia de um sistema automatizado, repassando parte da efetividade do sistema para o administrador da rede.

Quanto à terceira parte, a mineração de dados, esta tinha como proposta a aplicação das tarefas de mineração (classificação, associação, agrupamento e regressão), também de uma maneira automatizada. Com isso, o administrador de rede deveria precisar escolher no máximo os atributos e o tipo do padrão a ser descoberto neles.

Tendo em vista que todo o processo do administrador de redes, deveria estar envolvido apenas com a interface gráfica do sistema web, foi encontrado provavelmente o maior empecilho. A ferramenta responsável pela aplicação das tarefas de mineração, seria a Weka, e não foi levado em conta no projeto, que esta ferramenta é independente, com sua própria entrada de dados, sua própria interface gráfica, seu próprio ambiente de manipulação dos algoritmos e sua própria maneira de apresentar os resultados. Neste momento ficou claro que não era possível utilizar as tarefas de mineração dentro do sistema web, através do acoplamento da Weka. E também não seria possível realizar parte do processo na ferramenta Weka e parte no sistema web, pois além de fugir da ideia do sistema proposto, aumentaria extremamente a complexidade no desenvolvimento e inviabilizaria o uso pelo usuário final.

Apesar deste grande empecilho, as pesquisas em torno de uma solução para esta parte ainda persistiram. Durante as pesquisas, verificou-se a existência de uma API (*Application Programming Interface*) Java da própria Weka. Essa API seria capaz de incorporar no sistema web as funcionalidades das tarefas de mineração.

O uso da API poderia vir a resolver a problemática, porém, percebeu-se que sua utilização seria uma tarefa complexa. Pois analisando o uso da API, notou-se que a mineração de dados torna-se uma tarefa menos abstraída do que em relação o uso da ferramenta Weka. Ou seja, passos que já estão inclusos na ferramenta Weka, requisitariam que com o uso da API fossem especificados manualmente, como por exemplo a leitura do arquivo de entrada. E assim tomou-se por conclusão que não seria viável a utilização da API para a aplicação no projeto.

A API da Weka não é complexa para a mineração de dados em si, mas sim para a aplicação neste projeto, já que a demanda do aprendizado do uso da própria API não foi levado em conta no projeto, e assim viria a utilizar do tempo de que não havia, estourando os prazos.

Quanto a quarta parte dos sistema, a apresentação dos resultados, não haveriam dificuldades para a construção.

No final desta análise, foi possível perceber que era inviável a execução da construção de um sistema automatizado para o processo KDD, nas limitações do projeto atual. E como conclusão, foi decidido ajustar as metas do projeto. A construção do sistema foi eliminada do escopo, sendo definido que o processo KDD seria executado, mas agora tendo em foco conseguir responder os questionamentos do administrador de redes com apenas:

- Criação de uma ferramenta simples e independente para o preprocessamento dos dados;
- Utilização da ferramenta Weka para a mineração dos dados preprocessados.

3.3 AQUISIÇÃO DE DADOS

Com a definição das metas e do novo escopo, o próximo passo refere-se à aquisição dos dados.

Conforme as questões propostas pelo administrador, a aquisição dos dados se limitou ao servidor *proxy*, que detém as informações necessárias para responder o questionamento. Os servidores *firewall* e de páginas, para esta situação, não possuem dados relevantes para a descoberta de padrões. No entanto, poderiam perfeitamente terem seus *logs* adquiridos para uma mineração que exigisse informações específicas de seus serviços.

Os dados alvo do servidor *proxy* estão contidos em arquivos de *logs*, e são gerados pelo Squid e persistidos em formato de arquivo de texto nos próprios servidores. A cada sete dias, os arquivos são removidos do servidor e são arquivados, devido ao seu tamanho atingir a unidade dos Gigabytes.

Para o estudo foram adquiridos 15 arquivos de *logs*, que haviam sido arquivados. Referem-se cada um a 7 dias, num período de 21 de outubro de 2012 até 03 de fevereiro de 2013. O tamanho médio dos arquivos é de 940 Megabytes.

3.4 EXTRAÇÃO DOS DADOS

O objetivo desta fase no processo, é gerar um arquivo de entrada para o sistema Weka, de maneira que este possa ler os dados e lhe seja interpretável, a fim de conseguir realizar a mineração.

3.4.1 Arquivo de entrada da Weka

Para gerar o arquivo de entrada é necessário conhecê-lo. A Weka aceita como entrada, mais de um formato de arquivo. Porém os de uso mais comum são: ARFF (*Attribute-Relation File Format*) e o XRFF (*Extensible Attribute-Relation File Format*).

- ARFF: é um arquivo de texto ASCII, que descreve uma lista de instâncias referente a um conjunto de atributos definidos;
- XRFF: é um arquivo no formato XML (*Extensible Markup Language*) estendido do arquivo ARFF. O arquivo XRFF produz um tamanho em disco proporcionalmente maior do que o arquivo ARFF (WEKA, 2013d).

Como é estimada a criação de uma quantidade muito grande de registros, torna-se importante optar por um arquivo que venha ter menor tamanho, e por isso optou-se pelo uso do arquivo ARFF.

Para que seja possível gerar o arquivo ARFF legível, é necessário conhecer sua estrutura. A estrutura do arquivo ARFF se estabelece em 2 partes: cabeçalho e dados.

O cabeçalho do arquivo contém um nome para a relação dos dados, anotado com “@RELATION”. Em seguida contém a lista dos atributos anotados com “@ATTRIBUTE”, que podem ser dos tipos: *numeric*, *integer*, *real*, *nominal* (requer a declaração dos possíveis valores, conhecidos como classes), *string* e *date* (requer a especificação do formato). A Listagem 1 mostra um exemplo do cabeçalho do arquivo ARFF.

Listagem 1: Cabeçalho do arquivo ARFF

```
@RELATION exemplo

@ATTRIBUTE codigo NUMERIC
@ATTRIBUTE nome STRING
@ATTRIBUTE data DATE "yyyy-MM-dd HH:mm:ss"
@ATTRIBUTE tipo {A, B, C}
```

A parte do arquivo que se refere aos dados, inicia-se com a anotação “@DATA”. Em seguida, cada nova linha representa uma instância. A instância contém os valores correspondentes aos atributos declarados no cabeçalho. A Listagem 2 mostra um exemplo da parte dos dados do arquivo.

Listagem 2: Dados do arquivo ARFF

```
@DATA
324, 'Gustavo', "2004-05-04 19:12:11", B
739, 'Jaime, "2001-02-09 02:10:02", A
421, 'Teobaldo', "2009-12-01 17:42:32", B
```

A Listagem 3 mostra um exemplo completo de um arquivo ARFF, correspondendo a estrutura final necessária para a entrada na ferramenta Weka.

Listagem 3: Arquivo ARFF completo

```
@RELATION exemplo

@ATTRIBUTE codigo NUMERIC
@ATTRIBUTE nome STRING
@ATTRIBUTE data DATE "yyyy-MM-dd HH:mm:ss"
@ATTRIBUTE tipo {A, B, C}

@DATA
324, 'Gustavo', "2004-05-04 19:12:11", B
739, 'Jaime, "2001-02-09 02:10:02", A
421, 'Teobaldo', "2009-12-01 17:42:32", B
```

3.4.2 Arquivo de log bruto

Além de conhecer o arquivo de entrada da Weka, é necessário também conhecer o arquivo de origem: o arquivo de *log* bruto. Após a coleta, este arquivo não passou por qualquer tipo de processamento, e por isso chamado de bruto.

Este arquivo de *log* gerado pelo Squid, possui um padrão assim como o ARFF, mas um tanto mais simplificado. Consiste em um conjunto de linhas em que cada uma corresponde a um registro de uma requisição de clientes da rede interna para acesso à Internet. Todo acesso à Internet, feito por qualquer computador da UTFPR Câmpus Medianeira, é registrado pelo Squid neste arquivo de *log*. A Figura 3 apresenta parte de um arquivo de *log* bruto para exemplo.

```
1359267363.476  55750 192.168.7.210 TCP_MISS/200 287 GET http://notify37.dropbox.com/subscribe? - DIRECT/108.160.160.165 text/plain
1359267366.820    114 192.168.100.38 TCP_MISS/200 6199 CONNECT osce105.icrc.trendmicro.com:443 - DIRECT/23.44.84.42 -
1359267374.320    114 192.168.100.38 TCP_MISS/200 6199 CONNECT osce105.icrc.trendmicro.com:443 - DIRECT/23.44.84.42 -
1359267381.820    114 192.168.100.38 TCP_MISS/200 6199 CONNECT osce105.icrc.trendmicro.com:443 - DIRECT/23.44.84.42 -
1359267389.320    115 192.168.100.38 TCP_MISS/200 6199 CONNECT osce105.icrc.trendmicro.com:443 - DIRECT/23.44.84.42 -
1359267391.110     33 192.168.7.210 TCP_REFRESH_HIT/304 261 GET http://ead.utfpr.edu.br/moodle/lib/speller/spellChecker.js - DIRECT/200.19.73.196 -
1359267391.115     37 192.168.7.210 TCP_REFRESH_HIT/304 261 GET http://ead.utfpr.edu.br/moodle/lib/javascript-static.js - DIRECT/200.19.73.196 -
1359267391.122    312 192.168.7.210 TCP_MISS/200 58321 GET http://ead.utfpr.edu.br/moodle/message/index.php - DIRECT/200.19.73.196 text/html
1359267391.131     20 192.168.7.210 TCP_REFRESH_HIT/304 260 GET http://ead.utfpr.edu.br/moodle/lib/cookies.js - DIRECT/200.19.73.196 -
```

Figura 3: Exemplo de arquivo de log bruto.
Fonte: autoria própria.

Cada um dos registros, possui 10 atributos separados por um espaço em branco, contendo informações detalhadas do acesso à Internet. Os atributos, na mesma ordem do arquivo, são:

- Data do acesso: corresponde a data e hora em formato Unix do exato momento do acesso;
- Duração do acesso: corresponde em milissegundos, quanto tempo o servidor levou desde o recebimento da requisição de acesso até a total entrega da resposta ao cliente;
- Endereço do cliente: corresponde ao endereço IP do cliente;
- Códigos de resultado: corresponde ao código de resultado da transação do acesso. É composto por dois campos: campo numérico e campo descritivo. Este atributo indica se houve sucesso ou não no acesso;
- Tamanho: corresponde em bytes, a quantidade de dados entregues ao cliente;
- Método de requisição: refere-se ao método de requisição utilizado para obter o objeto;

- URL: refere-se à URL da requisição;
- Rfc931: corresponde ao Ident Lookup[1];
- Código de hierarquia: esta informação pode ser composta por 3 itens: *tag* de hierarquia, código que define como a requisição foi tratada em casos específicos e IP/*hostname* de onde a requisição foi encaminhada;
- Tipo: refere-se ao tipo de conteúdo encontrado no objeto de resposta de um cabeçalho HTTP.

3.4.3 Definição dos atributos

Observando o questionamento realizado pelo administrador de redes, foi definido como atributos necessários para a mineração: dia da semana, hora inteira do dia, *download*, grupo de usuários e URL.

Detalhamento dos atributos:

- Dia da semana (abreviado para “diaSemana” no arquivo ARFF): referindo-se ao dia da semana do exato momento do acesso. O formato original do atributo é *Unix Timestamp*. Assim torna-se necessário convertê-lo;
  - Exemplo do formato original: 1359380000.583;
  - Exemplo da conversão: 2 (os dias identificados como Domingo igual a “1”, Segunda-feira igual a “2”, Terça-feira igual a “3”, Quarta-feira igual a “4”, Quinta-feira igual a “5”, Sexta-feira igual a “6” e Sábado igual a “7”).
- Hora inteira do dia (abreviado para “horaCheia” no arquivo ARFF): considera-se a mesma situação do atributo “Dia da semana”;
  - Exemplo do formato original: 1359380000.583;
  - Exemplo da conversão: 13 (interpretando-se 13 horas).
- Download: como visto no arquivo de *log* bruto, não existe um atributo que indique se o registro refere-se a um *download*. Desta maneira, é necessário estabelecer os

1 O campo Ident lookup (RFC931), refere-se ao valor do nome de usuário atrelado ao *socket* TCP, que é fornecido por um serviço que opera na máquina do cliente. É utilizado para propósitos de controle de acesso e registro adicional em *logs* (WIKI.SQUID-CACHE.ORG, 2012).

critérios para que no preprocessamento seja possível definir se o registro refere-se à acesso de *download* ou acesso comum. Os critérios utilizados para definir um acesso como *download* foram:

- Se o atributo “tamanho” for maior ou igual a 1.000.000 de bytes;
- Se o atributo “tipo” for igual a “application” e o atributo “tamanho” for maior ou igual a 100.000 bytes;
- Se o atributo “tipo” for igual a “video” ou “audio” ou “octet-stream”.

- Grupo de usuários (abreviado para “ipGroup” no arquivo ARFF): este também é um atributo que não encontra-se no arquivo bruto de *logs*. No entanto, pode ser calculado observando-se as especificações e o atributo “endereço do cliente”. Conforme a especificação dada pelo administrador, existem 3 grupos de usuários de interesse: alunos, servidores (professores e técnicos administrativos) e *wireless*. Os grupos de usuários são classificados de acordo com as seguintes (sub)redes:
  - Wireless: 172.27.0.0/16;
  - Servidores: 192.168.0.0/24 até 192.168.7.0/24;
  - Alunos: 192.168.8.0/24, 192.168.13.0/24 até 192.168.18.0/24, e 192.168.24.0/24 até 192.168.26.0/24.
- URL: apesar deste campo ser encontrado no arquivo de *logs*, ele requer transformação a fim de atender os objetivos. Exemplo de uma URL original: “http://osce10-6-en.url.trendmicro.com/T/728/3txB3Gw”. A parte útil da URL, para a mineração deste projeto, é parte do domínio. No entanto, esta parte útil pode variar conforme a URL, e por isso foram definidos critérios:
  - O caminho e o recursos da URL foram descartados;
  - O protocolo e a porta da URL, quando presentes, foram removidos;
  - Domínio: caso o último período for igual a “.com” ou “.net” ou “.org”, serão considerados apenas os dois últimos períodos, por exemplo “trendmicro.com”;
  - Devido ao nível de interesse os domínios com a raiz igual a “utfpr.edu.br”, estes serão mantidos completos, por exemplo: “ead.utfpr.edu.br”.

3.4.4 Criação de uma ferramenta de extração e transformação

Após a definição dos atributos, passou a ser possível iniciar o processo de extração e transformação dos dados brutos.

Porém, este processo não deve ser realizado de uma maneira manual. Cada arquivo de *log* pode atingir cerca de 13 milhões de linhas totalizando em disco quase 2 Gigabytes. Neste caso os processadores e editores de texto comuns não conseguem abrir estes arquivos. Além disso, não é uma tarefa fácil encontrar uma ferramenta de preprocessamento dos dados, muito menos para a aplicação em uma situação tão específica, que envolve dezenas de interações com os atributos do arquivo. Assim, ficou clara a necessidade da criação de alguma ferramenta para manipulação específica dos *logs.*

A missão da ferramenta tornou-se bastante específica: ler os arquivos de *log*, processar os atributos conforme a definição e gerar o arquivo ARFF interpretável pela Weka. Não é necessária uma interface gráfica para interagir com o processamento. Além disso, o processamento não seria feito por um usuário final (administrador de redes), mas sim pelo próprio minerador, que constantemente realizaria ajustes no código-fonte da ferramenta para atender ao objetivo. Para tal tarefa, optou-se pela criação de uma ferramenta com a linguagem de programação Java. O código-fonte base da ferramenta é encontrado no APÊNDICE A.

