UNIVERSIDADE DO SAGRADO CORAÇÃO

KELVIN FERRAZ DA SILVA

BLOQUEIO DE SITES COM CONTEÚDOS IMPRÓPRIOS UTILIZANDO INTELIGÊNCIA ARTIFICIAL

BAURU

2016

KELVIN FERRAZ DAS SILVA

BLOQUEIO DE SITES COM CONTEÚDOS IMPRÓPRIOS UTILIZANDO INTELIGÊNCIA ARTIFICIAL

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, sob orientação do Prof. Me Patrick Pedreira Silva.

BAURU
2016

Silva, Kelvin Ferraz da

S5861b

Bloqueio de sites com conteúdos impróprios utilizando inteligência artificial/Kelvin Ferraz da Silva. -- 2016.

68f. : il.

Orientador: Prof. M.ePatrick Pedreira Silva.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1.Inteligência Artificial. 2. Linux.3.Mineração de Dados. 4. Weka. 5. Squid.I. Silva, Patrick Pedreira. II. Título.

KELVIN FERRAZ DA SILVA

BLOQUEIO DE SITES COM CONTEÚDOS IMPRÓPRIOS UTILIZANDO INTELIGÊNCIA ARTIFICIAL

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, sob orientação do Prof. Me Patrick Pedreira Silva.

Banca Examinadora:

_____

Prof. Me. Patrick Pedreira Silva
Universidade do Sagrado Coração

_____

Prof. Dr. Elvio Gilberto da Silva
Universidade do Sagrado Coração

_____

Prof. Me. Henrique Pachioni Martins
Universidade do Sagrado Coração

Bauru, 13 de Dezembro de 2016.

Dedico este trabalho, aos meus pais que me apoiaram e continuam me apoiando no decorrer de minha vida.

AGRADECIMENTOS

Agradeço aos meus pais, pelo amor, incentivo e apoio incondicional. A esta universidade e seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança e mérito e ética aqui presentes.

Ao meu orientador Patrick Pedreira Silva, pelo suporte no pouco tempo que lhe coube, pelas suas correções e implementações de ideias, no qual possui amplo conhecimento admirável no assunto aqui tratado.

A minha namorada Bruna Fernandes Antunes, por me apoiar e estar ao meu lado durante a graduação, com seus conselhos e amizade sempre presente.

E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

RESUMO

O acesso à informação atualmente se dá de forma rápida e ininterrupta, os sistemas de segurança atuais estão sempre inovando, seja na parte física ou na parte lógica da informação. Tais informações são acessadas constantemente na internet, seja ela de uma empresa, escola, universidade, comércio dentre outros, desse modo, sempre irá existir o risco de que certas atividades e conteúdos ilegais ou impróprios sejam acessados, de forma que o administrador da rede não tenha um controle tão eficaz 24h do seu dia. Com esse crescimento de usuários conectados, aumenta drasticamente o risco a empresas e ou instituições serem atacadas por algum tipo de vírus, o qual passa despercebido por acesso a conteúdo impróprio em uma rede. Atualmente, em muitas empresas, essa filtragem de conteúdo é realizada por um administrador de rede de forma manual por um humano especialista. Isso acarreta em custos para as organizações, incluindo a possibilidade de haver falhas nessa filtragem. Este quadro de insegurança reforça a necessidade de se utilizar métodos eficazes e praticamente em tempo real para detectar certos tipos de acessos a uma rede ampla como a internet. Assim, torna-se muito útil e de suma importância para a segurança da informação a utilização de aplicações que consigam identificar e bloquear, de forma automática, páginas de internet que possuam conteúdos impróprios (e potencialmente perigosos). Nesse sentido a utilização de Inteligência Artificial (IA) se torna muito profícua no quesito de aplicação de reconhecimento de padrões e mineração de dados, auxiliando na filtragem e identificação de conteúdo. Neste contexto, o objetivo dessa pesquisa é avaliar comparativamente diversos métodos de Aprendizado de Máquina, com foco em aprendizado supervisionado (classificação), na tarefa de gerenciamento do bloqueio de conteúdos impróprios acessados na internet. A conclusão do método realizado no presente trabalho foi considerada inviável a aplicação, tendo em vista que a quantidade de dados utilizados para realizar o bloqueio influenciou na forma de como cada usuário realiza a navegação bloqueando praticamente todo os acessos a sites da rede, a aplicação desse método pode ser melhorada e adaptada a diferentes situações e testes.

Palavras-chave: Inteligência Artificial, Mineração de Dados, Linux, Weka.

ABSTRACT

Access to information currently takes place quickly and uninterrupted, current security systems are always innovating, whether in the physical part or in the logical part of the information. Such information is constantly accessed on the internet, be it from a company, school, university, commerce among others, in this way, there will always be a risk that certain illegal and inappropriate activities and contents will be accessed, so that the network administrator does not Have 24 hours of effective control of your day. With this growth of connected users, drastically increases the risk to companies and institutions to be attacked by some type of virus, which goes unnoticed by access to inappropriate content in a network. Currently, in many companies, this content filtering is performed by a network administrator manually by a human expert. This entails costs for organizations, including the potential for filtering failures. This picture of insecurity reinforces the need to use efficient and practically real-time methods to detect certain types of access to a broad network such as the internet. It is therefore very useful and extremely important for information security to use applications that can automatically identify and block internet pages that contain inappropriate (and potentially dangerous) content. In this sense, the use of Artificial Intelligence (AI) becomes very profitable in the application of pattern recognition and data mining, helping in the filtering and identification of content. In this context, the objective of this research is to comparatively evaluate several methods of Machine Learning, focusing on supervised learning (classification), in the task of managing the blockage of inappropriate content accessed on the internet. The conclusion of the method carried out in this work was considered unviable, since the amount of data used to perform the blocking influenced how each user performs the navigation blocking almost all access to network sites, the application This method can be improved and adapted to different situations and tests.

Keywords: Artificial Intelligence, Data Mining, Linux, Weka.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O acesso à informação atualmente se dá de forma rápida interrupta, os sistemas de segurança atuais estão sempre inovando, seja na parte física ou na parte lógica da informação. Tais informações são acessadas constantemente na internet, seja ela de uma empresa, escola, universidade, comércio dentre outros, desse modo, sempre irá existir o risco de que certas atividades e conteúdos ilegais ou impróprios sejam acessados, de forma que o administrador da rede não tenha um controle tão eficaz 24h do seu dia.

Segundo o relatório da Trends in Telecommunication Reform 2015, a internet terá cerca de 3,2 bilhões de pessoas conectadas até 2019, apesar de o número ser alto, presume-se que ainda existirão mais de 40 milhões de usuários desconectados da internet (GLOBO, 2015). Com esse crescimento de usuários conectados, aumenta drasticamente o risco a empresas e ou instituições serem atacadas por algum tipo de vírus, o qual passa despercebido por acessar a conteúdos impróprio em uma rede.

Depois de investigar, a equipe de TI do departamento de inteligência dos EUA, descobriu que a imprudência dos usuários abriu a porta para softwares mal-intencionados que se instalaram na rede (BLUEPEX, 2015). Atualmente, em muitas empresas, essa filtragem de conteúdo é realizada por um administrador de rede de forma manual por um humano especialista. Isso acarreta em custos para as organizações, incluindo a possibilidade de haver falhas nessa filtragem.

Este quadro de insegurança reforça a necessidade de se utilizar métodos eficazes e praticamente em tempo real para detectar certos tipos de acessos a uma rede ampla como a internet. Assim, torna-se muito útil e de suma importância para a segurança da informação a utilização de aplicações que consigam identificar e bloquear, de forma automática, páginas de internet que possuam conteúdos impróprios (e potencialmente perigosos).

Processar conteúdos e classificá-los em próprios ou impróprios é uma tarefa relativamente simples para seres humanos, porém, à medida que o volume de documentos cresce e, ainda que o homem seja o melhor “recurso”, a quantidade de documentos torna a realização dessa tarefa, de modo manual, impraticável,

deixando evidente a necessidade do uso do computador. Devido a isso, este processo de análise automática de informações é um tema muito pesquisado e relevante. Entretanto, a realização automática dessa atividade pode exigir custosos recursos computacionais. Para viabilizar tal tarefa, teorias envolvendo Inteligência Artificial e suas subáreas tais como Processamento de Linguagem Natural e Aprendizado de Máquina, apresentam-se como alternativas. Segundo Luger (2014, p. 1) “A Inteligência Artificial pode ser definida como o ramo da Ciência da Computação que se ocupa da automação do comportamento inteligente”. O Processamento de Linguagem Natural é uma subárea da IA e da Linguística que estuda os problemas da geração e compreensão automática de línguas humanas naturais. Já o Aprendizado de Máquina (ou "machine learning") objetiva programar computadores para aprender um determinado comportamento ou padrão automaticamente a partir de exemplos ou observações. Nesse sentido a utilização de Inteligência Artificial (IA) se torna muito profícua no quesito de aplicação de reconhecimento de padrões e mineração de dados, auxiliando na filtragem e identificação de conteúdo.

Neste contexto, o objetivo dessa pesquisa é avaliar comparativamente diversos métodos de Aprendizado de Máquina, com foco em aprendizado supervisionado (classificação), na tarefa de gerenciamento do bloqueio de conteúdos impróprios acessados na internet.

2 OBJETIVOS

A seguir serão explicitados os objetivos do trabalho.

2.1 OBJETIVO GERAL

Avaliar comparativamente diversos métodos de Aprendizado de Máquina, com foco em aprendizado supervisionado (classificação), na tarefa de gerenciamento do bloqueio de conteúdos impróprios acessados na internet.

2.2 OBJETIVOS ESPECÍFICOS

a) Realizar um levantamento bibliográfico dos algoritmos de aprendizado de máquina a serem utilizados;

b) Integrar os algoritmos de melhor desempenho a um servidor de internet com sistema operacional Linux configurado com o Sarg e Squid;

c) Testar a aplicação em funcionamento e avaliar os resultados obtidos.·.

d) Aplicar os algoritmos, utilizando a ferramenta Weka

e) Integrar os algoritmos de melhor desempenho a um servidor de internet com sistema operacional Linux configurando com o Sarg e squid, visando criar uma lista de bloqueio na base dos dados coletados.

3 INTELIGÊNCIA ARTIFICIAL

A Inteligência artificial é o ramo da ciência computacional no qual se ocupa da automação do comportamento inteligente.

Russel e Norving (2004, p. 4) descrevem que,

> Durante milhares de anos, procuramos entender como pensamos; isto é, como um mero punhado de matéria pode perceber compreender, prever e manipular um mundo com maior e mais complicado que ela própria. O campo da I.A vai além: ele tenta não apenas compreender, mas também construir entidades inteligentes.

Segundo Nikolopoulos (1997) a Inteligência Artificial é um campo de estudos multidisciplinar, originado da computação, da engenharia, da psicologia, da matemática e da cibernética, cujo principal objetivo é construir sistemas que apresentem comportamento inteligente e desempenhem tarefas com um grau de competência equivalente ou superior ao grau com que um especialista humano as desempenharia.

Para Haykin (2001, p. 59), “um sistema de IA deve ser capaz de fazer três coisas: (1) armazenar conhecimento, (2) aplicar o conhecimento armazenado e (3) adquirir novo conhecimento”.

Segundo Araribóia (1988, p. 1), a “IA é o ramo do conhecimento que trata entre outras coisas de um projeto de computador e robôs inteligentes. Um computador é inteligente se possuí qualquer uma das habilidades mentais que fazem uma pessoa”.

Inteligência Artificial envolve utilizar métodos baseados no comportamento inteligente de humanos e outros animais para solucionar problemas complexos. (COPPIN, 2004).

Segundo Coppin (2004, p. 5) “Pode-se já certamente definir inteligência artificial pelas propriedades que ela exibe uma capacidade de lidar com novas situações; a capacidade de solucionar problemas, responder questões, de engendrar planos e assim exibidos por humanos”.

É provável que a IA torne-se mais predominantemente na nossa sociedade. E quer se crie ou não finalmente uma IA que seja verdadeiramente inteligente, provavelmente encontraremos computadores, máquinas e outros objetos parecendo ser mais inteligentes, pelo menos em modo que se comportam (COPPIN, 2013).

3.1 HISTÓRIA DA INTELIGÊNCIA ARTIFICIAL

De acordo com Luger (2014, p. 4),

> Os fundamentos filosóficos do trabalho moderno em inteligência artificial já haviam sido desenvolvidos durante milhares de anos [.] O ponto de partida lógico para história é o gênio Aristóteles [...].” Aristóteles reuniu descobertas, as maravilhas e os temores da antiga tradição grega com análise cuidadosa e o pensamento disciplinado que se tornaram o padrão para a ciência moderna.

A (IA) é uma das ciências mais recentes. O trabalho começou logo após a Segunda Guerra Mundial (com métodos ligados às Redes Neurais) e o próprio nome foi cunhado em 1956. “Juntamente com a biologia molecular, a IA é citada regularmente como “o campo em que eu mais gostaria de estar” por cientistas de outras disciplinas” (RUSSEL; NORVIG 2004, p. 4).

Esses métodos de Redes Neurais foram aprimorados mais tarde com Donald Hebb que desenvolveu algoritmos para manipulação das conexões dos neurônios. Tal metodologia foi chamada de Aprendizagem de Hebb (ou Teoria Hebbiana) e ainda tem influência até os dias de hoje (RUSSEL; NORVIG, 2004).

Na década de 50, Marvin Minsky construiu o primeiro computador neural usando 3.000 tubos a vácuo (essa máquina foi chamada de SNARC). Minsky expandiu seus estudos que, até nos dias atuais, possuem grande importância (RUSSEL; NORVIG, 2004).

Segundo Faceli et. al. (2011), a partir da década de 70, houve uma maior disseminação do uso de técnicas de computação baseadas em IA para a solução de problemas reais.

A ideia de inteligência na via clássica foi concebida sobre a tese de Alan Turing, proposta em 1950, que teve avanço nos anos 30 ao redor de máquinas que são capazes de jogar, raciocinar, de aprender, de planejar, de perceber, de formular conceitos, de utilizar uma língua natural e de pensar criativamente (COELHO, 1995).

Nas Décadas mais recentes, o estudo da Inteligência Artificial floresceu. Áreas de particular importância são as seguintes:

a) **Aprendizado de Máquina:** aprendizado está diretamente ligado com a inteligência, pois realmente se um sistema é capaz de

aprender a exercer determinada tarefa mereça então ser chamado de inteligente;

b) **Sistemas Multiagente:** compostos por múltiplos agentes são sistemas que interagem atonamente, mas também se interagem com outros agentes presentes;

c) **Vida Artificial:** sistemas que sejam definidos de um modo simples e consigam produzir seu próprio comportamento, que pode ser peculiarmente complexo. (COPPIN, 2004);

d) **Visão Computacional:** modo de como os sistemas computacionais enxerga;

e) **Planejamento:** Um planejador tem como estado inicial certo objetivo, para atingir esse objetivo o planejador desenvolve um plano e o executa. (COPPIN, 2004);

f) **Jogos:** Implementação de IA em jogos, como por exemplo, jogo de xadrez.

3.1.1 Aprendizado de Máquina (AM)

De acordo com Faceli et. al. (2011), nas últimas décadas, com a crescente complexidade dos problemas a serem tratados computacionalmente e do volume de dados gerados por diferentes setores, tornou-se clara a necessidade de ferramentas computacionais mais sofisticadas.

Ainda segundo Faceli et. al. (2011), um exemplo simples é a descoberta da hipótese na forma de uma regra ou conjunto de regras para definir clientes de um supermercado devem receber material de propaganda de um novo produto, utilizando para isso dado de compras passadas dos clientes cadastrados na base de dados do supermercado.

Para Coppin (2013), aprendizado de máquina (AM) é um segmento extremamente importante na IA. Ainda segundo ele na maioria dos problemas de aprendizado, a tarefa é aprender a classificar entradas de acordo com um conjunto

finito (ou, às vezes, infinito) de classificações. Tipicamente, um sistema de aprendizado é dotado de um conjunto de treinamento, cujos dados foram classificados manualmente.

O “Aprendizado de máquina é uma área de IA cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática” (REZENDE, 2005, p. 89).

Em AM, computadores são programados para aprender com à experiência passada. Para tal, empregam um princípio de inferência denominado indução, no qual se obtêm conclusões genéricas a partir de um conjunto particular de exemplos (FACELI, 2011).

Ainda segundo Faceli et. al. (2011) existem várias aplicações bem-sucedidas de técnicas de AM na solução de problemas reais, entre as quais podem ser citadas:

3.1.2 Redes Neurais

Para Haykin (2001, p. 28),

> ”uma rede neural é uma máquina que é projetada para modelar a maneira como o cérebro realiza uma tarefa particular ou função de interesse; a rede é normalmente implementada utilizando-se componentes eletrônicos ou é simulada por programação em um computador digital.”

Segundo Coppin (2004), redes neuras, são neurônios modelados em relação ao cérebro humano e são compostas por vários neurônios artificiais.