Durante a fase de criação da ferramenta, foi possível perceber que o processamento dos arquivos de *log* não é algo trivial. Foram necessárias muitas tentativas, até chegar ao modelo base final. Cada tentativa de gerar o arquivo correto, trata-se de um processo custoso, e definido assim devido a alguns fatores: capacidade do *hardware* de suportar o processamento, tempo gasto em cada processamento e tempo gasto para identificar problemas. Estes fatores são explicados a seguir.

As primeiras execuções da ferramenta, foram realizadas em um *hardware* comum. Tratava-se de um notebook em ambiente Linux 64bits, com processador Intel Core i5, 4 Gigabytes de memória RAM (*Random-access memory*) e disco rígido com 7200 RPM (Rotações Por Minuto). A execução do processamento utilizava 100% da capacidade do processador, e durava aproximadamente 3 minutos, para um único arquivo de *log* dentre os 15 disponíveis. Após alguns ajustes no código-fonte com objetivo de atender necessidades, e com o aumento do número de arquivos de *logs* em processamento de um para três, o tempo para a

execução completa da ferramenta passou para cerca de 7 minutos. Nesta última situação, houve superaquecimento do processador do computador, causando o desligamento emergencial do *hardware*. A partir deste ponto, com o *hardware* do momento, não estava sendo possível realizar o processamento.

Além do superaquecimento, a quantia de 4 Gigabytes de memória não eram suficientes, pois por diversas vezes tornou-se necessário abrir os arquivos (com ferramentas especiais) de *log* brutos ou os arquivos ARFF gerados, com intuito de analisar eventuais erros no processamento ou na leitura através da Weka. O computador utilizava toda a memória RAM e ainda parte da memória de Swap, inviabilizando qualquer outra atividade no computador. Além de não conseguir finalizar o processamento, inutilizava os demais aplicativos.

Neste ponto, ficou clara a necessidade de substituir o *hardware* que hospedaria o preprocessamento, para que se tornasse possível processar e também diminuir o tempo gasto com a execução dos testes.

Um novo *hardware* passou a ser utilizado: com sistema operacional Windows 7 64bits, com 2 processadores Intel Xeon X5660, 8 Gigabytes de memória RAM e discos em modo RAID (*Redundant Array of Independent Disks*). Com essa nova configuração, o tempo de processamento dos três arquivos foi reduzido para cerca de 4 minutos. Além disso, devido a quantidade de memória disponível, tornava-se possível abrir os arquivos de *log* bruto e ARFF para análise com facilidade, permitindo ajustes no código-fonte de uma maneira mais ágil. E o problema do superaquecimento deixou de existir.

Após o ajuste dos impedimentos citados, a ferramenta ganhou capacidade para gerar arquivos totalmente legíveis para a Weka. A Figura 4 mostra um exemplo de parte de um arquivo ARFF gerado para mineração de teste.

```
@RELATION LOGSQUID

@ATTRIBUTE diaSemana {1,2,3,4,5,6,7}
@ATTRIBUTE horaCheia {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23}
@ATTRIBUTE ipGroup {aluno, servidor, wireless}
@ATTRIBUTE uri {"dropbox.com", "ead.utfpr.edu.br"}
@ATTRIBUTE download {comum, download}

@DATA
1,6,servidor,dropbox.com,comum
1,6,servidor,ead.utfpr.edu.br,comum
1,6,servidor,ead.utfpr.edu.br,comum
```

Figura 4: Exemplo arquivo ARFF gerado.
Fonte: autoria própria.

3.4.4.1 Arquivos ARFF gerados

Com a ferramenta gerando arquivos ARFF legíveis, havia chegado o momento de criar os arquivos finais para uso da mineração.

Para responder os questionamentos do administrador de redes, um único arquivo não teria a capacidade de proporcionar a mineração correta. Algumas questões exigiam a disposição diferenciada para alguns atributos do arquivo ARFF. Então, inicialmente foram gerados 3 arquivos ARFF para suprir as necessidades das 7 questões. Basicamente foram divididos conforme algumas exigências:

- Algumas questões exigiam que o atributo "*download*" possuísse como valor exclusivo o valor "*download*". O arquivo também iria conter atributos "dia da semana", "hora cheia" e "grupo de usuários";
- Um outro arquivo seria criado, contrariando o primeiro quanto ao atributo "*download*", tendo como valor exclusivo o valor "comum". Também iria conter os atributos "dia da semana", "hora cheia" e "grupo de usuários";
- O terceiro e último arquivo traria uma restrição no atributo URL, trazendo em seu valor os domínios de raiz "utfpr.edu.br". Ou seja, somente domínios referentes a instituição. Também iria conter os atributos "dia da semana", "hora cheia" e "grupo de usuários".

3.4.5 Testes da Weka e ajustes nos arquivos ARFF

Antes de aplicar os algoritmos de mineração nos arquivos ARFF, testes foram realizados para identificar o comportamento da ferramenta Weka quanto ao tamanho do arquivo ARFF.

Os 15 arquivos de *log*, se processados todos juntos, geravam um arquivo ARFF com mais de 70 milhões de instâncias, cada instância com cerca de 5 atributos. A dedução era de que a Weka não fosse capaz de trabalhar com um arquivo tão grande, pelo menos nas condições conhecidas.

Os primeiros testes foram realizados com um arquivo ARFF de apenas 30.628.000 instâncias, 5 atributos cada instância, correspondendo a 9 semanas do período dos *logs* brutos (equivalente a 9 arquivos de *log*), produzindo um tamanho em disco de 1.081 Megabytes. A Figura 5 mostra o arquivo gerado, indicado pelas setas vermelhas.

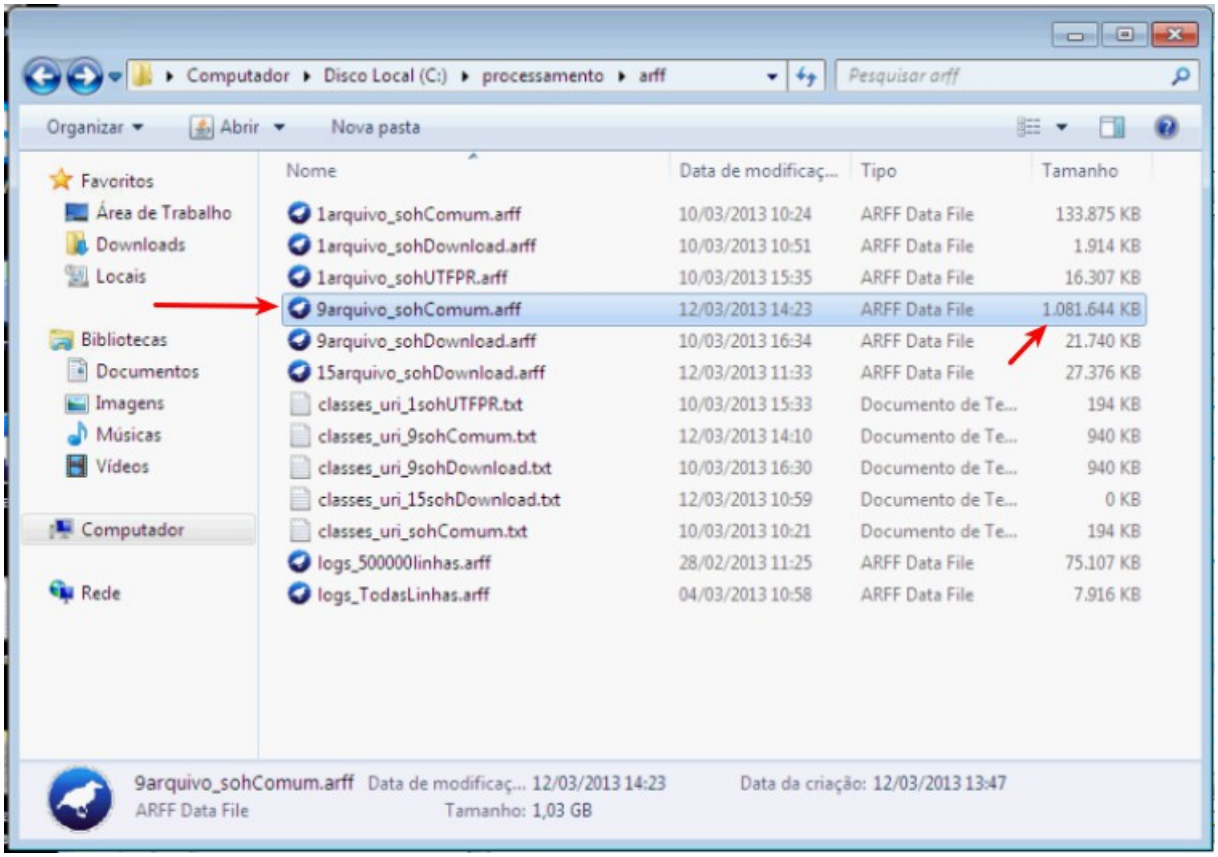

Figura 5: Tamanho do arquivo ARFF.
Fonte: autoria própria.

Durante estes testes iniciais, apenas no processo de leitura do arquivo ARFF, a Weka demorou cerca de 30 minutos para apresentar, na interface gráfica, que havia identificado o número de instâncias e atributos do arquivo. Ocupou, por quase todo o tempo 6 Gigabytes de memória, e reduziu para 5 Gigabytes após cerca de 20 minutos de processamento. Mesmo aguardando por mais de 8 horas, a interface da Weka permaneceu inerte, com a aparência de um sistema "travado", bloqueando qualquer ação do usuário ainda na tela de preprocessamento. Não era sequer possível tentar aplicar algum algoritmo das tarefas de mineração. A Figura 6 indica a situação citada, com pontos importantes destacados pelas setas vermelhas.

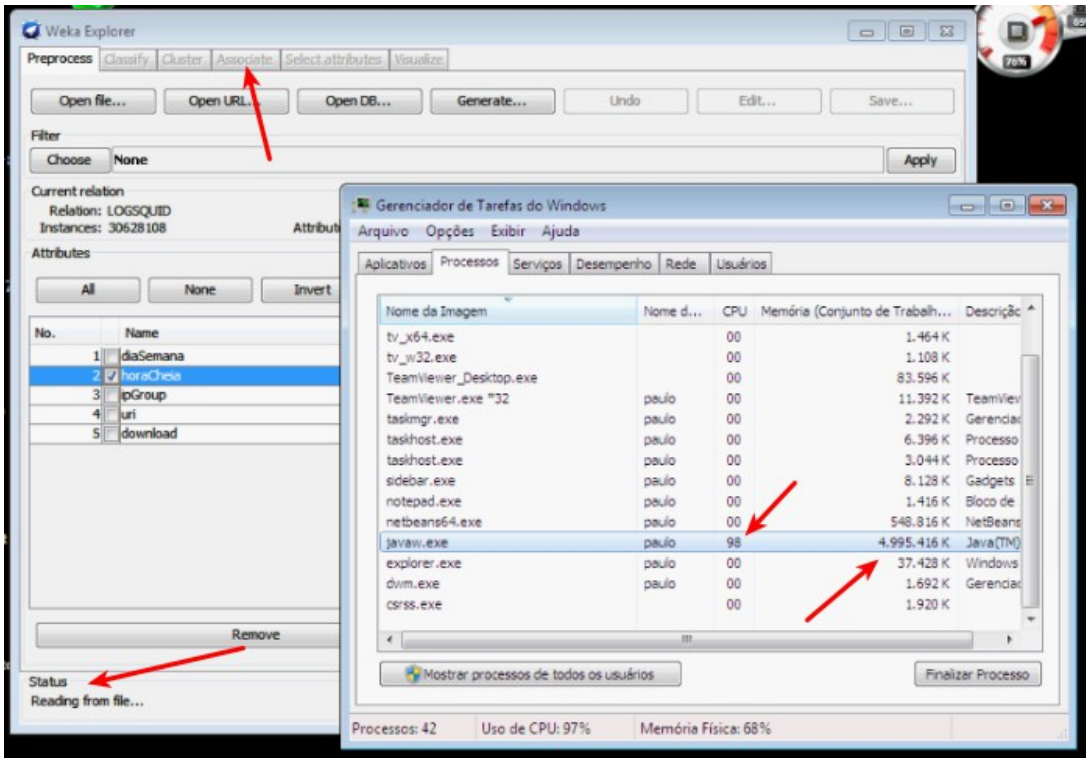

Figura 6: Weka travada.
Fonte: autoria própria.

Com a impossibilidade de processar um arquivo de tal tamanho, pesquisas na Internet foram realizadas com objetivo de encontrar métodos de processar arquivos ARFF grandes pela Weka. No entanto, após muitas horas investidas nas pesquisas, que foram feitas em fóruns e *wikis* da destinados à própria Weka, e também em materiais diversos, os únicos fatos encontrados foram:

- Seria necessário configurar a ferramenta Weka para que utilizasse mais memória do que o padrão. Porém tal configuração já havia sido feita, entregando à Weka 6

Gigabytes de memória, sendo que em alguns casos ela nem chegava a utilizá-la totalmente.

- E o outro fato interessante é que haviam muitos relatos de pessoas com o mesmo problema, e todos sem uma solução apresentada. Inclusive, haviam relatos dos mesmos problemas de “travamento” da Weka, que duravam um período de tempo indefinido, em servidores com 10 vezes mais capacidade de processamento do que aquele em estava sendo trabalhado no atual projeto.

Neste ponto, devido a falta de informações concretas sobre o problema, foi decidido que o arquivo ARFF deveria ser reduzido a fim de tentar propiciar sucesso no processamento. Outros testes foram realizados com um arquivo ARFF de apenas 8.267.788 instâncias (menos de um terço do primeiro teste), produzindo um tamanho em disco de 290 Megabytes. Este arquivo agora correspondia ao período de apenas 1 semana de *logs*. Apesar deste arquivo ser muito menor que o primeiro, o resultado do teste apresentou o mesmo problema.

Mais um teste foi realizado, reduzindo a quantidade de instâncias para 2.918.452, com tamanho em disco de 104 Megabytes. O novo arquivo referia-se a um período de *logs* gerados em uma semana de férias acadêmias da instituição, que poderia comprometer a validade dos padrões encontrados durante a mineração. Com intuito de apenas de saber se era possível minerar um arquivo daquele tamanho, o teste foi executado, e como resultado o problema foi identificado novamente.

Com a impossibilidade de minerar os arquivos testados, concluiu-se que não era possível processar na Weka o período de 1 semana de *logs*, e quanto menos os 15 arquivos de *logs* fornecidos pelo administrador de redes. Seria necessário que fosse feito um estudo aprofundado sobre a ferramenta Weka, revelando um possível método para minerar arquivos ARFF com tamanhos muito grandes.

Mas sem a possibilidade de realizar o estudo aprofundado sobre a mineração de arquivos grandes na Weka, e também com a necessidade inquestionável da mineração dos arquivos na finalidade de atender os objetivos do projeto, iniciou-se a busca por uma solução diferenciada. A quantidade de instâncias não parecia ser o problema, mas sim a quantidade de atributos e dos valores dos atributos presentes no arquivo ARFF.