A história das Redes Neuras pode ser resumida nos acontecimentos cronológicos seguintes:

a) **Anos 40**: Durante a Segunda Guerra mundial, houve a necessidade do avanço tecnológico, para serem utilizados em combate bélico. Todo investimento em áreas cientificas estava à disposição de se desenvolver mecanismos para morte em massa;
b) **Anos 50:** Época de iniciação de estudos e utilização da lógica de estratégias para fins matemáticos. Foram desses aperfeiçoamentos que surgiram, jogos, aplicações matemáticas e simuladoras (GANGORA, 2015);

c) **Anos 60:** Agora envolvido em linha biológica, os desenvolvimentos continuaram de conceitos relativos às redes neurais artificiais com aprimoramento do modelo Perceptron e o surgimento de sua variante a Adaline (GANGORA, 2015);
d) **Anos 70:** Com várias fragilidades apontadas nos modelos de redes neurais, poucos investimentos foram realizados nessa área (GANGORA, 2015);
e) **Anos 80:** Após anos com poucos avanços com as redes neurais, seu reconhecimento é retomado graças ao físico John Hopfield, que provou ser possível à simulação de um sistema físico através de um modelo matemático (GANGORA, 2015).
f) **Anos 90:** O ano que redes neurais tiveram diversos lançamentos e aplicações. Centenas de propostas de novos aperfeiçoamentos de modelos a cada ano. Dessa forma começaram a se construir os chamados Sistemas Híbridos (GANGORA, 2015).

3.1.2.1 Multilayer Perceptron (MLP)

"Quando utilizamos apenas redes neurais que possuem apenas uma camada, são sempre utilizados os padrões na entrada e saída dessa rede. Perceptron demonstram que não sejam linearmente separáveis" (FERNANDES. 2005. p. 76).

De acordo com Fernandes (2005), foi desenvolvido o algoritmo de treinamento **backpropagation (retropropagação),** por Rumelhart, Hinton e Williams em 1986 precedido por propostas semelhantes ocorridas nos anos 70 e 80, mostrou que é possível realizar esse treinamento com juntamente com a rede MLP. Nesse tipo de rede cada camada tem uma função específica, a camada de saída recebe certos estímulos da camada intermediária, construindo o padrão que será a resposta.

"Basicamente a aprendizagem por retropropagação de erro consiste de dois passos através das camadas diferentes da rede: um passo para frente, a propagação, e um passo para trás, a retropropagação." (HAYKIN, SIMON. 2001. p. 183).

De acordo com Haykin (2001), um Perceptron de múltiplas camadas tem três características distintivas:

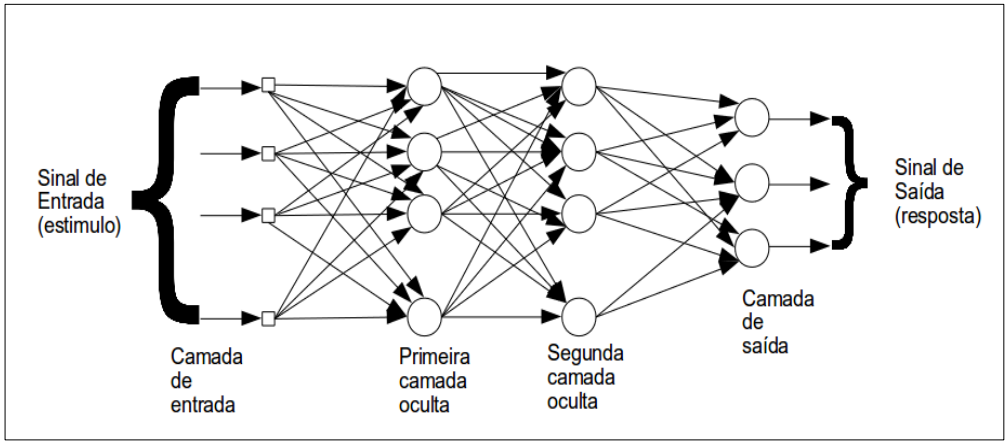
a) **Primeira:** Cada neurônio da rede possui uma função de ativação não linear;

b) **Segunda:** Possui uma ou mais camada de neurônios ocultos, que, não são parte da entrada ou da saída da rede;

c) **Terceira:** A rede neural exibe um alto grau de conectividade, no qual são determinados pela rede. Uma modificação na conectividade da rede requer mudança na população das conexões sinápticas ou de seus pesos.

A Figura 1 mostra o grafo arquitetural de um neurônio Perceptron de Múltiplas Camadas (MLP), possuindo duas camadas ocultas e uma camada de saída.

Figura 1: Grafo do Perceptron Múltiplas Camadas



Fonte: Haykin (2001, p. 186).

3.1.2.2 ZeroR

O método ZeroR é o modelo de classificação mais simples que se baseia no alvo e ignora todos os preditores. O classificador ZeroR simplesmente prevê a maioria das categorias (classe). Embora não se pode definir corretamente previsibilidade em ZeroR, é muito útil para determinar um desempenho de linha base como referência para outros métodos de classificação.

3.1.2.3 Árvore de Decisão

De acordo com Gonzalez (2014, p. 56):

> Numa Árvore de Decisão cada atributo é representado por um nó de decisão, cuja função é testar o valor desse atributo. Uma classe é representada por um nó folha, que reúne todos os Exemplos que chegarem a ele depois de satisfazerem os testes dos nós de decisão intermediários. Portanto, numa Árvore de Decisão, a classificação de um Exemplo desconhecido implica percorrer toda a árvore a partir de um nó raiz, testando atributos em sucessivos nós internos até chegar a um nó folha, que lhe atribuirá urna classe. O objetivo de urna Árvore de Decisão é retomar uma classe para um Exemplo desconhecido.

Para Faceli (Katti et al. 2011), uma árvore de decisão usa a estratégia de dividir para conquistar, para resolver um problema de decisão. Um problema complexo é divido em problemas mais simples, aos quais recursivamente é aplicada a mesma estratégia. Segundo Rich (Elaine, p. 69), “Uma maneira simples de programar qualquer estratégia de busca é a travessia de uma árvore. Cada nó da árvore é expandido pelas regras de produção para gerar um conjunto de nós sucessores”.

De acordo com Russel e Norvig (2004) árvores de decisões são a forma mais simples e bem-sucedidas de algoritmos de aprendizagem. Ainda segundo Russel e Norvig (2004. p. 633). “Uma árvore de decisão toma como entrada um objeto ou situação e retorna uma “decisão” - o valor de saída previsto, de acordo com a entrada. Os atributos de entrada podem ser discretos ou contínuos.”.

Ao configurar o problema como aprendizagem, inicia-se na declaração de um problema, no exemplo escolha da melhor decisão para ir a um local:

a) **Alternativa:** Cada neurônio da rede possui uma função de ativação não linear;

b) **Bar:** Se o restaurante tem uma área de bar que seja confortável onde possa esperar.

c) **Sex/Sáb:** Verdadeiro as sextas e sábados.

d) **Faminto:** Se há fome.

e) **Clientes:** Quantas pessoas estão no restaurante (valores dos atributos: Nenhum Alguns e Cheio).

f) **Preço:** A faixa de preço do restaurante (R$, R$R$, R$RR).

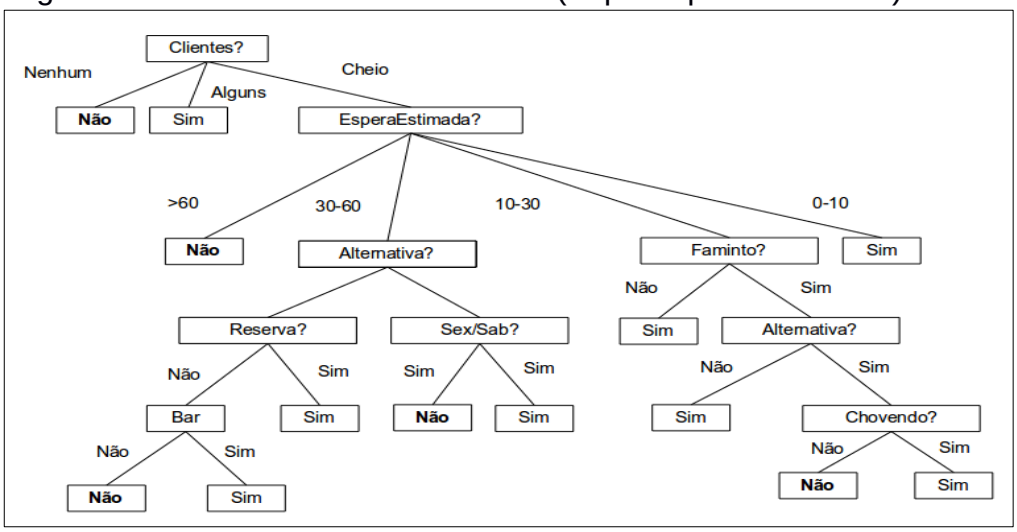
g) **Chovendo:** Se está chovendo do lado de fora.

h) **Reserva:** Se fizemos reserva.

i) **Tipo:** O tipo de restaurante (francês, italiano, tailandês ou só serve hambúrguer).

j) **EsperaEstimada:** A espera estimada pelo gerente (de 0 a 10 minutos, de 10 a 30, de 30 a 60, >60).(RUSSEL, NORVIG, 2004).

Figura 2:Resultado Árvore de decisão (Esperar por uma mesa)



Fonte: Russel e Norvig (2004. p. 634).

Na (Figura 2) a árvore de decisão não utilizou os atributos **preço** e **tipo**, pois foram considerados irrelevantes. Os exemplos são processados pela árvore a partir da raiz e seguindo sua ramificação até alcançar uma folha como, por exemplo, (Clientes = Cheio = EsperaEstimada = 0 – 10) sendo assim será classificada como positiva (sim, espera por uma mesa). (RUSSEL, NORVIG, 2004).

3.1.2.4 Classificadores Bayesianos Ingênuos

Baseado na teoria de Thomas Bayes tem a função de tratar acontecimentos de eventos, altera a estimativa inicial com uma base em novas informações, às novas informações atuam como um modificador de opinião, mostrando como fazer avaliações e como ajustá-las quando deparados com novos dados (ZEMBRZUSKI, M.C. 2010).

> Um método baseado em modelos de vetores mediante pelo cosseno, método baseado em decisões e dois métodos baseados em modelos probabilísticos de Nave-Bayes (um entrar sobre palavras e outras sobre slogan). Para cada subsistema as características incluídas não são somente um conjunto de palavras em uma janela fixa do contexto, suas análises morfológicas, etc., e que também incluem uma variedade de características sintáticas como sujeitos, objetos diretos circunstanciais e várias relações modificadores de nome e adjetivos. (ANTONÍN, 2004, p. 77).

Possui como características principais:

a) Oferecer aprendizado estatístico;

b) Ser incremental (incrementa/decrementa) a probabilidade de uma determinada hipótese ser corre;

c) Oferece predição estatística (a classificação permite predizer múltiplas hipóteses de acordo com sua probabilidade).

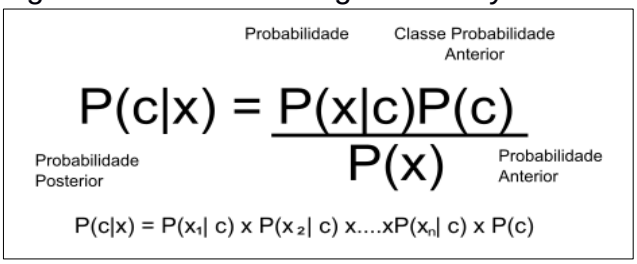
Exemplos de utilização desse tipo de classificador:

a) Classificador de textos;
b) Filtragens de spam;
c) Sistema de recomendação.

São atributos condicionalmente independentes, ou seja, o valor de um atributo não influência o valor dos outros atributos. Seu conjunto inicial de dados classifica em classes (clusters) e, a partir de um dado de entrada, busca classificar este novo dado.

É utilizado em situações nas quais os dados são demasiados, devidos sua simplicidade, a computação e o treinamento são facilitados. No entanto, estes classificadores não-ideais devido ao fato de, na maior parte dos casos, os atributos serem correlacionados. A (Figura 3) demonstra a fórmula do algoritmo.

Figura 3: Fórmula do Algoritmo Bayesiano



Fonte: Elaborada pelo autor

Para se calcular as classes mais prováveis de uma nova instância, calculam-se a probabilidade de todas possíveis classes, e no fim, escolhe-se a classe com maior probabilidade como rótulo da nova instância. Em estatística o equivalente a maximar P(classes|$X_1 \ldots X_n$).

a) P(x) e P(c) são probabilidades a priori de X e C
b) P(c|x) e P(x|c) são propriedades a posteriori de C condicional a X e a de X condicional a B respectivamente.

3.1.3 Método de avaliação de desempenho de classificadores (baseline)

Uma forma de comparar o desempenho de classificadores, além da comparação direto com outros métodos por meio de uma medida de desempenho, é confrontá-lo com um método baseline. No caso da tarefa de classificação, um método comumente usado é aquele definido pelo ZeroR. O método ZeroR é o modelo de classificação mais simples que se baseia no alvo e ignora todos os preditores. O classificador ZeroR simplesmente prevê a maioria das categorias (classe). Embora não se pode definir corretamente previsibilidade em ZeroR, é muito útil para determinar um desempenho de linha base como referência para outros métodos de classificação.

A matriz de confusão oferece uma medida efetiva do modelo de classificação, ao mostrar o número de classificações corretas versus as classificações preditas para cada classe, sobre um conjunto de exemplos. O número de acertos, para cada classe, se localiza na diagonal principal (i=j) da matriz. Os demais elementos , para i ≠ j, representam erros na classificação. A matriz de confusão de um classificador ideal possui todos esses elementos iguais a zero uma vez que ele não comete erros.

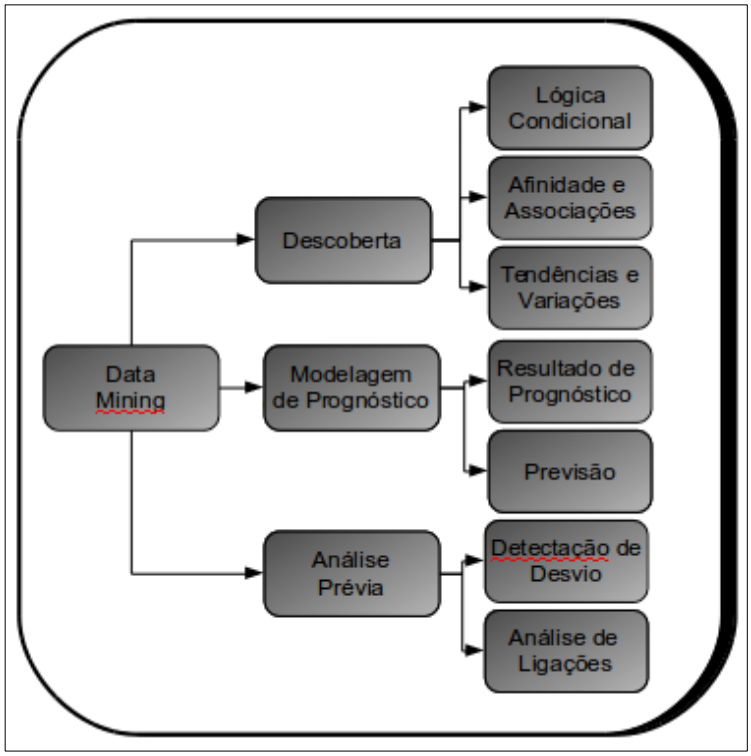
4 MINERAÇÃO DE DADOS

Segundo Figueira (1998), a tecnologia tornou relativamente fácil o acúmulo de dados, possibilitando o desenvolvimento de processos que permitam gerar conhecimento a partir da extração automática desses dados. Para Groth (1998) Data Mining é o processo de descoberta automática de informações.

Para Luger (2014), há uma série de importantes questões filosóficas e epistemológicas subjacentes à pesquisa em aprendizado. Já para Ávila (1998), Data mining é uma área de pesquisa da IA que busca encontrar padrões em bases de dados.

Polito (1997, p.28), diz que, “de uma vista de processo orientado, há três classes de data mining: descobrimento, modelagem de prognóstico e análise prévia”, como mostra na Figura 4.

Figura 4: Processo de Data Mining



Fonte: Elaborada pelo autor

Uma explicativa dos processos de Data Mining:

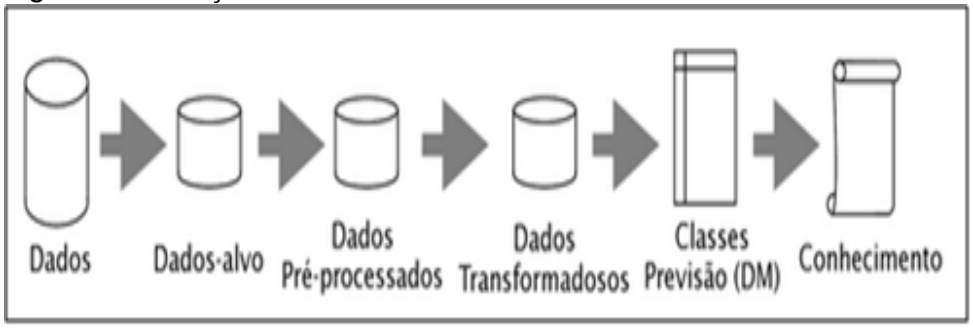
a) Descoberta: adquirir informações que podem revelar conhecimentos e estruturas, que podem levar a decisões em condições de certa limitada.
b) Análise Prévia: é o processo que analisa uma base com algo que deseja verificar se existe anomalias ou resultados raros que influenciam os resultados da mineração de dados.
c) Modelagem de prognóstico: os padrões descobertos no banco de dados são usados para prognosticar o futuro. Permitindo o usuário submeter valores desconhecidos de campos nos registros.

Data Mining é uma tecnologia usada para revelar informação estratégica escondida em grandes massas de dados (KREMER, 1999). Conforme Goldschmidt e Passos (2005, p.1).

> [...]análise de grandes quantidades de dados pelo homem é inviável sem o auxílio de ferramentas computacionais apropriadas. Portanto, torna-se imprescindível o desenvolvimento de ferramentas que auxiliem o homem, de forma automática e inteligente, na tarefa de analisar, interpretar e relacionar esses dados para que se possa desenvolver e selecionar estratégias de ação em cada contexto de aplicação[...].