Dois métodos dedutivos foram aplicados na ferramenta de preprocessamento criada, com intuito de melhorar o arquivo ARFF:

- Criar um arquivo ARFF para cada questão. Isso reduziria drasticamente a quantidade de dados desnecessários para a mineração de uma questão, já que envolveria a redução do número de atributos. No processo anterior, os atributos desnecessários seriam removidos, mas através da própria Weka, em sua tela de preprocessamento. Porém, a ferramenta não permitiu tal operação. Os arquivos ficaram dispostos desta maneira:
  - Arquivo da questão 1: contemplando apenas os atributos “dia da semana”, “hora cheia” e o atributo “*download*” restrito ao valor “*download*”;
  - Arquivo da questão 2: contemplando apenas os atributos “dia da semana”, “hora cheia” e o atributo “*download*” restrito ao valor “comum”;
  - Arquivo da questão 3: contemplando apenas os atributos “dia da semana”, “hora cheia” e “grupo de usuários”;
  - Arquivo da questão 4: contemplando apenas os atributos “dia da semana”, “hora cheia”, “grupo de usuários” e o atributo “*download*” restrito ao valor “*download*”;
  - Arquivo da questão 5: contemplando apenas os atributos “dia da semana”, “hora cheia”, “grupo de usuários” e o atributo “*download*” restrito ao valor “comum”;
  - Arquivo da questão 6: contemplando apenas os atributos “dia da semana”, “hora cheia”, “grupo de usuários” e o atributo “URL” restrito aos valores correspondentes à domínios com raiz “utfpr.edu.br”;
  - Arquivo da questão 7: contemplando apenas os atributos “grupo de usuários” e “URL”.
- O outro método levou em conta a remoção de classes do atributo “URL”. A não ser em relação à questão 6, nas demais situações a declaração deste atributo continha 15.018 classes. Essa quantidade muito grande poderia ser responsável por parte do travamento da ferramenta. Observando que as regras mineradas levariam em consideração apenas as classes mais importantes deste atributo, decidiu-se manter apenas as 50 classes mais utilizadas (que são as mais importantes). As demais classes além de não participarem de padrões importantes, reduziriam drasticamente o peso do processamento.

Enfim, esta última tentativa trouxe resultados positivos. Com a aplicação dos dois últimos métodos na ferramenta de preprocessamento criada, a Weka pôde pela primeira vez realizar o processamento do arquivo, aplicando os algoritmos e apresentando os resultados em tela. Assim tornou-se possível dar sequência no processo KDD.

3.5 APLICAÇÃO DO ALGORITMO DE ASSOCIAÇÃO PARA EXTRAÇÃO DE REGRAS

Nesta seção, ocorre a aplicação da etapa mais conhecida do processo Descoberta de Conhecimento em Bases de Dados, a Mineração de Dados. Os arquivos ARFF gerados são inseridos na ferramenta Weka e através da aplicação das tarefas de mineração necessárias, torna-se possível obter conhecimento que responda os questionamentos do administrador de redes. Cada um dos subcapítulos, a seguir, representa um dos questionamentos propostos, seguidos de uma breve interpretação do resultado minerado.

3.5.1 Questão 1: Qual a relação entre dias da semana, faixas de horários e os *downloads* realizados?

O arquivo utilizado na mineração desta questão, foi o arquivo com os atributos: “dia da semana”, “hora cheia”, atributo “*download*” contendo o valor “*download*”, “grupo de usuários” e “URL”. Porém os atributos “grupo de usuários” e “URL” foram removidos, durante o preprocessamento, já que não influenciam nos resultados.

Este arquivo ARFF gerado, corresponde ao período de 18 de novembro de 2012 até 25 de novembro de 2012, e com 121.104 instâncias.

Para encontrar um padrão da ocorrência de um atributo em relação a ocorrência de outro atributo, foi utilizada a Associação como tarefa de mineração de dados. E para a Associação foi utilizado o algoritmo Apriori.

Após algumas tentativas, os melhores valores encontrados para os parâmetros do algoritmo Apriori foram:

- Suporte: 1%;
- Confiança: 10%;
- Número máximo de regras: 25.

3.5.1.1 Interpretação da mineração da questão 1

A Listagem 4 apresenta o resultado da mineração de dados apresentado pela Weka para a questão 1:

Listagem 4: Regras de associação da questão 1

```
Best rules found:
 1. diaSemana=5 34358 ==> download=download 34358    conf:(1)
 2. diaSemana=2 20983 ==> download=download 20983    conf:(1)
 3. diaSemana=3 20878 ==> download=download 20878    conf:(1)
 4. diaSemana=4 19542 ==> download=download 19542    conf:(1)
 5. horaCheia=11 18556 ==> download=download 18556    conf:(1)
 6. diaSemana=6 17591 ==> download=download 17591    conf:(1)
 7. horaCheia=12 15538 ==> download=download 15538    conf:(1)
 8. horaCheia=18 11698 ==> download=download 11698    conf:(1)
 9. horaCheia=16 10628 ==> download=download 10628    conf:(1)
10. horaCheia=15 10560 ==> download=download 10560    conf:(1)
11. diaSemana=5 horaCheia=11 10293 ==> download=download 10293    conf:(1)
12. horaCheia=10 9442 ==> download=download 9442    conf:(1)
13. horaCheia=13 8474 ==> download=download 8474    conf:(1)
14. horaCheia=17 8140 ==> download=download 8140    conf:(1)
15. diaSemana=7 5807 ==> download=download 5807    conf:(1)
16. horaCheia=9 4935 ==> download=download 4935    conf:(1)
17. horaCheia=21 4893 ==> download=download 4893    conf:(1)
18. horaCheia=19 4078 ==> download=download 4078    conf:(1)
19. diaSemana=5 horaCheia=12 3889 ==> download=download 3889    conf:(1)
20. horaCheia=14 3756 ==> download=download 3756    conf:(1)
21. diaSemana=5 horaCheia=15 3737 ==> download=download 3737    conf:(1)
22. diaSemana=4 horaCheia=12 3456 ==> download=download 3456    conf:(1)
23. horaCheia=20 3311 ==> download=download 3311    conf:(1)
24. diaSemana=3 horaCheia=12 3247 ==> download=download 3247    conf:(1)
25. diaSemana=6 horaCheia=12 3148 ==> download=download 3148    conf:(1)
```

Observando-se o resultado entregue, pode-se no mínimo encontrar os seguintes padrões interessantes:

- Levando em consideração o dia da semana e o horário, a quinta-feira às 11h, é o intervalo de 1 hora com a maior concentração de *downloads* durante a semana. Os 3 próximos são: quinta-feira às 12h, quinta-feira às 15h e quarta-feira às 12h;
- Na sequência de mais *download* para menos *downloads* seguem os 8 horários com mais ocorrências: 11h, 12h, 18h, 16h, 15h, 10h 13h e 17h. Pode-se notar que o período da tarde é o momento com maiores ocorrências de *download* do que o período diurno e noturno, pois entre os primeiros 8 horários, 6 deles são vespertinos;
- A quinta-feira, é o dia em que mais ocorre *download* na instituição;
- Na sequência de mais *download* para menos *downloads* seguem os demais dias da semana: segunda-feira, terça-feira, quarta-feira, sexta-feira, sábado e domingo;
- Independente do dia da semana, o horário das 11 horas é o momento em que mais ocorrem *downloads*.

3.5.2 Questão 2: Qual a relação entre dias da semana, faixas de horários e os acessos a páginas em geral?

O arquivo utilizado na mineração desta questão, foi o arquivo com os atributos: “dia da semana”, “hora cheia”, atributo “*download*” contendo o valor “comum”, “grupo de usuários” e “URL”. Porém os atributos “grupo de usuários” e “URL” foram removidos, durante o preprocessamento, já que não influenciam nos resultados.

Este arquivo ARFF gerado, corresponde ao período de 18 de novembro de 2012 até 25 de novembro de 2012, e com 8.267.788 instâncias.

Para encontrar um padrão da ocorrência de um atributo em relação a ocorrência de outro atributo, foi utilizada a Associação como tarefa de mineração de dados. E para a Associação foi utilizado o algoritmo Apriori.

Após algumas tentativas, os melhores valores encontrados para os parâmetros do algoritmo Apriori foram:

- Suporte: 1%;
- Confiança: 90%;

- Número máximo de regras: 50.

3.5.2.1 Interpretação da mineração da questão 2

A Listagem 5 apresenta o resultado da mineração de dados apresentado pela Weka para a questão 2:

Listagem 5: Regras de associação da questão 2

```
Best rules found:

 1. diaSemana=3 1791496 ==> download=comum 1791496    conf:(1)
 2. diaSemana=2 1682476 ==> download=comum 1682476    conf:(1)
 3. diaSemana=5 1638800 ==> download=comum 1638800    conf:(1)
 4. diaSemana=6 1444713 ==> download=comum 1444713    conf:(1)
 5. diaSemana=4 1393772 ==> download=comum 1393772    conf:(1)
 6. horaCheia=18 961962 ==> download=comum 961962    conf:(1)
 7. horaCheia=12 857180 ==> download=comum 857180    conf:(1)
 8. horaCheia=16 844040 ==> download=comum 844040    conf:(1)
 9. horaCheia=13 780239 ==> download=comum 780239    conf:(1)
10. horaCheia=10 746236 ==> download=comum 746236    conf:(1)
11. horaCheia=11 725889 ==> download=comum 725889    conf:(1)
12. horaCheia=17 724167 ==> download=comum 724167    conf:(1)
13. horaCheia=15 694103 ==> download=comum 694103    conf:(1)
14. horaCheia=19 407366 ==> download=comum 407366    conf:(1)
15. horaCheia=14 294170 ==> download=comum 294170    conf:(1)
16. horaCheia=22 277527 ==> download=comum 277527    conf:(1)
17. diaSemana=7 262555 ==> download=comum 262555    conf:(1)
18. diaSemana=5 horaCheia=18 237125 ==> download=comum 237125    conf:(1)
19. horaCheia=21 219850 ==> download=comum 219850    conf:(1)
20. horaCheia=9 217194 ==> download=comum 217194    conf:(1)
21. horaCheia=20 216312 ==> download=comum 216312    conf:(1)
22. diaSemana=2 horaCheia=18 211755 ==> download=comum 211755    conf:(1)
23. diaSemana=3 horaCheia=12 195133 ==> download=comum 195133    conf:(1)
24. diaSemana=3 horaCheia=18 194434 ==> download=comum 194434    conf:(1)
25. diaSemana=6 horaCheia=12 190892 ==> download=comum 190892    conf:(1)
26. diaSemana=2 horaCheia=16 189874 ==> download=comum 189874    conf:(1)
27. diaSemana=3 horaCheia=16 189339 ==> download=comum 189339    conf:(1)
28. diaSemana=3 horaCheia=13 188852 ==> download=comum 188852    conf:(1)
29. horaCheia=23 185533 ==> download=comum 185533    conf:(1)
30. diaSemana=2 horaCheia=15 167100 ==> download=comum 167100    conf:(1)
31. diaSemana=3 horaCheia=17 165454 ==> download=comum 165454    conf:(1)
32. diaSemana=5 horaCheia=11 164174 ==> download=comum 164174    conf:(1)
33. diaSemana=6 horaCheia=13 161034 ==> download=comum 161034    conf:(1)
34. diaSemana=4 horaCheia=18 159884 ==> download=comum 159884    conf:(1)
35. diaSemana=4 horaCheia=16 155794 ==> download=comum 155794    conf:(1)
36. diaSemana=5 horaCheia=16 155221 ==> download=comum 155221    conf:(1)
37. diaSemana=3 horaCheia=10 154043 ==> download=comum 154043    conf:(1)
```

```
38. diaSemana=3 horaCheia=11 151913 ==> download=comum 151913    conf:(1)
39. diaSemana=6 horaCheia=10 147029 ==> download=comum 147029    conf:(1)
40. diaSemana=2 horaCheia=12 145505 ==> download=comum 145505    conf:(1)
41. diaSemana=5 horaCheia=17 144400 ==> download=comum 144400    conf:(1)
42. diaSemana=2 horaCheia=10 143432 ==> download=comum 143432    conf:(1)
43. diaSemana=5 horaCheia=10 142511 ==> download=comum 142511    conf:(1)
44. diaSemana=2 horaCheia=17 142294 ==> download=comum 142294    conf:(1)
45. diaSemana=4 horaCheia=12 142294 ==> download=comum 142294    conf:(1)
46. diaSemana=5 horaCheia=12 141726 ==> download=comum 141726    conf:(1)
47. diaSemana=3 horaCheia=15 139291 ==> download=comum 139291    conf:(1)
48. diaSemana=6 horaCheia=18 138354 ==> download=comum 138354    conf:(1)
49. diaSemana=4 horaCheia=13 135934 ==> download=comum 135934    conf:(1)
50. diaSemana=4 horaCheia=10 135210 ==> download=comum 135210    conf:(1)
```

Observando-se o resultado entregue, pode-se no mínimo encontrar os seguintes padrões interessantes:

- Entre as 18h00min e as 18h59min, é o período de 1 hora com o uso comum da Internet, considerando todos os dias da semana. Em seguida, as próximas oito são 12h, 16h, 13h, 10h, 11h, 17h, 15h e 19h, apontando que o período da tarde contém horários de mais uso do que os demais períodos;
- Levando em consideração o dia da semana e juntamente a hora do dia, é possível identificar que na quinta-feira às 18h é o período de 1 hora com mais uso comum da Internet. Em seguida, aparecem nas próximas 7 regras: segunda-feira às 18h, terça-feira às 12h, terça-feira às 18h, sexta-feira às 12h, segunda-feira às 16h, terça-feira às 16h e terça-feira às 13;
- A terça-feira é o dia da semana com o maior uso da Internet para fins comuns;
- Em ordem decrescente, os demais dias com os maiores uso da Internet para fins comuns são: segunda-feira, quinta-feira, sexta-feira, quarta-feira, sábado e domingo.

3.5.3 Questão 3: Qual a relação entre dias da semana, faixas de horários e os grupos de usuários?

O arquivo utilizado na mineração desta questão, foi o arquivo com os atributos: “dia da semana”, “hora cheia”, “*downloads*”, “grupo de usuários” e “URL”. Porém os atributos “*download*” e “URL” foram removidos, durante o pré-processamento, já que não influenciam nos resultados.

Este arquivo ARFF gerado, corresponde ao período de 18 de novembro de 2012 até 25 de novembro de 2012, e com 8.388.892 instâncias.

Para encontrar um padrão da ocorrência de um atributo em relação a ocorrência de outro atributo, foi utilizada a Associação como tarefa de mineração de dados. E para a Associação foi utilizado o algoritmo Apriori.

Após algumas tentativas, os melhores valores encontrados para os parâmetros do algoritmo Apriori foram:

- Suporte: 1%;
- Confiança: 10%;
- Número máximo de regras: 100.

3.5.3.1 Interpretação da mineração da questão 3

A Listagem 6 apresenta o resultado da mineração de dados apresentado pela Weka para a questão 3:

Listagem 6: Regras de associação da questão 3

```
Best rules found:

 1. diaSemana=5 horaCheia=18 239001 ==> ipGroup=servidor 200311    conf:(0.84)
 2. diaSemana=6 horaCheia=18 140145 ==> ipGroup=servidor 115480    conf:(0.82)
 3. diaSemana=6 horaCheia=11 133204 ==> ipGroup=servidor 108412    conf:(0.81)
 4. diaSemana=2 horaCheia=11 132367 ==> ipGroup=servidor 104713    conf:(0.79)
 5. diaSemana=2 horaCheia=10 146057 ==> ipGroup=servidor 115089    conf:(0.79)
 6. diaSemana=5 horaCheia=16 156899 ==> ipGroup=servidor 122350    conf:(0.78)
```

```
 7. diaSemana=5 horaCheia=17 145535 ==> ipGroup=servidor 112454    conf:(0.77)
 8. diaSemana=4 horaCheia=13 137266 ==> ipGroup=servidor 105158    conf:(0.77)
 9. diaSemana=6 horaCheia=12 194040 ==> ipGroup=servidor 146020    conf:(0.75)
10. diaSemana=4 horaCheia=12 145750 ==> ipGroup=servidor 108385    conf:(0.74)
11. diaSemana=4 horaCheia=17 126426 ==> ipGroup=servidor 93605    conf:(0.74)
12. diaSemana=2 horaCheia=15 169331 ==> ipGroup=servidor 125212    conf:(0.74)
13. diaSemana=6 horaCheia=13 162543 ==> ipGroup=servidor 117866    conf:(0.73)
14. horaCheia=19 411444 ==> ipGroup=servidor 298253    conf:(0.72)
15. diaSemana=6 horaCheia=10 148283 ==> ipGroup=servidor 107217    conf:(0.72)
16. diaSemana=6 horaCheia=16 126766 ==> ipGroup=servidor 90840    conf:(0.72)
17. diaSemana=3 horaCheia=13 191579 ==> ipGroup=servidor 137100    conf:(0.72)
18. diaSemana=6 1462304 ==> ipGroup=servidor 1039465    conf:(0.71)
19. diaSemana=2 horaCheia=13 132149 ==> ipGroup=servidor 93768    conf:(0.71)
20. horaCheia=13 788713 ==> ipGroup=servidor 553430    conf:(0.7)
21. diaSemana=2 horaCheia=12 146692 ==> ipGroup=servidor 102929    conf:(0.7)
22. horaCheia=14 297926 ==> ipGroup=servidor 208615    conf:(0.7)
23. diaSemana=5 horaCheia=13 127536 ==> ipGroup=servidor 89271    conf:(0.7)
24. diaSemana=5 horaCheia=10 143915 ==> ipGroup=servidor 100723    conf:(0.7)
25. diaSemana=3 horaCheia=16 191600 ==> ipGroup=servidor 133340    conf:(0.7)
26. diaSemana=5 1673158 ==> ipGroup=servidor 1161927    conf:(0.69)
27. horaCheia=12 872718 ==> ipGroup=servidor 605493    conf:(0.69)
28. diaSemana=3 horaCheia=12 198380 ==> ipGroup=servidor 137327    conf:(0.69)
29. horaCheia=20 219623 ==> ipGroup=servidor 151501    conf:(0.69)
30. diaSemana=2 1703459 ==> ipGroup=servidor 1162092    conf:(0.68)
31. horaCheia=10 755678 ==> ipGroup=servidor 514920    conf:(0.68)
32. diaSemana=3 horaCheia=11 153130 ==> ipGroup=servidor 103540    conf:(0.68)
33. horaCheia=11 744445 ==> ipGroup=servidor 501607    conf:(0.67)
34. diaSemana=5 horaCheia=12 145615 ==> ipGroup=servidor 97843    conf:(0.67)
35. horaCheia=17 732307 ==> ipGroup=servidor 491410    conf:(0.67)
36. horaCheia=18 973660 ==> ipGroup=servidor 652847    conf:(0.67)
37. diaSemana=3 horaCheia=10 155247 ==> ipGroup=servidor 103993    conf:(0.67)
38. horaCheia=16 854668 ==> ipGroup=servidor 562127    conf:(0.66)
39. diaSemana=7 268362 ==> ipGroup=aluno 175962    conf:(0.66)
40. diaSemana=4 1413314 ==> ipGroup=servidor 922579    conf:(0.65)
41. diaSemana=3 1812374 ==> ipGroup=servidor 1177641    conf:(0.65)
42. diaSemana=3 horaCheia=17 167127 ==> ipGroup=servidor 106013    conf:(0.63)
43. diaSemana=3 horaCheia=18 197002 ==> ipGroup=servidor 123526    conf:(0.63)
44. horaCheia=15 704663 ==> ipGroup=servidor 440988    conf:(0.63)
45. diaSemana=2 horaCheia=16 191895 ==> ipGroup=servidor 119804    conf:(0.62)
46. horaCheia=21 224743 ==> ipGroup=servidor 137707    conf:(0.61)
47. diaSemana=4 horaCheia=18 162828 ==> ipGroup=servidor 99758    conf:(0.61)
48. diaSemana=3 horaCheia=15 141128 ==> ipGroup=servidor 85561    conf:(0.61)
49. diaSemana=2 horaCheia=17 144604 ==> ipGroup=servidor 86482    conf:(0.6)
50. horaCheia=22 280153 ==> ipGroup=servidor 166995    conf:(0.6)
51. horaCheia=9 222129 ==> ipGroup=servidor 128221    conf:(0.58)
52. diaSemana=4 horaCheia=16 157287 ==> ipGroup=servidor 90456    conf:(0.58)
53. diaSemana=5 horaCheia=11 174467 ==> ipGroup=servidor 98392    conf:(0.56)
54. horaCheia=23 186862 ==> ipGroup=servidor 102195    conf:(0.55)
55. diaSemana=2 horaCheia=18 214054 ==> ipGroup=servidor 106780    conf:(0.5)
56. diaSemana=2 horaCheia=18 214054 ==> ipGroup=aluno 99566    conf:(0.47)
57. horaCheia=18 ipGroup=aluno 262757 ==> diaSemana=2 99566    conf:(0.38)
58. horaCheia=15 704663 ==> ipGroup=aluno 221778    conf:(0.31)
59. horaCheia=18 ipGroup=servidor 652847 ==> diaSemana=5 200311    conf:(0.31)
60. ipGroup=wireless 650994 ==> diaSemana=3 189371    conf:(0.29)
61. horaCheia=15 ipGroup=servidor 440988 ==> diaSemana=2 125212    conf:(0.28)
62. horaCheia=18 973660 ==> ipGroup=aluno 262757    conf:(0.27)
63. diaSemana=4 1413314 ==> ipGroup=aluno 378393    conf:(0.27)
```

```
 64. horaCheia=16 854668 ==> ipGroup=aluno 228052    conf:(0.27)
 65. horaCheia=10 755678 ==> ipGroup=aluno 198549    conf:(0.26)
 66. diaSemana=2 1703459 ==> ipGroup=aluno 446260    conf:(0.26)
 67. horaCheia=13 ipGroup=servidor 553430 ==> diaSemana=3 137100    conf:(0.25)
 68. horaCheia=19 411444 ==> diaSemana=5 101298    conf:(0.25)
 69. diaSemana=3 1812374 ==> ipGroup=aluno 445362    conf:(0.25)
 70. horaCheia=18 973660 ==> diaSemana=5 239001    conf:(0.25)
 71. horaCheia=12 872718 ==> ipGroup=aluno 213656    conf:(0.24)
 72. horaCheia=13 788713 ==> diaSemana=3 191579    conf:(0.24)
 73. horaCheia=11 744445 ==> ipGroup=aluno 179748    conf:(0.24)
 74. horaCheia=12 ipGroup=servidor 605493 ==> diaSemana=6 146020    conf:(0.24)
 75. horaCheia=15 704663 ==> diaSemana=2 169331    conf:(0.24)
 76. horaCheia=16 ipGroup=servidor 562127 ==> diaSemana=3 133340    conf:(0.24)
 77. horaCheia=17 732307 ==> ipGroup=aluno 172702    conf:(0.24)
 78. horaCheia=11 744445 ==> diaSemana=5 174467    conf:(0.23)
 79. horaCheia=13 788713 ==> ipGroup=aluno 184506    conf:(0.23)
 80. horaCheia=17 ipGroup=servidor 491410 ==> diaSemana=5 112454    conf:(0.23)
 81. horaCheia=17 732307 ==> diaSemana=3 167127    conf:(0.23)
 82. diaSemana=6 1462304 ==> ipGroup=aluno 332784    conf:(0.23)
 83. horaCheia=12 872718 ==> diaSemana=3 198380    conf:(0.23)
 84. horaCheia=12 ipGroup=servidor 605493 ==> diaSemana=3 137327    conf:(0.23)
 85. diaSemana=5 1673158 ==> ipGroup=aluno 375858    conf:(0.22)
 86. horaCheia=16 854668 ==> diaSemana=2 191895    conf:(0.22)
 87. horaCheia=16 854668 ==> diaSemana=3 191600    conf:(0.22)
 88. horaCheia=10 ipGroup=servidor 514920 ==> diaSemana=2 115089    conf:(0.22)
 89. diaSemana=2 ipGroup=aluno 446260 ==> horaCheia=18 99566    conf:(0.22)
 90. horaCheia=12 872718 ==> diaSemana=6 194040    conf:(0.22)
 91. horaCheia=18 973660 ==> diaSemana=2 214054    conf:(0.22)
 92. horaCheia=16 ipGroup=servidor 562127 ==> diaSemana=5 122350    conf:(0.22)
 93. horaCheia=11 ipGroup=servidor 501607 ==> diaSemana=6 108412    conf:(0.22)
 94. horaCheia=17 ipGroup=servidor 491410 ==> diaSemana=3 106013    conf:(0.22)
 95. horaCheia=19 411444 ==> diaSemana=2 88563    conf:(0.22)
 96. horaCheia=16 ipGroup=servidor 562127 ==> diaSemana=2 119804    conf:(0.21)
 97. horaCheia=13 ipGroup=servidor 553430 ==> diaSemana=6 117866    conf:(0.21)
 98. ipGroup=servidor 5576135 ==> diaSemana=3 1177641    conf:(0.21)
 99. horaCheia=11 ipGroup=servidor 501607 ==> diaSemana=2 104713    conf:(0.21)
100. ipGroup=servidor 5576135 ==> diaSemana=2 1162092    conf:(0.21)
```

Observando-se o resultado entregue, pode-se no mínimo encontrar os seguintes padrões interessantes:

- Independente do dia da semana, o grupo “alunos” tem os 4 horários em que mais tendem a utilizar de maneira geral a Internet, sendo às 15h (31%), 18h (27%), 16h (27%) e 10h (26%);
- Na quinta-feira às 18 horas, referente ao uso em geral de Internet, existe 84% de chance do usuário ser do grupo “servidor”. Os servidores têm as 5 outras mais fortes tendências em: sexta-feira às 18h (82%), sexta-feira às 11h (81%), segunda-feira às 11h (79%), segunda-feira às 10h (79%) e quinta-feira às 16h (78%);

- Independente do dia da semana, às 19h existe tendência de 72% do uso da Internet de maneira geral estar sendo realizado pelo grupo “servidor”. Isso ocorre também aos cinco próximos horários mais fortes para as 13h (70%), 14h (70%), 12h (69%), 20h (69%), 10h (68%) e 11h (67%);
- O grupo de usuários “aluno”, só aparecem a partir da trigésima nona regra. No sábado, a tendência do grupo “aluno” estar utilizando a Internet de maneira geral é de 66%;
- Na segunda-feira às 18h o grupo “aluno” corresponde a 47% do uso em geral;
- O grupo de usuários “wireless”, dentre as primeiras 100 regras encontradas, apresenta-se em apenas 1. A regra informa que o grupo corresponde a 29% do uso da Internet na terça-feira.

3.5.4 Questão 4: Qual a relação entre dias da semana, faixas de horários, grupos de usuários e os *downloads* realizados?

O arquivo utilizado na mineração desta questão, foi o arquivo com os atributos: “dia da semana”, “hora cheia”, atributo “*download*” contendo o valor “*download*”, “grupo de usuários” e “URL”. Porém o atributo “URL” foi removido durante o pré-processamento, já que não influencia nos resultados.

Este arquivo ARFF gerado, corresponde ao período de 18 de novembro de 2012 até 25 de novembro de 2012, e com 121.104 instâncias.

Para encontrar um padrão da ocorrência de um atributo em relação a ocorrência de outro atributo, foi utilizada a Associação como tarefa de mineração de dados. E para a Associação foi utilizado o algoritmo Apriori.

Após algumas tentativas, os melhores valores encontrados para os parâmetros do algoritmo Apriori foram:

- Suporte: 1%;
- Confiança: 90%;
- Número máximo de regras: 75.

3.5.4.1 Interpretação da mineração da questão 4

A Listagem 7 apresenta o resultado da mineração de dados apresentado pela Weka para a questão 4:

Listagem 7: Regras de associação da questão 4

```
Best rules found:

 1. ipGroup=servidor 70095 ==> download=download 70095    conf:(1)
 2. diaSemana=5 34358 ==> download=download 34358    conf:(1)
 3. ipGroup=aluno 33220 ==> download=download 33220    conf:(1)
 4. diaSemana=2 20983 ==> download=download 20983    conf:(1)
 5. diaSemana=3 20878 ==> download=download 20878    conf:(1)
 6. diaSemana=4 19542 ==> download=download 19542    conf:(1)
 7. horaCheia=11 18556 ==> download=download 18556    conf:(1)
 8. ipGroup=wireless 17789 ==> download=download 17789    conf:(1)
 9. diaSemana=6 17591 ==> download=download 17591    conf:(1)
10. horaCheia=12 15538 ==> download=download 15538    conf:(1)
11. diaSemana=5 ipGroup=servidor 14071 ==> download=download 14071    conf:(1)
12. diaSemana=5 ipGroup=aluno 13680 ==> download=download 13680    conf:(1)
13. diaSemana=2 ipGroup=servidor 13590 ==> download=download 13590    conf:(1)
14. diaSemana=4 ipGroup=servidor 12961 ==> download=download 12961    conf:(1)
15. diaSemana=6 ipGroup=servidor 12778 ==> download=download 12778    conf:(1)
16. diaSemana=3 ipGroup=servidor 11841 ==> download=download 11841    conf:(1)
17. horaCheia=18 11698 ==> download=download 11698    conf:(1)
18. horaCheia=16 10628 ==> download=download 10628    conf:(1)
19. horaCheia=15 10560 ==> download=download 10560    conf:(1)
20. diaSemana=5 horaCheia=11 10293 ==> download=download 10293    conf:(1)
21. horaCheia=12 ipGroup=servidor 9982 ==> download=download 9982    conf:(1)
22. horaCheia=11 ipGroup=aluno 9630 ==> download=download 9630    conf:(1)
23. horaCheia=10 9442 ==> download=download 9442    conf:(1)
24. diaSemana=5 horaCheia=11 ipGroup=aluno 8683 ==> download=download 8683    conf:(1)
25. horaCheia=13 8474 ==> download=download 8474    conf:(1)
26. horaCheia=17 8140 ==> download=download 8140    conf:(1)
27. horaCheia=11 ipGroup=servidor 6831 ==> download=download 6831    conf:(1)
28. diaSemana=5 ipGroup=wireless 6607 ==> download=download 6607    conf:(1)
29. horaCheia=10 ipGroup=servidor 6509 ==> download=download 6509    conf:(1)
30. diaSemana=3 ipGroup=aluno 6230 ==> download=download 6230    conf:(1)
31. horaCheia=16 ipGroup=servidor 6195 ==> download=download 6195    conf:(1)
32. horaCheia=18 ipGroup=servidor 6177 ==> download=download 6177    conf:(1)
33. horaCheia=13 ipGroup=servidor 6089 ==> download=download 6089    conf:(1)
34. diaSemana=7 5807 ==> download=download 5807    conf:(1)
35. horaCheia=17 ipGroup=servidor 5091 ==> download=download 5091    conf:(1)
36. horaCheia=9 4935 ==> download=download 4935    conf:(1)
37. horaCheia=21 4893 ==> download=download 4893    conf:(1)
38. diaSemana=2 ipGroup=aluno 4826 ==> download=download 4826    conf:(1)
39. horaCheia=18 ipGroup=aluno 4748 ==> download=download 4748    conf:(1)
40. horaCheia=15 ipGroup=servidor 4442 ==> download=download 4442    conf:(1)
41. horaCheia=12 ipGroup=aluno 4221 ==> download=download 4221    conf:(1)
42. horaCheia=19 4078 ==> download=download 4078    conf:(1)
43. diaSemana=4 ipGroup=aluno 3998 ==> download=download 3998    conf:(1)
```

```
44. horaCheia=21 ipGroup=servidor 3903 ==> download=download 3903    conf:(1)
45. diaSemana=5 horaCheia=12 3889 ==> download=download 3889    conf:(1)
46. horaCheia=14 3756 ==> download=download 3756    conf:(1)
47. diaSemana=5 horaCheia=15 3737 ==> download=download 3737    conf:(1)
48. horaCheia=15 ipGroup=wireless 3597 ==> download=download 3597    conf:(1)
49. diaSemana=4 horaCheia=12 3456 ==> download=download 3456    conf:(1)
50. horaCheia=20 3311 ==> download=download 3311    conf:(1)
51. diaSemana=3 horaCheia=12 3247 ==> download=download 3247    conf:(1)
52. diaSemana=6 horaCheia=12 3148 ==> download=download 3148    conf:(1)
53. diaSemana=4 horaCheia=12 ipGroup=servidor 3089 ==> download=download 3089    conf:(1)
54. horaCheia=19 ipGroup=servidor 3050 ==> download=download 3050    conf:(1)
55. diaSemana=5 horaCheia=15 ipGroup=wireless 3042 ==> download=download 3042    conf:(1)
56. diaSemana=5 horaCheia=21 3008 ==> download=download 3008    conf:(1)
57. horaCheia=16 ipGroup=aluno 2988 ==> download=download 2988    conf:(1)
58. diaSemana=7 ipGroup=servidor 2978 ==> download=download 2978    conf:(1)
59. horaCheia=9 ipGroup=servidor 2966 ==> download=download 2966    conf:(1)
60. diaSemana=4 horaCheia=18 2944 ==> download=download 2944    conf:(1)
61. diaSemana=3 ipGroup=wireless 2807 ==> download=download 2807    conf:(1)
62. diaSemana=4 horaCheia=11 2805 ==> download=download 2805    conf:(1)
63. diaSemana=5 horaCheia=21 ipGroup=servidor 2748 ==> download=download 2748    conf:(1)
64. diaSemana=3 horaCheia=13 2727 ==> download=download 2727    conf:(1)
65. diaSemana=6 horaCheia=12 ipGroup=servidor 2716 ==> download=download 2716    conf:(1)
66. diaSemana=5 horaCheia=12 ipGroup=aluno 2680 ==> download=download 2680    conf:(1)
67. horaCheia=22 2626 ==> download=download 2626    conf:(1)
68. diaSemana=2 horaCheia=10 2625 ==> download=download 2625    conf:(1)
69. diaSemana=4 ipGroup=wireless 2583 ==> download=download 2583    conf:(1)
70. diaSemana=3 horaCheia=18 2568 ==> download=download 2568    conf:(1)
71. diaSemana=2 ipGroup=wireless 2567 ==> download=download 2567    conf:(1)
72. diaSemana=3 horaCheia=12 ipGroup=servidor 2539 ==> download=download 2539    conf:(1)
73. horaCheia=15 ipGroup=aluno 2521 ==> download=download 2521    conf:(1)
74. diaSemana=6 horaCheia=16 2506 ==> download=download 2506    conf:(1)
75. diaSemana=6 ipGroup=aluno 2505 ==> download=download 2505    conf:(1)
```