Ainda segundo Goldschmidt e Passos (2005), surge uma nova área denominada Descoberta de Conhecimento de Dados (Knowledge Discovery in Data-base – KDD). O processo de descoberta de conhecimento é ilustrado na Figura 5: Esquema para geração de conhecimento em banco de dados KDD.

Figura 5: Geração de conhecimento em banco de dados KDD



Fonte: (Braga Vieira, 2005).

a) Definição dos problemas: Definir o tipo de conhecimento que se deseja extrair do banco de dados;

b) Aquisição de características e realce: Selecionar um conjunto de dados ou focalizar-se em um subconjunto de atributos;

c) Plano de prototipagem. Prototipagem e Desenvolvimento do Modelo: Realiza busca de estruturas (padrões), classificação/regressão e sua evolução;

d) Avaliação do modelo: Avalia o modelo de prototipagem se a implementação é viável;

e) Implementação: Implementa o que foi filtrado na prototipagem, para um melhor desempenho e custo beneficio;

f) Avaliação do retorno do investimento (pós-projeto): Avalia o retorno de investimento, pós o desenvolvimento do projeto.

4.1 TÉCNICAS DE DATA MINING

Segundo Figueira (1998, p.44) “muitas técnicas de Data Mining se originaram na pesquisa em I.A nas décadas de 1980 e 1990.” Para Ric Vill (1999) Data Mining, basicamente, é aplicação de técnicas estatísticas, muitas vezes complexas e que precisam ser analisadas por técnicos especializados.

De acordo com Goldschmidt e Passos (2005) existem diversos tipos de técnicas e de algoritmos de Mineração de Dados que podem ser subdivididas as quais serão apresentadas a seguir:

a)Técnicas tradicionais: É o tipo de técnica que existe independente o contexto de Mineração de Dados. Em suma produzem bons resultados também em aplicações em KDD. (RONALD; PASSOS EMMANUEL, 2005);

b)Técnicas Específicas: São especificamente para tarefas KDD, podemos criar algoritmos nos quais são desenvolvidos especificamente para a tarefa de descoberta de associação. (RONALD; PASSOS EMMANUEL, 2005);

c)Técnica Híbrida: Maneira de combinar técnicas e chamar de sistemas Híbridos. (RONALD; PASSOS EMANUEL, 2005).

4.2 PROCESSAMENTO DE LINGUAGEM NATURAL (PLN)

"O homo sapiens é separado de outras espécies pela capacidade da linguagem. Algo em torno de 100.000 anos atrás, os seres humanos aprendeu a falar e cerca de 7.000 anos atrás aprenderam a escrever". (RUSSEL ; NORVIG, 2013, p. 763).

O Processamento de Linguagem Natural (PLN) é uma subárea da IA e da Linguística que estuda os problemas da geração e compreensão automática de línguas humanas naturais.

De acordo com Kumar (2011) alguns acontecimentos marcaram a história do PLN:

a) **Primeira Era (1940 – 1950):** Neste período, o trabalho limitou-se principalmente a dois paradigmas fundamentais; ou seja, a automação e o uso de modelos probabilísticos. A automação começou em 1950 com o modelo de computação algorítmico de Turing;

b) **Segunda Era (1957 – 1970):** Na segunda era o processamento de linguagem tinha dividido de forma muito clara em dois paradigmas: simbólica e estocástica. O trabalho simbólico iniciou de duas linhas de trabalho. O primeiro foi o trabalho de Chornsky sobre teoria da linguagem formal e sintaxe generativa, e do trabalho de muitos linguístas e cientistas da computação em algoritmos de análise, inicialmente utilizando top-down e abordagens de baixo para cima e em seguida, através de programação dinâmica;

c) **Terceira Era (1970 – 1993):** O próximo período viu uma explosão na pesquisa em processamento de linguagem. Mostrou que a análise foi suficientemente bem compreendida para começar a concentrar-se na semântica e modelos de discurso. Roger Shank e seus colegas construíram uma série de programas de compreensão da linguagem que incidiu sobre o conhecimento conceitual humano, tais como roteiros, planos e objetivos;

d) **Quarta Era (1993 – até a presente data):** Nesta época, o uso de modelos probabilísticos e de dados impulsionada tornou-se bastante padrão em todo o processamento de linguagem natural. Algoritmos de análise, resolução de referência e processamento do discurso, todos passaram a incorporar probabilidades e empregar estratégias de avaliação emprestadas de reconhecimento de voz. Além disso, as inovações tecnológicas em hardware, tais como o aumento da velocidade e memória de computadores.

Segundo Coppin (2013) está se tornando cada vez mais importante que computadores sejam capazes de entender linguagens naturais.

Ainda Coppin (2013) diz que ao lidar com linguagem natural, um sistema computacional precisa ser capaz de processar e manipular linguagens de diversos níveis:

a) Fonologia: Necessário que o computador entenda a linguagem falada;

b) Morfologia: Estágio de análise que é aplicado às palavras;

c) Sintaxe: Aplica as regras da gramática da linguagem que está em utilização;

d) Programática: Entendimento de tipo humano para determinar significados que não estejam claros a partir da semântica. (COPPIN, BEN. 2013. p. 497).

Segundo Coppin (2013), existem diversos tipos de tarefas comuns sendo executadas por algum tipo de linguística, através de aplicações especializadas e voltadas para tarefas especificas do PLN:

a) Escrita e produção de um texto

b) Leitura e afolhamento

c) Tradução

d) Aprendizagem e ensino

e) Sistemas de informação

f) Sistemas interativos

g) Indexação

h) Entrada de dados

i) Segurança e identificação

Independentemente da tarefa executa, o passo inicial do processamento de um texto envolve geralmente uma etapa denominada de pré-processamento. O pré-processamento de texto tem a função de transformar o texto escrito em sua forma ortográfica em unidades fonológicas.

Segundo Braga (2005. pg. 183) “o pré-processamento de texto (ou front-end), corresponde ao processo de descodificação e de normalização de elementos presentes nos textos (tais como são os numerais, siglas, os estrangeirismo, as abreviaturas, os acrônimos e as siglas.”.

A etapa do pré-processamento vai auxiliar no processo de manipulação de um texto da melhor forma possível. Dentre as atividades de pré-processamento existentes destacam-se as seguintes:

a) **Tokenização:** É a primeira fase do pré-processamento de texto, no qual segmenta o fluxo de caracteres em tokens (palavras, números e pontuação);

b) **Lematização:** as variantes são levadas a sua forma canônica (lema), nesse caso, os verbos vão para a forma infinitiva e os adjetivos e substantivos, para masculino singular (se existir). Como exemplo as palavras, “brasileiro”, “brasileirão” apontam para o lexema “brasil”.

c) **Radicalização:** é baseada na remoção de afixos, transformando as variantes em seus radicais. Por exemplo, os termos “brasileiro”, “brasileirão” com a radicalização irão ficar “brasiler”.·.

5 REDES, APLICAÇÕES E SISTEMA OPERACIONAL

“Um software de computador pode ser divido em dois tipos: os programas de sistema, que gerenciam a Operação do computador em si, e programas aplicativos, que realizam o trabalho real desejado pelo usuário [.]”. (Tanenbaum, 2008 et al. p. 21).

Ainda para Tanenbaum (2008), o S.O tem o trabalho de fornecer aos programas de usuários um modelo de computador melhor, mais simples e mais limpo de lidar com o gerenciamento de todos os recursos possíveis.

Para Silberschatz (2010), um SO é um programa que gerencia o hardware do computador, fornecendo também como base para programas aplicativos e atua como intermediário entre o usuário e o h**ardware.**

5.1 SISTEMA OPERACIONAL UNIX

Para Soares (2010), UNIX é basicamente um sistema operacional simples, mas você precisa ser um gênio para entender a simplicidade.

Ele foi originalmente desenvolvido na década de 1970, nos Laboratórios Bell, como um sistema multitarefa e computadores de grande porte.

De acordo com Silberschatz, a primeira versão do UNIX foi desenvolvida por Ken Thompson e Dennis Ritichie, no qual se juntaram para ser usado no PDP-7 no qual seria trocado posteriormente. Ainda para Silberschatz, o nome UNIX surgiu como trocadilho para MULTICS, no qual era o projeto que Ritichie trabalhava.

De acordo com Soares, com a capacidade de o código UNIX ser reutilizado, sua popularidade aumentou, pois até então softwares eram desenvolvidos para tarefas específicas e precisavam ser totalmente reescritos para funcionar em outros computadores. Ainda para Soares, a única parte do código que precisava ser alterada para o sistema funcionar em outros computadores, era o núcleo do sistema (kernel).

De acordo com Tanenbaum, o UNIX foi desenvolvido em diversos considerados minicomputadores daquela época, além do PDP-7 que foi o precursor

do UNIX, o sistema também foi atualizado para outros microcomputadores, tais como:

a) PDP-11/20 – Bem mais moderno que o obsoleto PDP-7;
b) PDP-11/45 e PDP-11/70 – Eram máquinas poderosas com grandes memórias físicas para sua época (256kb e 2MB).

Esses computadores dominaram o mercado na década de 1970, o segundo momento do UNIX foi a qual linguagem o sistema seria produzido que seria a linguagem C. No qual passou a dominar o mercado desde então.

Ainda para Tanenbaum, no decorrer das evoluções do UNIX, surgiu com base nessas evoluções, o sistema MINIX que na verdade era o primeiro sistema do tipo UNIX baseado no projeto de um micronúcleo. Poucos meses após a aparição do MINIX, havia um grupo de discussão conhecido como USENETE (hoje em dia, Google) e mais de 40 mil usuários. O MINIX se tornou um empreendimento coletivo mantido por um grande número de usuários pela internet. Na Figura 6 imagem do PDP-7 e os criadores do UNIX.

Figura 6: PDP-7 - Unix e os criadores do Unix

Fonte: http://blog.iso50.com/wp-content/uploads/2009/01/leopard-preview-server-1-450x360.jpg

5.2 GERÊNCIA DE REDES

Segundo McCabe (2007), a gerência de rede é um processo no qual se utiliza para controlar, planejar, alocar, implantar, coordenar e monitorar recursos de uma rede. Uma rede que possui uma gerência adequada é fundamental para o sucesso.

De acordo com Kurose e Keith (2006), a rede é composta por diversos ativos complexos (hardware e software), sejam compostos físicos, enlaces, protocolos todos entregam entre si. Todo componente de rede demanda uma organização e uma gerência para o seu melhor funcionamento. Segundo McCabe (2007), os recursos de uma rede dependem de sua integridade e a mesma deve ser mensurada e verificada constantemente.

5.3 PROXY

Segundo Marcelo (2006), o acesso a internet cada vez mais se encontra presente em organizações tornando um recurso crítico sendo mais acessado e evidente.

“O conceito de Proxy refere-se a um software que atua como uma aplicação entre o serviço a ser acessado, interpretando as requisições e repassando-as ao servidor de destino.” (RICCI; MENDONÇA, 2006, p. 1).

5.4 SISTEMA OPERACIONAL LINUX

De acordo com Tanenbaum (2008), o Linux foi desenvolvido por Linus Torvalds, no qual decidiu escrever um cópia do UNIX, denominando assim como Linux, a idéia do desenvolvimento, seria um sistema de produção completo, com muitas características que faltavam (inicialmente) no MINIX. A primeira versão do Linux (0.01) foi liberada em 1991, o sistema foi desenvolvido em modo cross-developed (modo cruzado), em uma máquina MINIX, tendo alterações desde sua estrutura da árvore da fonte até o layout do sistema de arquivos.

Para Soares (2010), o Linux foi desenvolvido em conformidade com o POSIX, que era um padrão para construção de sistemas operacionais criado para normalizar

o UNIX. De acordo com Soares, o Linux é desenvolvido cooperativamente e mantido sobre a licença GNU e por milhões de usuários e empresas espalhadas pelo mundo.

Para Soares (2010), a ideia do projeto consiste na construção de um S.O como o UNIX, mas contendo somente programas livres suficientes para subsistirem qualquer sistema não livre.

De diversas opiniões apresentadas, Silberschatz (2010, p. 429) afirma que, “O sistema Linux como um todo é mantido por uma rede de desenvolvedores que colaboram através da Internet, com pequenos grupos ou indivíduos tendo a responsabilidade de manter a integridade de componentes físicos[..]”.

De acordo com Tanenbaum (2009, p. 447) diz que, “Linux cresceu rapidamente de tamanho e evoluiu para um completo clone UNIX de produção quando a memória virtual, um sistema de arquivos mais sofisticado e muitas outras características lhe foram adicionados […].”.

5.4.1 Distribuição Debian

De acordo com Hertzog (2015), Debian é um sistema operacional completo, sua ideia de desenvolvimento era lançar um S.O com digna de um kernel Linux e também uma distribuição não comercial, suficientemente credível para competir com grandes distribuições comerciais.

5.4.2 Ferramenta WEKA

Weka foi desenvolvido na Universidade de Waikato, na Nova Zelândia, o seu nome significa Waikam Ambiente para a análise do conhecimento. O programa é desenvolvido na linguagem Java, e distribuídos sobe os termos da Licença GNU, ele funciona em qualquer plataforma, Windows, Macintosh e Linux. A ferramenta fornece uma interface bem uniforme para muitos algoritmos de aprendizagem diferentes, juntamente com métodos de pré-processamento e para avaliar o resultado de esquemas em um determinado conjunto de dados de aprendizagem (WITTEN H IAN. 2005 p. 398 tradução nossa).

5.4.3 Ferramenta SQUID

De acordo com Marcelo (2006, p. 3) diz que,

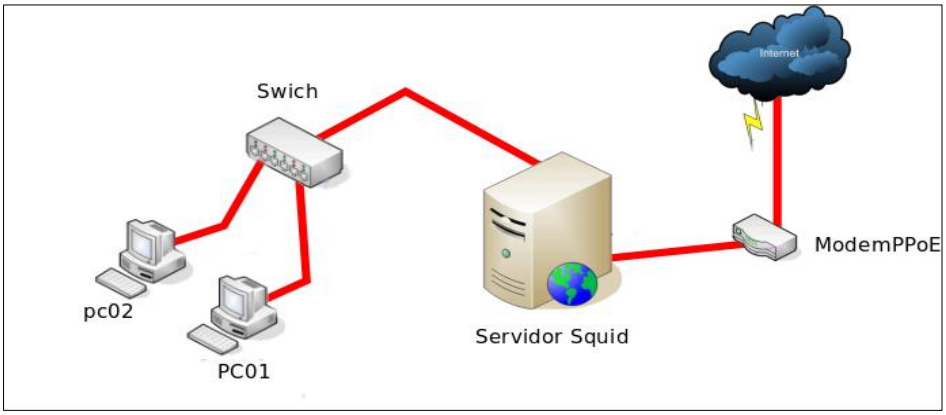
> O SQUID é um dos proxies para Linux mais utilizado hoje na Internet. Trata-se de um programa robusto, simples e extremamente confiável que se tornou um dos pacotes obrigatórios, a serem instalados em qualquer provedor ou empresa que deseja aumentar a performance de sua conexão ou criar regras de acesso (ACLs) para servidores web.

Ainda para Marcelo, o SQUID foi feito a partir de um projeto Harvest da ARPA, com um dos principais envolvido é Duane Wessels, do National Laboratory for Applied Network Research, dando o nome de SQUID (lula) apenas para distinguir um software do outro.

“O Squid é um servidor proxy utilizado para gerenciar o acesso a internet (rede externa), pois implementa um controle sobre o conteúdo que deve ou não ser acessado pelas máquinas clientes gerenciadas por este servidor.[..]”.(FORMICE, 2013, p. 19).

Ainda de acordo com Formice (2013), o Squid é considerado também um Firewall de Aplicação, no qual trabalhando como uma ferramenta que não permite acesso a Internet diretamente, e sim passando pelo Firewall no que atua como intermediador, passando assim para o proxy (squid) que efetua a comunicação entre ambos os lados por meio da avaliação do número de sessão TCP dos pacotes. A Figura 7 demonstra o funcionamento do Squid.

Figura 7: Esquema de funcionamento do Squid



Fonte: Elaborada pelo autor.

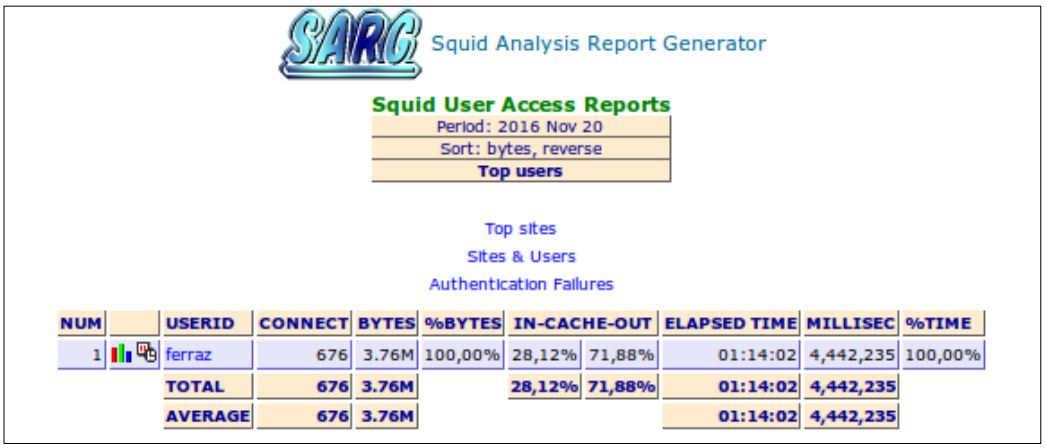
"O arquivo *squid.conf* é o responsável por todas as configurações. Neste arquivo serão criadas as listas de acesso (ACLs) onde pode-se inserir/modificar parâmetros importantes no sistema" (MARCELO, 2005, p. 9).