Observando-se o resultado entregue, pode-se no mínimo encontrar os seguintes padrões interessantes:

- Durante a semana, o grupo “servidor” tende a fazer mais *downloads* ao meio-dia. Já o grupo “aluno” prefere às 11 horas, e o grupo “wireless” às 15 horas;
- O período de 1 hora com mais *downloads* durante a semana para cada grupo de usuários é: quinta-feira às 11 horas para o “aluno”, quarta-feira às 12 horas para o “servidor”, e quinta-feira às 15 horas para o grupo “wireless”;
- Durante toda a semana, o grupo “servidor” são os que mais praticam *download* na instituição. Em segundo lugar fica o grupo “aluno”, seguidos por último pelo grupo “wireless”;
- Referindo-se a dias de semana, os grupos “servidor”, “wireless” e “aluno” fazem mais *downloads* na quinta-feira.

3.5.5 Questão 5: Qual a relação entre dias da semana, faixas de horários, grupos de usuários e os acessos a páginas em geral?

O arquivo utilizado na mineração desta questão, foi o arquivo com os atributos: “dia da semana”, “hora cheia”, atributo “*download*” contendo o valor “comum”, “grupo de usuários” e “URL”. Porém o atributo “URL” foi removido durante o pré-processamento, já que não influencia nos resultados.

Este arquivo ARFF gerado, corresponde ao período de 18 de novembro de 2012 até 25 de novembro de 2012, e com 8.267.788 instâncias.

Para encontrar um padrão da ocorrência de um atributo em relação a ocorrência de outro atributo, foi utilizada a Associação como tarefa de mineração de dados. E para a Associação foi utilizado o algoritmo Apriori.

Após algumas tentativas, os melhores valores encontrados para os parâmetros do algoritmo Apriori foram:

- Suporte: 1%;
- Confiança: 90%;
- Número máximo de regras: 100.

3.5.5.1 Interpretação da mineração da questão 5

A Listagem 8 apresenta o resultado da mineração de dados apresentado pela Weka para a questão 5:

Listagem 8: Regras de associação da questão 5

```
Best rules found:

 1. ipGroup=servidor 5506040 ==> download=comum 5506040    conf:(1)
 2. ipGroup=aluno 2128543 ==> download=comum 2128543    conf:(1)
 3. diaSemana=3 1791496 ==> download=comum 1791496    conf:(1)
 4. diaSemana=2 1682476 ==> download=comum 1682476    conf:(1)
 5. diaSemana=5 1638800 ==> download=comum 1638800    conf:(1)
 6. diaSemana=6 1444713 ==> download=comum 1444713    conf:(1)
```

```
 7. diaSemana=4 1393772 ==> download=comum 1393772    conf:(1)
 8. diaSemana=3 ipGroup=servidor 1165800 ==> download=comum 1165800    conf:(1)
 9. diaSemana=2 ipGroup=servidor 1148502 ==> download=comum 1148502    conf:(1)
10. diaSemana=5 ipGroup=servidor 1147856 ==> download=comum 1147856    conf:(1)
11. diaSemana=6 ipGroup=servidor 1026687 ==> download=comum 1026687    conf:(1)
12. horaCheia=18 961962 ==> download=comum 961962    conf:(1)
13. diaSemana=4 ipGroup=servidor 909618 ==> download=comum 909618    conf:(1)
14. horaCheia=12 857180 ==> download=comum 857180    conf:(1)
15. horaCheia=16 844040 ==> download=comum 844040    conf:(1)
16. horaCheia=13 780239 ==> download=comum 780239    conf:(1)
17. horaCheia=10 746236 ==> download=comum 746236    conf:(1)
18. horaCheia=11 725889 ==> download=comum 725889    conf:(1)
19. horaCheia=17 724167 ==> download=comum 724167    conf:(1)
20. horaCheia=15 694103 ==> download=comum 694103    conf:(1)
21. horaCheia=18 ipGroup=servidor 646670 ==> download=comum 646670    conf:(1)
22. ipGroup=wireless 633205 ==> download=comum 633205    conf:(1)
23. horaCheia=12 ipGroup=servidor 595511 ==> download=comum 595511    conf:(1)
24. horaCheia=16 ipGroup=servidor 555932 ==> download=comum 555932    conf:(1)
25. horaCheia=13 ipGroup=servidor 547341 ==> download=comum 547341    conf:(1)
26. horaCheia=10 ipGroup=servidor 508411 ==> download=comum 508411    conf:(1)
27. horaCheia=11 ipGroup=servidor 494776 ==> download=comum 494776    conf:(1)
28. horaCheia=17 ipGroup=servidor 486319 ==> download=comum 486319    conf:(1)
29. diaSemana=2 ipGroup=aluno 441434 ==> download=comum 441434    conf:(1)
30. diaSemana=3 ipGroup=aluno 439132 ==> download=comum 439132    conf:(1)
31. horaCheia=15 ipGroup=servidor 436546 ==> download=comum 436546    conf:(1)
32. horaCheia=19 407366 ==> download=comum 407366    conf:(1)
33. diaSemana=4 ipGroup=aluno 374395 ==> download=comum 374395    conf:(1)
34. diaSemana=5 ipGroup=aluno 362178 ==> download=comum 362178    conf:(1)
35. diaSemana=6 ipGroup=aluno 330279 ==> download=comum 330279    conf:(1)
36. horaCheia=19 ipGroup=servidor 295203 ==> download=comum 295203    conf:(1)
37. horaCheia=14 294170 ==> download=comum 294170    conf:(1)
38. horaCheia=22 277527 ==> download=comum 277527    conf:(1)
39. diaSemana=7 262555 ==> download=comum 262555    conf:(1)
40. horaCheia=18 ipGroup=aluno 258009 ==> download=comum 258009    conf:(1)
41. diaSemana=5 horaCheia=18 237125 ==> download=comum 237125    conf:(1)
42. horaCheia=16 ipGroup=aluno 225064 ==> download=comum 225064    conf:(1)
43. horaCheia=21 219850 ==> download=comum 219850    conf:(1)
44. horaCheia=15 ipGroup=aluno 219257 ==> download=comum 219257    conf:(1)
45. horaCheia=9 217194 ==> download=comum 217194    conf:(1)
46. horaCheia=20 216312 ==> download=comum 216312    conf:(1)
47. diaSemana=2 horaCheia=18 211755 ==> download=comum 211755    conf:(1)
48. horaCheia=12 ipGroup=aluno 209435 ==> download=comum 209435    conf:(1)
49. horaCheia=14 ipGroup=servidor 206766 ==> download=comum 206766    conf:(1)
50. diaSemana=5 horaCheia=18 ipGroup=servidor 198743 ==> download=comum 198743   conf:(1)
51. horaCheia=10 ipGroup=aluno 197000 ==> download=comum 197000    conf:(1)
52. diaSemana=3 horaCheia=12 195133 ==> download=comum 195133    conf:(1)
53. diaSemana=3 horaCheia=18 194434 ==> download=comum 194434    conf:(1)
54. diaSemana=6 horaCheia=12 190892 ==> download=comum 190892    conf:(1)
55. diaSemana=2 horaCheia=16 189874 ==> download=comum 189874    conf:(1)
56. diaSemana=3 horaCheia=16 189339 ==> download=comum 189339    conf:(1)
57. diaSemana=3 horaCheia=13 188852 ==> download=comum 188852    conf:(1)
58. diaSemana=3 ipGroup=wireless 186564 ==> download=comum 186564    conf:(1)
59. horaCheia=23 185533 ==> download=comum 185533    conf:(1)
60. horaCheia=13 ipGroup=aluno 182788 ==> download=comum 182788    conf:(1)
61. diaSemana=7 ipGroup=aluno 174050 ==> download=comum 174050    conf:(1)
62. horaCheia=17 ipGroup=aluno 170563 ==> download=comum 170563    conf:(1)
63. horaCheia=11 ipGroup=aluno 170118 ==> download=comum 170118    conf:(1)
```

```
64. diaSemana=2 horaCheia=15 167100 ==> download=comum 167100    conf:(1)
65. diaSemana=3 horaCheia=17 165454 ==> download=comum 165454    conf:(1)
66. horaCheia=22 ipGroup=servidor 165344 ==> download=comum 165344    conf:(1)
67. diaSemana=5 horaCheia=11 164174 ==> download=comum 164174    conf:(1)
68. diaSemana=6 horaCheia=13 161034 ==> download=comum 161034    conf:(1)
69. diaSemana=4 horaCheia=18 159884 ==> download=comum 159884    conf:(1)
70. diaSemana=4 horaCheia=16 155794 ==> download=comum 155794    conf:(1)
71. diaSemana=5 horaCheia=16 155221 ==> download=comum 155221    conf:(1)
72. diaSemana=3 horaCheia=10 154043 ==> download=comum 154043    conf:(1)
73. diaSemana=3 horaCheia=11 151913 ==> download=comum 151913    conf:(1)
74. horaCheia=20 ipGroup=servidor 149064 ==> download=comum 149064    conf:(1)
75. diaSemana=6 horaCheia=10 147029 ==> download=comum 147029    conf:(1)
76. diaSemana=2 horaCheia=12 145505 ==> download=comum 145505    conf:(1)
77. diaSemana=5 horaCheia=17 144400 ==> download=comum 144400    conf:(1)
78. diaSemana=2 horaCheia=10 143432 ==> download=comum 143432    conf:(1)
79. diaSemana=6 horaCheia=12 ipGroup=servidor 143304 ==> download=comum 143304   conf:(1)
80. diaSemana=5 horaCheia=10 142511 ==> download=comum 142511    conf:(1)
81. diaSemana=2 horaCheia=17 142294 ==> download=comum 142294    conf:(1)
82. diaSemana=4 horaCheia=12 142294 ==> download=comum 142294    conf:(1)
83. diaSemana=5 horaCheia=12 141726 ==> download=comum 141726    conf:(1)
84. diaSemana=3 horaCheia=15 139291 ==> download=comum 139291    conf:(1)
85. diaSemana=6 horaCheia=18 138354 ==> download=comum 138354    conf:(1)
86. diaSemana=4 horaCheia=13 135934 ==> download=comum 135934    conf:(1)
87. diaSemana=4 horaCheia=10 135210 ==> download=comum 135210    conf:(1)
88. diaSemana=3 horaCheia=13 ipGroup=servidor 134795 ==> download=comum 134795   conf:(1)
89. diaSemana=3 horaCheia=12 ipGroup=servidor 134788 ==> download=comum 134788   conf:(1)
90. horaCheia=21 ipGroup=servidor 133804 ==> download=comum 133804    conf:(1)
91. diaSemana=3 horaCheia=16 ipGroup=servidor 132310 ==> download=comum 132310   conf:(1)
92. diaSemana=6 horaCheia=11 131348 ==> download=comum 131348    conf:(1)
93. diaSemana=2 horaCheia=13 131060 ==> download=comum 131060    conf:(1)
94. diaSemana=4 horaCheia=15 130920 ==> download=comum 130920    conf:(1)
95. diaSemana=2 horaCheia=11 130401 ==> download=comum 130401    conf:(1)
96. diaSemana=5 ipGroup=wireless 128766 ==> download=comum 128766    conf:(1)
97. diaSemana=5 horaCheia=13 126233 ==> download=comum 126233    conf:(1)
98. diaSemana=4 horaCheia=17 125427 ==> download=comum 125427    conf:(1)
99. horaCheia=9 ipGroup=servidor 125255 ==> download=comum 125255    conf:(1)
100. diaSemana=6 horaCheia=16 124260 ==> download=comum 124260    conf:(1)
```

Observando-se o resultado entregue, pode-se no mínimo encontrar os seguintes padrões interessantes:

- O período de 1 hora, entre os dias da semana, com o maior uso comum de Internet é na quinta-feira às 18h pelo grupo de usuários “servidor”;
- O grupo “aluno” possui o maior uso comum da Internet na segunda-feira. E durante a semana são às 18h que este grupo tem maior tendência ao uso comum;
- Em ordem decrescente, o uso comum de Internet para o grupo “servidor” se estabelece nos dias de semana: terça-feira, segunda-feira, quinta-feira, sexta-feira, quarta-feira, sábado e domingo;

- Conforme a regra mais confiável encontrada, o grupo de usuários “servidor” são os que mais utilizam a Internet de maneira comum, correspondendo a 5.506.040 de instâncias. Os alunos estão em segundo lugar com 2.128.543 instâncias, e por último o grupo “wireless” com 633.205 instâncias;
- Já o grupo “wireless” possui o maior uso comum da Internet na terça-feira.

3.5.6 Questão 6: Qual a relação entre dias da semana, faixas de horários, grupos de usuários e o acesso aos sistemas da UTFPR?

O arquivo utilizado na mineração desta questão, foi o arquivo com os atributos: “dia da semana”, “hora cheia”, “*downloads*”, “grupo de usuários” e “URL” com a raiz do domínio igual a “utfpr.edu.br”. Porém o atributo “*downloads*” foi removido durante o pré-processamento, já que não influencia nos resultados.

Este arquivo ARFF gerado, corresponde ao período de 18 de novembro de 2012 até 25 de novembro de 2012, e com 716.569 instâncias.

Para encontrar um padrão da ocorrência de um atributo em relação a ocorrência de outro atributo, foi utilizada a Associação como tarefa de mineração de dados. E para a Associação foi utilizado o algoritmo Apriori.

Após algumas tentativas, os melhores valores encontrados para os parâmetros do algoritmo Apriori foram:

- Suporte: 1%;
- Confiança: 50%;
- Número máximo de regras: 100.

3.5.6.1 Interpretação da mineração da questão 6

A Listagem 9 apresenta o resultado da mineração de dados apresentado pela Weka para a questão 6:

Listagem 9: Regras de associação da questão 6

```
Best rules found:

 1. uri=wpad.md.utfpr.edu.br 25035 ==> ipGroup=wireless 25034    conf:(1)
 2. horaCheia=10 uri=mail.utfpr.edu.br 11667 ==> ipGroup=servidor 11370    conf:(0.97)
 3. diaSemana=3 uri=biblioteca.utfpr.edu.br 10064 ==> ipGroup=servidor 9750    conf:(0.97)
 4. diaSemana=2 uri=mail.utfpr.edu.br 23452 ==> ipGroup=servidor 22437    conf:(0.96)
 5. horaCheia=10 uri=ead.utfpr.edu.br 9170 ==> ipGroup=servidor 8773    conf:(0.96)
 6. horaCheia=18 uri=scooby.utfpr.edu.br 19118 ==> ipGroup=servidor 18146    conf:(0.95)
 7. horaCheia=17 uri=scooby.utfpr.edu.br 16716 ==> ipGroup=servidor 15839    conf:(0.95)
 8. horaCheia=16 uri=scooby.utfpr.edu.br 20669 ==> ipGroup=servidor 19570    conf:(0.95)
 9. diaSemana=5 uri=scooby.utfpr.edu.br 27897 ==> ipGroup=servidor 26077    conf:(0.93)
10. diaSemana=6 uri=scooby.utfpr.edu.br 19664 ==> ipGroup=servidor 18364    conf:(0.93)
11. diaSemana=5 horaCheia=16 17404 ==> ipGroup=servidor 16199    conf:(0.93)
12. horaCheia=18 uri=ead.utfpr.edu.br 12873 ==> ipGroup=servidor 11835    conf:(0.92)
13. horaCheia=12 uri=scooby.utfpr.edu.br 11653 ==> ipGroup=servidor 10699    conf:(0.92)
14. diaSemana=5 horaCheia=18 15482 ==> ipGroup=servidor 14059    conf:(0.91)
15. horaCheia=12 uri=mail.utfpr.edu.br 12057 ==> ipGroup=servidor 10855    conf:(0.9)
16. horaCheia=15 uri=scooby.utfpr.edu.br 9968 ==> ipGroup=servidor 8969    conf:(0.9)
17. horaCheia=15 uri=mail.utfpr.edu.br 11679 ==> ipGroup=servidor 10493    conf:(0.9)
18. diaSemana=5 horaCheia=19 9233 ==> ipGroup=servidor 8294    conf:(0.9)
19. uri=scooby.utfpr.edu.br 130890 ==> ipGroup=servidor 116595    conf:(0.89)
20. horaCheia=13 uri=scooby.utfpr.edu.br 8968 ==> ipGroup=servidor 7987    conf:(0.89)
21. diaSemana=4 uri=scooby.utfpr.edu.br 25792 ==> ipGroup=servidor 22910    conf:(0.89)
22. horaCheia=11 uri=scooby.utfpr.edu.br 12599 ==> ipGroup=servidor 11173    conf:(0.89)
23. diaSemana=4 uri=mail.utfpr.edu.br 21184 ==> ipGroup=servidor 18733    conf:(0.88)
24. diaSemana=2 horaCheia=13 11696 ==> ipGroup=servidor 10201    conf:(0.87)
25. diaSemana=2 uri=scooby.utfpr.edu.br 24926 ==> ipGroup=servidor 21736    conf:(0.87)
26. horaCheia=10 uri=scooby.utfpr.edu.br 10935 ==> ipGroup=servidor 9522    conf:(0.87)
27. diaSemana=5 uri=ead.utfpr.edu.br 33199 ==> ipGroup=servidor 28535    conf:(0.86)
28. diaSemana=5 horaCheia=10 10407 ==> ipGroup=servidor 8941    conf:(0.86)
29. diaSemana=6 uri=mail.utfpr.edu.br 22716 ==> ipGroup=servidor 19492    conf:(0.86)
30. diaSemana=4 horaCheia=13 13451 ==> ipGroup=servidor 11429    conf:(0.85)
31. diaSemana=3 uri=scooby.utfpr.edu.br 30354 ==> ipGroup=servidor 25710    conf:(0.85)
32. diaSemana=4 horaCheia=12 14014 ==> ipGroup=servidor 11797    conf:(0.84)
33. diaSemana=6 horaCheia=11 14181 ==> ipGroup=servidor 11907    conf:(0.84)
34. horaCheia=16 uri=mail.utfpr.edu.br 9695 ==> ipGroup=servidor 8138    conf:(0.84)
35. horaCheia=18 uri=mail.utfpr.edu.br 11095 ==> ipGroup=servidor 9310    conf:(0.84)
36. horaCheia=13 uri=mail.utfpr.edu.br 9768 ==> ipGroup=servidor 8189    conf:(0.84)
37. diaSemana=2 horaCheia=10 12692 ==> ipGroup=servidor 10548    conf:(0.83)
38. uri=mail.utfpr.edu.br 117118 ==> ipGroup=servidor 96216    conf:(0.82)
39. diaSemana=3 horaCheia=10 17372 ==> ipGroup=servidor 14177    conf:(0.82)
40. diaSemana=6 horaCheia=10 9703 ==> ipGroup=servidor 7903    conf:(0.81)
41. diaSemana=5 horaCheia=15 9839 ==> ipGroup=servidor 7977    conf:(0.81)
42. diaSemana=4 horaCheia=17 13774 ==> ipGroup=servidor 11059    conf:(0.8)
43. horaCheia=12 uri=ead.utfpr.edu.br 14608 ==> ipGroup=servidor 11720    conf:(0.8)
44. horaCheia=16 uri=ead.utfpr.edu.br 18206 ==> ipGroup=servidor 14571    conf:(0.8)
45. diaSemana=3 horaCheia=12 15124 ==> ipGroup=servidor 12030    conf:(0.8)
46. diaSemana=2 horaCheia=11 10980 ==> ipGroup=servidor 8730    conf:(0.8)
47. diaSemana=2 horaCheia=18 16973 ==> ipGroup=servidor 13442    conf:(0.79)
48. horaCheia=10 62868 ==> ipGroup=servidor 49667    conf:(0.79)
49. diaSemana=4 horaCheia=15 12838 ==> ipGroup=servidor 10032    conf:(0.78)
50. diaSemana=2 uri=moodle2.md.utfpr.edu.br 17587 ==> ipGroup=servidor 13738    conf:(0.78)
51. diaSemana=5 horaCheia=17 16623 ==> ipGroup=servidor 12900    conf:(0.78)
52. horaCheia=18 74038 ==> ipGroup=servidor 57435    conf:(0.78)
53. diaSemana=4 uri=ead.utfpr.edu.br 24991 ==> ipGroup=servidor 19365    conf:(0.77)
```

```
54. diaSemana=3 horaCheia=18 20226 ==> ipGroup=servidor 15649    conf:(0.77)
55. diaSemana=4 horaCheia=16 14594 ==> ipGroup=servidor 11251    conf:(0.77)
56. horaCheia=13 uri=ead.utfpr.edu.br 21868 ==> ipGroup=servidor 16764    conf:(0.77)
57. horaCheia=16 81746 ==> ipGroup=servidor 62593    conf:(0.77)
58. horaCheia=13 uri=www.utfpr.edu.br 12313 ==> ipGroup=servidor 9406    conf:(0.76)
59. diaSemana=5 horaCheia=12 12037 ==> ipGroup=servidor 9194    conf:(0.76)
60. uri=ead.utfpr.edu.br 134598 ==> ipGroup=servidor 102256    conf:(0.76)
61. diaSemana=2 uri=ead.utfpr.edu.br 20396 ==> ipGroup=servidor 15494    conf:(0.76)
62. diaSemana=3 horaCheia=16 17817 ==> ipGroup=servidor 13416    conf:(0.75)
63. horaCheia=17 uri=ead.utfpr.edu.br 13921 ==> ipGroup=servidor 10472    conf:(0.75)
64. horaCheia=19 31829 ==> ipGroup=servidor 23929    conf:(0.75)
65. horaCheia=17 uri=mail.utfpr.edu.br 14466 ==> ipGroup=servidor 10848    conf:(0.75)
66. diaSemana=6 horaCheia=16 11089 ==> ipGroup=servidor 8221    conf:(0.74)
67. horaCheia=13 65579 ==> ipGroup=servidor 48415    conf:(0.74)
68. diaSemana=4 129152 ==> ipGroup=servidor 95158    conf:(0.74)
69. diaSemana=3 uri=ead.utfpr.edu.br 40525 ==> ipGroup=servidor 29815    conf:(0.74)
70. horaCheia=12 70951 ==> ipGroup=servidor 51902    conf:(0.73)
71. diaSemana=5 uri=mail.utfpr.edu.br 23724 ==> ipGroup=servidor 17329    conf:(0.73)
72. horaCheia=10 uri=www.utfpr.edu.br 14716 ==> ipGroup=servidor 10707    conf:(0.73)
73. diaSemana=5 145458 ==> ipGroup=servidor 105013    conf:(0.72)
74. horaCheia=17 74428 ==> ipGroup=servidor 53480    conf:(0.72)
75. horaCheia=11 uri=mail.utfpr.edu.br 12814 ==> ipGroup=servidor 9204    conf:(0.72)
76. uri=biblioteca.utfpr.edu.br 37207 ==> ipGroup=servidor 26705    conf:(0.72)
77. horaCheia=16 uri=www.utfpr.edu.br 18086 ==> ipGroup=servidor 12967    conf:(0.72)
78. diaSemana=4 uri=www.utfpr.edu.br 29778 ==> ipGroup=servidor 20938    conf:(0.7)
79. diaSemana=2 156423 ==> ipGroup=servidor 109758    conf:(0.7)
80. diaSemana=6 horaCheia=12 11536 ==> ipGroup=servidor 8062    conf:(0.7)
81. horaCheia=12 uri=www.utfpr.edu.br 14071 ==> ipGroup=servidor 9829    conf:(0.7)
82. diaSemana=3 uri=mail.utfpr.edu.br 24974 ==> ipGroup=servidor 17369    conf:(0.7)
83. diaSemana=4 horaCheia=10 11094 ==> ipGroup=servidor 7714    conf:(0.7)
84. diaSemana=2 horaCheia=17 15675 ==> ipGroup=servidor 10825    conf:(0.69)
85. horaCheia=15 57337 ==> ipGroup=servidor 39434    conf:(0.69)
86. diaSemana=6 103291 ==> ipGroup=servidor 70994    conf:(0.69)
87. diaSemana=3 horaCheia=13 13787 ==> ipGroup=servidor 9460    conf:(0.69)
88. diaSemana=6 uri=www.utfpr.edu.br 23302 ==> ipGroup=servidor 15950    conf:(0.68)
89. diaSemana=2 horaCheia=12 14635 ==> ipGroup=servidor 10001    conf:(0.68)
90. horaCheia=20 20675 ==> ipGroup=servidor 14126    conf:(0.68)
91. horaCheia=18 uri=www.utfpr.edu.br 15378 ==> ipGroup=servidor 10433    conf:(0.68)
92. diaSemana=3 horaCheia=11 11750 ==> ipGroup=servidor 7958    conf:(0.68)
93. diaSemana=5 uri=www.utfpr.edu.br 32590 ==> ipGroup=servidor 22045    conf:(0.68)
94. diaSemana=3 166001 ==> ipGroup=servidor 112105    conf:(0.68)
95. diaSemana=3 horaCheia=17 20103 ==> ipGroup=servidor 13509    conf:(0.67)
96. horaCheia=11 65886 ==> ipGroup=servidor 44236    conf:(0.67)
97. diaSemana=2 horaCheia=16 20018 ==> ipGroup=servidor 13413    conf:(0.67)
98. diaSemana=5 horaCheia=13 13403 ==> ipGroup=servidor 8938    conf:(0.67)
99. diaSemana=6 horaCheia=13 10828 ==> ipGroup=servidor 7184    conf:(0.66)
100. horaCheia=11 uri=www.utfpr.edu.br 11951 ==> ipGroup=servidor 7927    conf:(0.66)
```

Observando-se o resultado entregue, pode-se no mínimo encontrar os seguintes padrões interessantes:

- Em relação ao endereço “www.utfpr.edu.br”, este possui em ordem decrescente, as cinco maiores quantias de acessos às 10h, 16h, 12h, 18h e 11h. Já referente ao dia

da semana, as três maiores quantias de acessos ocorrem na quarta-feira, e em seguida na sexta-feira e quinta-feira;

- Dentre as regras, é possível perceber que o endereço “biblioteca.utfpr.edu.br” é principalmente acessado na terça-feira;
- Já o endereço “ead.utfpr.edu.br”, possui as 8 maiores quantias de acessos, em ordem decrescente, às 10h, 18h, 12h, 16h, 13h e 17h. Em relação aos dias da semana, tem as quatro maiores quantias de acesso na quinta-feira, e depois na quarta-feira, segunda-feira e terça-feira;
- Quanto ao endereço “scooby.utfpr.edu.br”, este tem as 8 maiores quantias de acesso, em ordem decrescente, às 18h, 17h, 16h, 12h, 15h, 13h, 11h e 10h. Com relação ao dia da semana tem as 5 maiores quantias de acessos na quinta-feira, sexta-feira, quarta-feira, segunda-feira e terça-feira;
- A única regra referente ao endereço “moodle2.md.utfpr.edu.br” indica que seu maior uso concentra-se na segunda-feira;
- A melhor regra encontrada indica que o grupo “wireless” corresponde a 25.034 acessos dos 25.035 realizados para o endereço “wpad.md.utfpr.edu.br”. Esta também foi a única regra que conteve este grupo;
- Também foi possível perceber que o grupo “aluno” não apareceu nas 100 primeiras regras encontradas;
- Quanto ao endereço “mail.utfpr.edu.br”, é possível identificar que independente do dia da semana tem mais acessos às 10h, e os 7 outros maiores horários são às 12h, 15h, 16h 18h, 13h, 17h e 11h. Em relação ao dia da semana, tem mais acessos na segunda-feira.

3.5.7 Questão 7: Qual a relação entre os grupos de usuários e as URL's acessadas?

O arquivo utilizado na mineração desta questão, foi o arquivo com os atributos: “dia da semana”, “hora cheia”, “*downloads*”, “grupo de usuários” e “URL”. Porém os atributos “dia da semana”, “hora cheia” e “*downloads*” foram removidos durante o pré-processamento, já que não influenciam nos resultados.

Este arquivo ARFF gerado, corresponde ao período de 18 de novembro de 2012 até 25 de novembro de 2012, e com 5.225.073 instâncias.

Para encontrar um padrão da ocorrência de um atributo em relação a ocorrência de outro atributo, foi utilizada a Associação como tarefa de mineração de dados. E para a Associação foi utilizado o algoritmo Apriori.