No momento que o Squid é iniciado com o sistema, o arquivo *squid.conf* é executado, lendo todas ACLs já configuradas.

5.4.4 Ferramenta SARG (Squid Analysis Report Generation)

O SARG é uma ferramenta que gera relatórios de análise de acesso dos usuários a sites através do Squid, ela foi desenvolvida pelo brasileiro Pedro Orso. Ainda para Marcelo (2006), trata-se de uma ferramenta que é essencial para os administradores que desejam informações sobre o conteúdo e comportamento dos usuários em sua rede, dentro desses relatórios gerados, tem-se informações como IP, nome de usuário e quantidades de bytes utilizados na rede. Na Figura 8 é feita a demonstração do relatório de acesso gerado pelo Sarg (MARCELO, 2006).

Figura 8: Exemplo de Relatório gerado pelo SARG



NUM		USERID	CONNECT	BYTES	%BYTES	IN-CACHE-OUT		ELAPSED TIME	MILLISEC	%TIME
1		ferraz	676	3.76M	100,00%	28,12%	71,88%	01:14:02	4,442,235	100,00%
		TOTAL	676	3.76M		28,12%	71,88%	01:14:02	4,442,235	
		AVERAGE	676	3.76M				01:14:02	4,442,235	

Fonte: Elaborada pelo autor.

6 TRABALHOS CORRELATOS

A aplicação dos métodos para realizar o processo de mineração de dados e aprendizagem de máquina, para detectar, classificar e agrupar conteúdos de sites, definindo qual algoritmo deve utilizar para o desenvolvimento de uma aplicação, ainda tem sido pouco explorado. Em uma breve pesquisa realizada, é possível encontrar trabalhos e artigos que citam a respeito dos métodos e das aplicações das técnicas de classificação de textos para capturar conteúdos de sites.

Dentro dos contextos citados, pode-se citar o trabalho de apresentação no curso superior de tecnologia em análise e desenvolvimento de sistemas da Universidade Tecnológica Federal do Paraná, intitulado, uso de técnicas e ferramentas de mineração de dados na extração de informações sobre o comportamento de uso dos recursos da internet na tufar, de autoria de Gustavo Rafaeli Valiati, que aborda o tema de captura de conteúdos de arquivos de logs, gerados pelos servidores da UTFPR (Valiati, 2008).

Esse estudo teve o intuito de analisar e verificar os conteúdos adquiridos para classificação na ferramenta Weka, utilizando o algoritmo Apriori, com objetivo de verificar em que horário, havia uma oscilação maior no interior da Universidade.

Com esses logs e aplicado o algoritmo no Weka obteve-se o resultado dos horários da pesquisa.

7 METODOLOGIA

Para o desenvolvimento deste trabalho foram adotadas duas etapas distintas e de grande importância para atingir o seu objetivo que consiste em comparar diferentes técnicas de classificação de conteúdos de sites. As etapas deste trabalho incluem:

a) Levantamento Bibliográfico.

b) Comparação de algoritmos.

7.1 LEVANTAMENTO BIBLIOGRÁFICO

Uma pesquisa exploratória ajuda o pesquisador a desenvolver os problemas de pesquisa e prioridades sobre determinado assunto. É importante salientar, que as prioridades serão estabelecidas de acordo com hipóteses explicativas surgidas durantes esse processo. Além disso, a pesquisa exploratória gera informações sobre as práticas do projeto de pesquisa, um fator fundamental para o avanço da proposta pesquisada (MATTAR, 2012).

Sendo assim, foi realizada uma investigação dos aspectos teóricos. Portanto, um levantamento bibliográfico foi o primeiro passo para a construção da pesquisa, avaliando bibliografias em livros, artigos científicos, buscando um aprofundamento maior relacionado às temáticas Inteligência Artificial, Aprendizado de Máquina e Processamento de Língua Natural.

7.2 COMPARAÇÃO DE ALGORITMOS

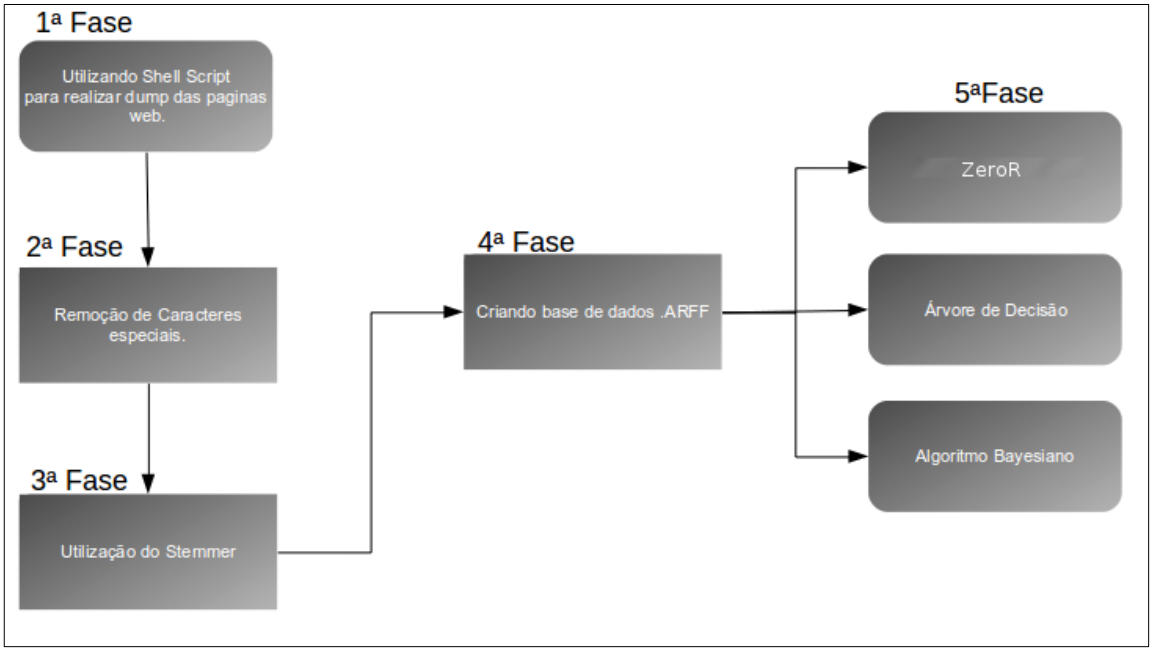
A etapa de comparação de algoritmos exigiu alguns passos anteriores que serão explicitados ao longo desta seção.

7.2.1 Coleta de Dados

A metodologia aplicada consistiu em realizar testes utilizando base de dados montadas com diferentes tamanhos, com 100 e 400 páginas de sites da internet

respectivamente. As bases foram formadas com conteúdo impróprio. Para a coleta de conteúdo foi acessado o site http://www.alexa.com/topsites.Este site é de uma empresa que possui uma ferramenta de estatística aplicada em marketing digital, possuindo ranking dos sites mais acessados de todos os países. Para montagem das bases foram utilizadas 100 e 400 páginas consideradas impróprias (conteúdo pornográfico) e outras 100 e 400 consideradas próprias (para os conteúdos considerados próprios foram acessadas páginas aleatórias, de notícias, esportes, e etc..). Os textos contidos nas páginas formam a base que foi aplicada na ferramenta WEKA, utilizando como o sistema operacional Linux e algumas aplicações nativas, com intuito de verificar a adequação de algoritmos na tarefa de classificar o que é e o que não é considerado como conteúdo impróprio. Sendo assim foram realizadas algumas etapas, visando essa comparação. A Figura 9 ilustra os processos para coleta de dados da base.