Após algumas tentativas, os melhores valores encontrados para os parâmetros do algoritmo Apriori foram:

- Suporte: 0.5%;
- Confiança: 10%;
- Número máximo de regras: 100.

3.5.7.1 Interpretação da mineração da questão 7

A Listagem 10 apresenta o resultado da mineração de dados apresentado pela Weka para a questão 7:

Listagem 10: Regras de associação da questão 7

```
Best rules found:

 1. uri=terra.com 70900 ==> ipGroup=servidor 69114    conf:(0.97)
 2. uri=scooby.utfpr.edu.br 130890 ==> ipGroup=servidor 116595    conf:(0.89)
 3. uri=mail.utfpr.edu.br 117118 ==> ipGroup=servidor 96216    conf:(0.82)
 4. uri=fbcdn.net 721965 ==> ipGroup=servidor 579816    conf:(0.8)
 5. uri=facebook.com 751221 ==> ipGroup=servidor 576842    conf:(0.77)
 6. uri=ead.utfpr.edu.br 134598 ==> ipGroup=servidor 102256    conf:(0.76)
 7. uri=msn.com 69056 ==> ipGroup=servidor 50030    conf:(0.72)
 8. uri=trendmicro.com 700433 ==> ipGroup=servidor 501919    conf:(0.72)
 9. uri=akamaihd.net 191960 ==> ipGroup=servidor 134166    conf:(0.7)
10. uri=youtube.com 70059 ==> ipGroup=servidor 45804    conf:(0.65)
11. uri=www.utfpr.edu.br 157745 ==> ipGroup=servidor 101659    conf:(0.64)
12. uri=live.com 171958 ==> ipGroup=servidor 109027    conf:(0.63)
13. uri=gstatic.com 172723 ==> ipGroup=servidor 107494    conf:(0.62)
14. uri=googlesyndication.com 59629 ==> ipGroup=servidor 36353    conf:(0.61)
15. uri=ytimg.com 55901 ==> ipGroup=servidor 34059    conf:(0.61)
16. uri=google.com 267464 ==> ipGroup=servidor 162266    conf:(0.61)
17. uri=imguol.com 80500 ==> ipGroup=servidor 48190    conf:(0.6)
18. uri=globo.com 210190 ==> ipGroup=servidor 124937    conf:(0.59)
19. uri=google-analytics.com 123593 ==> ipGroup=servidor 73397    conf:(0.59)
20. uri=google.com.br 219463 ==> ipGroup=servidor 128373    conf:(0.58)
21. uri=glbimg.com 400400 ==> ipGroup=servidor 226161    conf:(0.56)
```

```
22. uri=doubleclick.net 113833 ==> ipGroup=servidor 61456    conf:(0.54)
23. uri=uol.com.br 86433 ==> ipGroup=servidor 45864    conf:(0.53)
24. uri=moodle2.md.utfpr.edu.br 56945 ==> ipGroup=aluno 26584    conf:(0.47)
25. uri=twitter.com 90096 ==> ipGroup=servidor 41932    conf:(0.47)
26. uri=doubleclick.net 113833 ==> ipGroup=aluno 45989    conf:(0.4)
27. uri=twitter.com 90096 ==> ipGroup=aluno 35635    conf:(0.4)
28. uri=glbimg.com 400400 ==> ipGroup=aluno 151070    conf:(0.38)
29. uri=uol.com.br 86433 ==> ipGroup=aluno 31272    conf:(0.36)
30. uri=google.com.br 219463 ==> ipGroup=aluno 78399    conf:(0.36)
31. uri=globo.com 210190 ==> ipGroup=aluno 72425    conf:(0.34)
32. uri=google-analytics.com 123593 ==> ipGroup=aluno 42559    conf:(0.34)
33. uri=google.com 267464 ==> ipGroup=aluno 87486    conf:(0.33)
34. uri=gstatic.com 172723 ==> ipGroup=aluno 56411    conf:(0.33)
35. uri=trendmicro.com 700433 ==> ipGroup=aluno 193605    conf:(0.28)
36. uri=www.utfpr.edu.br 157745 ==> ipGroup=aluno 43161    conf:(0.27)
37. uri=live.com 171958 ==> ipGroup=aluno 44840    conf:(0.26)
38. uri=akamaihd.net 191960 ==> ipGroup=aluno 46936    conf:(0.24)
39. uri=ead.utfpr.edu.br 134598 ==> ipGroup=wireless 32215    conf:(0.24)
40. ipGroup=servidor 3599823 ==> uri=fbcdn.net 579816    conf:(0.16)
41. ipGroup=servidor 3599823 ==> uri=facebook.com 576842    conf:(0.16)
42. uri=facebook.com 751221 ==> ipGroup=aluno 119407    conf:(0.16)
43. ipGroup=wireless 349665 ==> uri=facebook.com 54972    conf:(0.16)
44. ipGroup=aluno 1275585 ==> uri=trendmicro.com 193605    conf:(0.15)
45. uri=fbcdn.net 721965 ==> ipGroup=aluno 104609    conf:(0.14)
46. ipGroup=servidor 3599823 ==> uri=trendmicro.com 501919    conf:(0.14)
47. ipGroup=aluno 1275585 ==> uri=glbimg.com 151070    conf:(0.12)
48. ipGroup=wireless 349665 ==> uri=fbcdn.net 37540    conf:(0.11)
```

Observando-se o resultado entregue, pode-se no mínimo encontrar os seguintes padrões interessantes:

- Os 3 endereços que o grupo “servidor” mais tende a acessar são “fbcdn.net” (16%), “facebook.com” (16%) e “trendmicro.com” (14%);
- Já o grupo “alunos”, possui como os 2 endereços de mais acesso sendo “trendmicro.com” (15%) e “glbimg.com” (12%);
- Os 2 endereços que o grupo “wireless” mais tende a acessar são “facebook.com” (16%) e “fbcdn.net” (11%);
- O endereço “moodle2.md.utfpr.edu.br” tende a ser mais acessado pelo grupo “aluno” 47%;
- O endereço “twitter.com” tende a ser mais acessado pelo grupo “servidor”;
- Os vinte endereços mais acessados correspondem ao grupo “servidor”, e são eles, em ordem decrescente: “terra.com”, “scooby.utfpr.edu.br”, “mail.utfpr.edu.br”, “fbcdn.net”, “facebook.com”, “ead.utfpr.edu.br”, “msn.com”, “trendmicro.com”, “akamaihd.net”, “youtube.com”, “www.utfpr.edu.br”, “live.com”, “gstatic.com”,

“googlesyndication.com”, “ytimg.com”, “google.com”, “imguol.com”, “globo.com”, “google-analytics.com” e “google.com.br”.

4. CONSIDERAÇÕES FINAIS

4.1 CONCLUSÃO

Durante a execução do projeto, foi possível experimentar a situação da impossibilidade de execução de parte do planejamento. Em relação à isto, os dois principais momentos foram:

- Impossibilidade de criação de um sistema automatizado: a maior parte do sistema proposto teve sua criação inviabilizada. O tratamento do preprocessamento e o acoplamento das funcionalidades da Weka ao sistema web foram os principais motivos;
- Impossibilidade de processamento dos arquivos ARFF iniciais: os primeiros modelos de arquivo ARFF propostos não permitiam a ferramenta Weka realizar seu processamento.

Quanto ao primeiro problema, a fim de poder dar sequência ao objetivo central do projeto (o processo KDD), decidiu-se por não desenvolver o sistema web, e com isso não automatizar boa parte do processo. Para permanecer dentro dos prazos, e não atingir uma complexidade extrema de desenvolvimento, a solução aderida foi a única opção observada.

Já quanto ao segundo problema, a persistência na busca por uma solução foi o que possibilitou a continuação do projeto. Foi necessário expandir a visão, com propósito de perceber uma solução simples e efetiva, que veio a possibilitar o processamento dos arquivos ARFF.

Quanto a interpretação dos padrões encontrados, o próximo passo cabe ao administrador de redes, que deve analisá-los com intuito de transformar o conhecimento obtido em uma aplicação prática em seu escopo de trabalho.

Observa-se que foi identificado durante a mineração, que o sistema de e-mails da UTFPR (mail.utfpr.edu.br), independente do dia da semana tem mais acessos às 10h, e os 7 outros maiores horários são às 12h, 15h, 16h 18h, 13h, 17h e 11h. Em relação ao dia da semana, tem mais acessos na segunda-feira. Isso significa, que estes são os períodos de tempo com as piores indicações para se realizar uma manutenção do sistema de e-mail. Com o

mesmo padrão é possível notar também, que os servidores tendem a ler/responder seus e-mails com mais intensidade no período da tarde. Assim, o administrador de redes supostamente poderia realizar um questionamento como: “Mas qual é o motivo pelo qual os usuários estão acessando menos seus e-mails pela manhã? Será que há algo de errado?”.

Observando estes pontos, pode-se concluir que o objetivo central do projeto foi conquistado. Pois houve a execução do processo KDD para obtenção de conhecimento em arquivos de *logs* de servidor, que trouxeram padrões sobre o uso da Internet pelos usuários, na UTFPR Câmpus Medianeira. Permitindo então, a reafirmação de informações já existentes, trazendo certeza sobre conhecimentos. E também a apresentação de fatos pouco conhecidos sobre o comportamento dos usuários, trazendo conhecimento direto ou indicando a necessidade de uma análise mais profunda.

4.2 TRABALHOS FUTUROS/CONTINUAÇÃO DO TRABALHO

Este trabalho abriu várias oportunidades para projetos futuros. Dentre elas pode-se citar ao menos:

- Um caso de uso sobre a criação de um sistema automatizado de mineração genérica de *logs* de servidores. Já que as principais problemáticas foram citadas no presente trabalho. Este tem forte indicações de poder se tornar um produto comercializável;
- Um estudo de caso sobre a aplicação da API da Weka. Neste trabalho ficou clara a necessidade do conhecimento do uso da API, para aplicação em situações que são específicas;
- Um estudo sobre as capacidades de processamento da ferramenta Weka, tanto como limitações e configurações avançadas do aplicativo. Também, sobre como a ferramenta Weka pode ser utilizada para mineração de dados de arquivos extremamente grandes;
- Estudo comparativo entre ferramentas de mineração de dados que vão além da Weka, a fim de apresentar alternativas de uso, e pontos fortes e fracos sobre cada uma.

5. REFERÊNCIAS BIBLIOGRÁFICAS

ABERNETHY, Michael. **Mineração de dados com WEKA, Parte 1: Introdução e regressão**. 2010 . Disponível em: <http://www.ibm.com/developerworks/br/opensource/library/os-weka1/>. Acesso em: 01 de março de 2013.

AGRAWAL, R.; SRIKANT, R.; **Fast Algorithms for Mining Association Rules, In Proc. Of the 20th Int'l Conference on Very Large Databases**, Santiago, Chile, Set, 1994.

AMO, Sandra de. **Técnicas de Mineração de Dados**. 2004. Faculdade de Computação – Universidade de Uberlândia. Disponível em: <http://www.lsi.ufu.br/documentos/publicacoes/ano/2004/JAI-cap5.pdf>. Acesso em: 01 de março de 2013.

AMORIM, Thiago. **Conceitos, técnicas, ferramentas e aplicações de Mineração de Dados para gerar conhecimento a partir de bases de dados**. 2006. 50 f. Graduação em Ciência da Computação – Centro de Informática – Universidade Federal de Pernambuco. Disponível em: <http://www.cin.ufpe.br/~tg/2006-2/tmas.pdf>. Acesso em: 01 de março de 2013.

APACHE. **What is the Apache HTTP Server Project?** 2013. Disponível em: <http://httpd.apache.org/ABOUT_APACHE.html>. Acesso em: 01 de março de 2013.

BOGORNY, Vania. **Algoritmos e Ferramentas de Descoberta de Conhecimento em Bancos de Dados Geográficos**. 2003. 43 f. Programa de Pós-graduação em Computação – Instituto de Informática -Universidade Federal do Rio Grande do Sul. Disponível em: < http://www.inf.ufrgs.br/~vbogorny/ti3-final.pdf>.Acesso em: 01 de março de 2013.

CGI.BR. **CGI.br divulga os resultados da Pesquisa TIC Empresas 2009**. 2010. Disponível em: <http://www.nic.br/imprensa/releases/2010/rl-2010-08.htm>. Acesso em: 01 de março de 2013.

CORREA, A. C. G.; SCHIABEL, H.; **Descoberta de Conhecimento em Base de Imagens Mamográficas**. USP. USP, São Carlos, 2004.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. **From Data Mining to Knowledge Discovery in Databases**. 1996. Disponível em:

<http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf>. Acesso em: 01 de março de 2013.

GALVÃO, Noemi D; MARIN, Heimar F. **Técnica de mineração de dados: uma revisão da literatura**. 2009. Disponível em: <http://www.scielo.br/pdf/ape/v22n5/14.pdf>. Acesso em: 01 de março de 2013.

GRENNAN, Mark. **Firewall and Proxy Server HOWTO**. 2000. Disponível em: <http://www.tldp.org/HOWTO/Firewall-HOWTO.html#toc1>. Acesso em: 01 de março de 2013.

MACEDO, Dayana C. de; MATOS, Simone N. **Extração de conhecimento através da Mineração de Dados**. 2010. Revista de Engenharia e Tecnologia. Versão 2, Nº 2, Ago/2010. Disponível em: <http://www.revistaret.com.br/ojs-2.2.3/index.php/ret/article/viewFile/38/73>. Acesso em: 01 de março de 2013.

MICROSOFT. **O que é um servidor proxy?** 2013. Disponível em: <http://windows.microsoft.com/pt-br/windows-vista/what-is-a-proxy-server>. Acesso em: 01 de março de 2013.

MORIMOTO, Carlos E. **Redes e Servidores Linux – Guia prático, 2ed**. 2006. Disponível em: <http://www.hardware.com.br/livros/linux-redes/>. Acesso em: 01 de março de 2013.

ROECKL, Chris. **Stateful Inspection Firewalls. An overview of firewall technology and how Juniper Networks implements it**. 2004. Disponível em: <http://www.abchost.cz/download/204-4/juniper-%25EE%2580%2580stateful%25EE %2580%2581-inspection-firewall.pdf>. Acesso em: 01 de março de 2013.

SANTOS, Rafael. **Conceitos de Mineração de Dados na Web**. 2009. Disponível em: <http://www.lac.inpe.br/~rafael.santos/Docs/WebMedia/2009/webmedia2009.pdf>. Acesso em: 01 de março de 2013.

SFERRA, Heloisa H.; CORRÊA, Ângela M. C. J. **Conceitos e Aplicações de Data Mining**. 2003. Disponível em: <http://www.unimep.br/phpg/editora/revistaspdf/rct22art02.pdf>. Acesso em: 01 de março de 2013.

SQUID-CACHE.ORG. **What is Squid?** 2013. Disponível em: <http://www.squid-cache.org/Intro/>. Acesso em: 01 de março de 2013.

SOUZA, Ailton. **Servidor WEB Apache, o que esperar?** 2013. Disponível em: <http://www.devmedia.com.br/servidor-web-apache-o-que-esperar/7096>. Acesso em: 01 de março de 2013.

SOUZA, Pablo S. **Processo de descoberta de conhecimento aplicado a uma base de dados agroclimáticos**. 2008. 25 f. Monografia – Curso de Ciência da Computação – Universidade Estadual de Mato Grosso do Sul. Disponível em: <http://www.uems.br/portal/biblioteca/repositorio/2011-08-11_21-22-41.pdf>. Acesso em: 01 de março de 2013.

WEKA. **Weka 3: Data Mining Software in Java**. 2013a. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/index.html>. Acesso em: 01 de março de 2013.

WEKA. **Weka user manual**. 2013b. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/documentation.html>. Acesso em: 01 de março de 2013.

WEKA. **Weka's history**. 2013c. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/documentation.html>. Acesso em: 01 de março de 2013.

WEKA. **XRFF**. 2013d. Disponível em: <http://weka.wikispaces.com/XRFF>. Acesso em: 01 de março de 2013.

WESSELS, D. **Squid: The Definitive Guide**. O'Reilly, 2004.

WIKI.SQUID-CACHE.ORG. **Squid FAQ/Squid ACL**. 2012. Disponível em: <http://wiki.squid-cache.org/SquidFaq/SquidAcl>. Acesso em: 01 de março de 2013.

APÊNDICES

APÊNDICE A

```
package pacote;

import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.StringTokenizer;
import java.util.TimeZone;

public class ProcessadorTCC {
   private final String OUTPUT_FILE_ARFF = "C:\\processamento\\arff\\arquivo.arff";
   private final String INPUT_FILE_LOG = "C:\\processamento\\LogsSquid\\log.log";
   private final String OUTPUT_FILE_CLASSES_URI = "C:\\processamento\\arff\\classes.txt";
   Path pathInput = Paths.get(INPUT_FILE_LOG);
   private final Charset ENCODING = StandardCharsets.UTF_8;
   private final String SEPARADOR = ",";
   int linhasTotais = 0;
   int linhasArquivo = 0;
   int linhasIgnoradas = 0;
   String linhaCrua = null;
   StringBuilder linhaARFF = null;
   boolean descartar = false;
   boolean utfprUri = false;
   List<String> listaURI = new ArrayList<>();
   HashMap<String, Integer> listaClassesUriMaisUsadas = new HashMap<>();
   HashMap<String, Integer> uriHashMap = new HashMap<>();

   public ProcessadorTCC() {}

   public void preprocessar() throws Exception, IOException {
      try (Scanner scanner = new Scanner(pathInput, ENCODING.name())) {
         while (scanner.hasNextLine()) {
            linhaCrua = null;
            linhaCrua = scanner.nextLine();
            StringTokenizer st = new StringTokenizer(linhaCrua, " ");
            descartar = false;
            utfprUri = false;
            if (st.countTokens() != 10) {descartar = true;}
            if (!descartar) {
               linhaARFF = new StringBuilder();
               st.nextToken();
               st.nextToken();
               String clientGroupStr = st.nextToken();
               if (clientGroupStr.split("\\.").length == 4) {
                  String clientGroup[] = clientGroupStr.split("\\.");
                  String a = clientGroup[0] + "." + clientGroup[1];
                  if (a.equals("172.27")) {
                  } else if (a.equals("192.168")) {
                     Integer b = Integer.parseInt(clientGroup[2]);
                     if ((b >= 0) && (b <= 7)) {
                     } else if ((b == 8) || ((b >= 13) && (b <= 18)) || ((b >= 24) && (b <= 26))) {
                     } else { descartar = true; }
                  } else { descartar = true; }
               } else { descartar = true; }
               String actionCodeStr = st.nextToken();
```

```
                String resultGroup = null;
                if (actionCodeStr.split("/").length > 1) {
                    String actionCode[] = actionCodeStr.split("/");
                    if (actionCode[1].charAt(0) == '2') {
                        resultGroup = "success";
                    } else {
                        resultGroup = "other";
                        descartar = true;
                    }
                } else { descartar = true; }
                Long size = Long.parseLong(st.nextToken());
                st.nextToken();
                String uriStr = st.nextToken();
                int uriPosInicial = 0;
                String uri[] = uriStr.split("://");
                if (uri.length > 1) {
                    uriPosInicial = 1;
                }
                String uri2[] = uri[uriPosInicial].split("/");
                String uri3[] = uri2[0].split(":");
                String uri4[] = uri3[0].split("\\.");
                StringBuilder uriFinal = new StringBuilder();
                if (uri3[0].contains("utfpr.edu.br")) {
                    utfprUri = true;
                    uriFinal.append("\"").append(uri3[0]).append("\"");
                    linhaARFF.append(uriFinal).append(SEPARADOR);
                } else if (uri4[uri4.length - 1].matches("[0-9]*")) {
                    uriFinal.append("\"").append(uri3[0]).append("\"");
                    linhaARFF.append(uriFinal).append(SEPARADOR);
                } else {
                    if (uri4.length >= 3) {
                        if (uri4[uri4.length - 1].equals("com") || uri4[uri4.length - 1].equals("org") || uri4[uri4.length -
1].equals("net")) {
                            uriFinal.append("\"").append(uri4[uri4.length - 2]).append(".").append(uri4[uri4.length -
1]).append("\"");
                            linhaARFF.append(uriFinal).append(SEPARADOR);
                        } else {
                            uriFinal.append("\"").append(uri4[uri4.length - 3]).append(".").append(uri4[uri4.length -
2]).append(".").append(uri4[uri4.length - 1]).append("\"");
                            linhaARFF.append(uriFinal).append(SEPARADOR);
                        }
                    } else if (uri4.length == 2) {
                        uriFinal.append("\"").append(uri4[uri4.length - 2]).append(".").append(uri4[uri4.length -
1]).append("\"");
                        linhaARFF.append(uriFinal).append(SEPARADOR);
                    } else { descartar = true; }
                }
                st.nextToken();
                st.nextToken();
                String contentTypeStr = st.nextToken();
                String contentType[] = null;
                contentTypeStr = contentTypeStr.replace("\"", "");
                if (contentTypeStr.split("/").length > 1) {
                    contentType = contentTypeStr.split("/");
                }
                boolean download = false;
                if (contentType != null) {
                    if ((resultGroup.equals("success"))
                            && ((size > 1000000)
                            || (contentType[0].equals("application") && size > 100000)
                            || (contentType[0].equals("video"))
                            || (contentType[0].equals("audio"))
                            || (contentType[1].equals("octet-stream")))) {
                        linhaARFF.append("download");
                        download = true;
                    } else {
                        linhaARFF.append("comum");
                    }
                } else {
                    linhaARFF.append("comum");
                }
```

```
                if (!descartar) {
                    if (!download) {
                        if (uriHashMap.containsKey(uriFinal.toString())) {
                            uriHashMap.put(uriFinal.toString(), uriHashMap.get(uriFinal.toString()) + 1);
                        } else {
                            uriHashMap.put(uriFinal.toString(), 1);
                        }
                    }
                } else { linhasIgnoradas++; }
                linhaARFF = null;
            } else {
                System.err.println("Linha ignorada: " + linhasArquivo + " + 1. Nr de tokens da linha diferente de
10. >> " + linhaCrua);
                linhasIgnoradas++;
            }
            linhasArquivo++;
            if ((linhasArquivo % 100000) == 0) { System.out.println(linhasArquivo); }
        }
        System.out.println("Linhas ignoradas: " + linhasIgnoradas);
        System.out.println("Linhas Arquivo: " + linhasArquivo);
        linhasTotais = linhasTotais + linhasArquivo;
        System.out.println("Linhas Totais: " + linhasTotais);
        linhasArquivo = 0;
        linhasIgnoradas = 0;
    }
    int qtdMaxClasses = 25;
    for (int i = 0; i < qtdMaxClasses; i++) {
        Iterator it = uriHashMap.entrySet().iterator();
        Map.Entry entradaUriHashMap = (Map.Entry) it.next();
        Integer maiorValor = (Integer) entradaUriHashMap.getValue();
        String maiorKey = (String) entradaUriHashMap.getKey();
        while (it.hasNext()) {
            entradaUriHashMap = (Map.Entry) it.next();
            Integer a = (Integer) entradaUriHashMap.getValue();
            if (a > maiorValor) {
                maiorValor = (Integer) entradaUriHashMap.getValue();
                maiorKey = (String) entradaUriHashMap.getKey();
            }
        }
        uriHashMap.remove(maiorKey);
        listaClassesUriMaisUsadas.put(maiorKey, maiorValor);
    }
    System.out.println(listaClassesUriMaisUsadas.toString());
  }

  public void processar() throws Exception, IOException {
    Path pathOutput = Paths.get(OUTPUT_FILE_ARFF);
    Path pathOutputClassesURI = Paths.get(OUTPUT_FILE_CLASSES_URI);
    OutputStreamWriter writer = new OutputStreamWriter(new FileOutputStream(pathOutput.toString(),
false), ENCODING.toString());
    BufferedWriter bw = new BufferedWriter(writer);
    OutputStreamWriter writer2 = new OutputStreamWriter(new
FileOutputStream(pathOutputClassesURI.toString(), false), ENCODING.toString());
    BufferedWriter bwClassesURI = new BufferedWriter(writer2);
    bw.write("@RELATION LOGSQUID");
    bw.newLine();
    bw.newLine();
    bw.write("@ATTRIBUTE diaSemana {1,2,3,4,5,6,7}");
    bw.newLine();
    bw.write("@ATTRIBUTE horaCheia {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23}");
    bw.newLine();
    bw.write("@ATTRIBUTE ipGroup {aluno, servidor, wireless}");
    bw.newLine();
    bw.write("@ATTRIBUTE uri {}");
    bw.newLine();
    bw.write("@ATTRIBUTE download {comum, download}");
    bw.newLine();
    bw.newLine();
    bw.write("@DATA");
    bw.newLine();
    System.out.println("Arquivo de log: " + pathInput.toString());
```

```
        System.out.println("Arquivo ARFF: " + pathOutput.toString());
        try (Scanner scanner = new Scanner(pathInput, ENCODING.name())) {
            while (scanner.hasNextLine()) {
                linhaCrua = null;
                linhaCrua = scanner.nextLine();
                StringTokenizer st = new StringTokenizer(linhaCrua, " ");
                descartar = false;
                utfprUri = false;
                if (st.countTokens() != 10) { descartar = true; }
                if (!descartar) {
                    linhaARFF = new StringBuilder();
                    Calendar cal = Calendar.getInstance();
                    cal.setTimeZone(TimeZone.getTimeZone("UTC"));
                    cal.setTimeInMillis(Long.parseLong(st.nextToken().substring(0, 10)) * 1000);
                    linhaARFF.append(cal.get(Calendar.DAY_OF_WEEK)).append(SEPARADOR);
                    linhaARFF.append(cal.get(Calendar.HOUR_OF_DAY)).append(SEPARADOR);
                    st.nextToken();
                    String clientGroupStr = st.nextToken();
                    if (clientGroupStr.split("\\.").length == 4) {
                        String clientGroup[] = clientGroupStr.split("\\.");
                        String a = clientGroup[0] + "." + clientGroup[1];
                        if (a.equals("172.27")) {
                            linhaARFF.append("wireless").append(SEPARADOR);
                        } else if (a.equals("192.168")) {
                            Integer b = Integer.parseInt(clientGroup[2]);
                            if ((b >= 0) && (b <= 7)) {
                                linhaARFF.append("servidor").append(SEPARADOR);
                            } else if ((b == 8) || ((b >= 13) && (b <= 18)) || ((b >= 24) && (b <= 26))) {
                                linhaARFF.append("aluno").append(SEPARADOR);
                            } else { descartar = true; }
                        } else { descartar = true; }
                    } else { descartar = true; }
                    String actionCodeStr = st.nextToken();
                    String resultGroup = null;
                    if (actionCodeStr.split("/").length > 1) {
                        String actionCode[] = actionCodeStr.split("/");
                        if (actionCode[1].charAt(0) == '2') {
                            resultGroup = "success";
                        } else {
                            resultGroup = "other";
                            descartar = true;
                        }
                    } else { descartar = true; }
                    Long size = Long.parseLong(st.nextToken());
                    st.nextToken();
                    String uriStr = st.nextToken();
                    int uriPosInicial = 0;
                    String uri[] = uriStr.split("://");
                    if (uri.length > 1) {
                        uriPosInicial = 1;
                    }
                    String uri2[] = uri[uriPosInicial].split("/");
                    String uri3[] = uri2[0].split(":");
                    String uri4[] = uri3[0].split("\\.");
                    StringBuilder uriFinal = new StringBuilder();
                    if (uri3[0].contains("utfpr.edu.br")) {
                        utfprUri = true;
                        uriFinal.append("\"").append(uri3[0]).append("\"");
                        linhaARFF.append(uriFinal);
                    } else if (uri4[uri4.length - 1].matches("[0-9]*")) {
                        uriFinal.append("\"").append(uri3[0]).append("\"");
                        linhaARFF.append(uriFinal);
                    } else {
                        if (uri4.length >= 3) {
                            if (uri4[uri4.length - 1].equals("com") || uri4[uri4.length - 1].equals("org") || uri4[uri4.length -
1].equals("net")) {
                                uriFinal.append("\"").append(uri4[uri4.length - 2]).append(".").append(uri4[uri4.length -
1]).append("\"");
                                linhaARFF.append(uriFinal);
                            } else {
                                uriFinal.append("\"").append(uri4[uri4.length - 3]).append(".").append(uri4[uri4.length -
```

```
2]).append(".").append(uri4[uri4.length - 1]).append("\"");
                        linhaARFF.append(uriFinal);
                    }
                } else if (uri4.length == 2) {
                    uriFinal.append("\"").append(uri4[uri4.length - 2]).append(".").append(uri4[uri4.length -
1]).append("\"");
                    linhaARFF.append(uriFinal);
                } else {
                    descartar = true;
                }
            }
            if (listaClassesUriMaisUsadas.get(uriFinal.toString()) != null) {
                if (!listaURI.contains(uriFinal.toString())) {
                    listaURI.add(uriFinal.toString());
                }
            } else {
                descartar = true;
            }
            st.nextToken();
            st.nextToken();
            String contentTypeStr = st.nextToken();
            String contentType[] = null;
            contentTypeStr = contentTypeStr.replace("\"", "");
            if (contentTypeStr.split("/").length > 1) {
                contentType = contentTypeStr.split("/");
            }
            boolean download = false;
            if (contentType != null) {
                if ((resultGroup.equals("success"))
                        && ((size > 1000000)
                        || (contentType[0].equals("application") && size > 100000)
                        || (contentType[0].equals("video"))
                        || (contentType[0].equals("audio"))
                        || (contentType[1].equals("octet-stream")))) {
                    linhaARFF.append("download");
                    download = true;
                } else {
                    linhaARFF.append("comum");
                }
            } else {
                linhaARFF.append("comum");
            }
            try {
                if (!descartar) {
                    if (download) {
                        bw.write(linhaARFF.toString());
                        bw.newLine();
                    }
                } else {
                    linhasIgnoradas++;
                }
                linhaARFF = null;
            } catch (Exception e) {
                throw new Exception("Erro ao escrever nova linha: " + e.getMessage());
            }
        } else {
            System.err.println("Linha ignorada: " + linhasArquivo + " + 1. Nr de tokens da linha diferente de
10. >> " + linhaCrua);
            linhasIgnoradas++;
        }
        linhasArquivo++;
        if ((linhasArquivo % 100000) == 0) {
            System.out.println(linhasArquivo);
        }
    }
    System.out.println("Linhas ignoradas: " + linhasIgnoradas);
    System.out.println("Linhas Arquivo: " + linhasArquivo);
    linhasTotais = linhasTotais + linhasArquivo;
    System.out.println("Linhas Totais: " + linhasTotais);
    linhasArquivo = 0;
    linhasIgnoradas = 0;
```

```
        }
        bw.close();
        bwClassesURI.write(listaURI.toString());
        bwClassesURI.newLine();
        bwClassesURI.close();
    }
    public static void main(String[] args) {
        try {
            ProcessadorTCC p = new ProcessadorTCC();
            p.preprocessar();
            p.processar();
        } catch (Exception ex) { System.out.println("Erro: "+ex.getMessage()); }
    }
}
```