Figura 9: Processos para coleta de dados



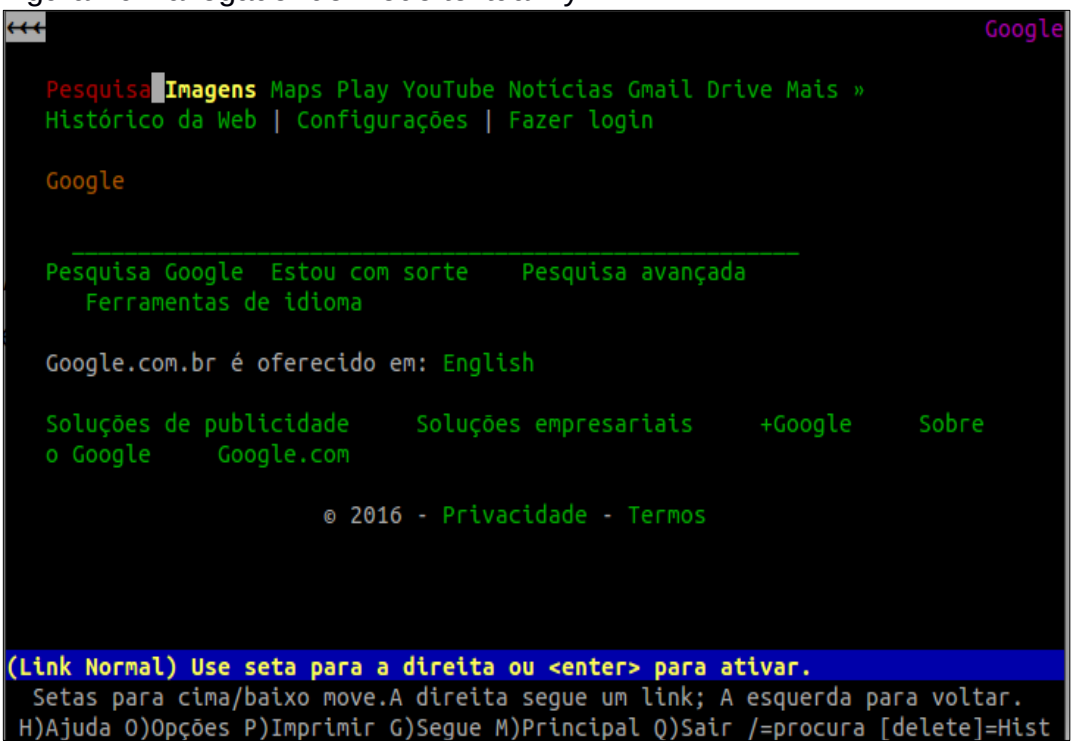
Fonte: Elaborada pelo autor.

Na Figura 9 a 1ª, a 2ª e a 3ª fases correspondem à etapa de pré-processamento. Na 1ª fase tem-se como meta capturar e “filtrar” de forma automática o conteúdo (textos) de páginas web utilizando uma linguagem conhecida

como Shell Script e a ferramenta Lynx que é um navegador em modo texto. O Shell, por ser um interpretador de comando de alto nível de sofisticação, também pode ser utilizado como linguagem de programação. A 4ª fase, corresponde à transferências dos arquivos para base de dados .arff e por fim a 5ª fase é a realização dos testes propriamente ditos na ferramenta WEKA.

A ferramenta Lynx é de grande importância para essa tarefa, pois por suas características (navegador puramente textual) garante apenas a captura de textos, removendo quaisquer outros elementos tais como imagens e formatações que não são relevantes para esta pesquisa. A Figura 10 ilustra uma interface do Lynx aberta na página do Google.

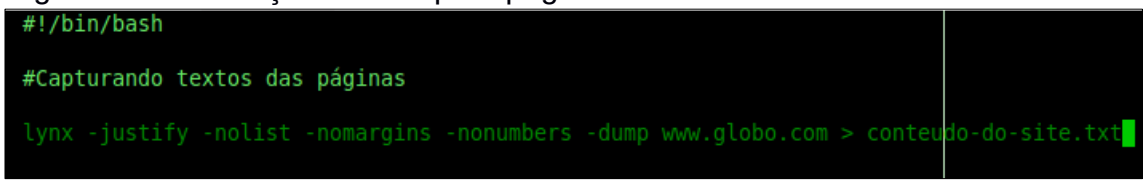
Figura 10:Navegador de modo textual Lynx



Fonte: Elaborada pelo autor

Juntamente com a linguagem Shell, o Lynx permite gerar de forma eficiente arquivos de textos simples (. txt) com os conteúdos da página. A integração entre o Lynx a Shell se dá por meio de um comando que permite realizar o dump das páginas (Figura 11).

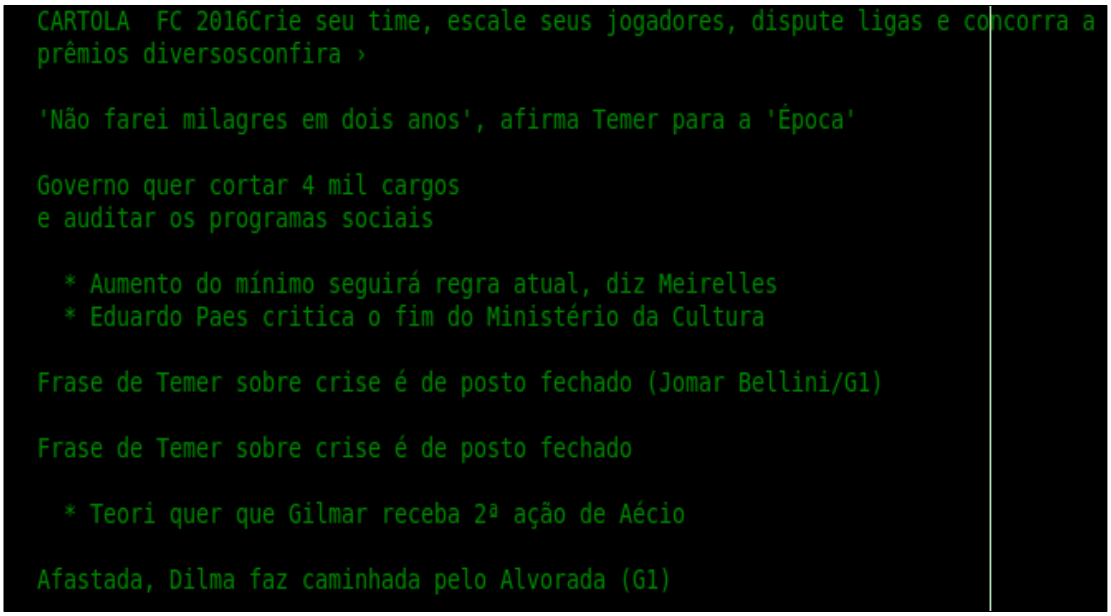
Figura 11: Realização do dump da página



Fonte: Elaborada pelo autor.

Os comandos ilustrados na Figura 11 permitem ainda outras tarefas importantes para o pré-processamento: **(-justify) -**captura de modo justificado, **(-nolist) -** desativa os links de conteúdos quando se utiliza o dump**, (-nomargins) –** sem configuração de margens, **(-nonumbers) –** desabilita campos de numerações das páginas, **(-dump) –** realiza o dump (captura dos conteúdos das páginas), direcionando o conteúdo para um arquivo no qual foi denominado **conteudo-do-site.txt.**A Figura 12 mostra o resultado da captura do conteúdo da página <www.globo.com>

Figura 12: Resultado da captura do site www.globo.com



Fonte: Elaborada pelo autor.

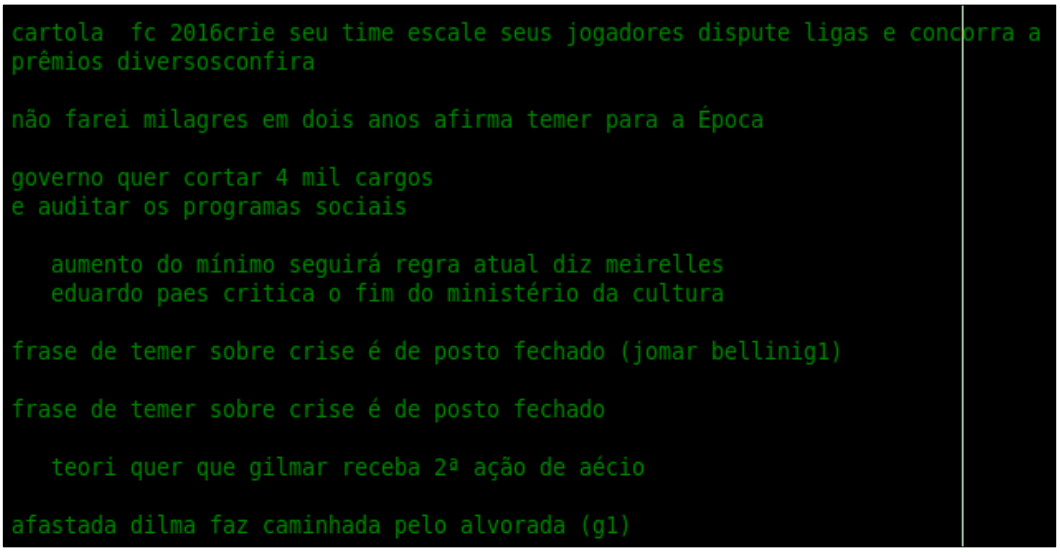
Na segunda fase é realizada a remoção de caracteres especiais, com objetivo de filtrar o texto usado para montar a base de dados a ser utilizada pela ferramenta WEKA. Para isso são usados os comandos **cat** e **sed** (instruções de Shell Script), conforme mostrado na Figura 13.

Figura 13: Comando - set e cat "filtragem"

Fonte: Elaborada pelo autor.

O comando **cat** captura o conteúdo do arquivo.txt, passando as informações para o comando **sed** que tem a função de remover os caracteres que não serão utilizados pela aplicação WEKA. Após essa remoção o texto estará pronto para ser processado na terceira fase de uma forma mais eficiente. A Figura 14 ilustra o resultado do processamento de um texto após a segunda fase

Figura 14: Resultado do processamento do texto



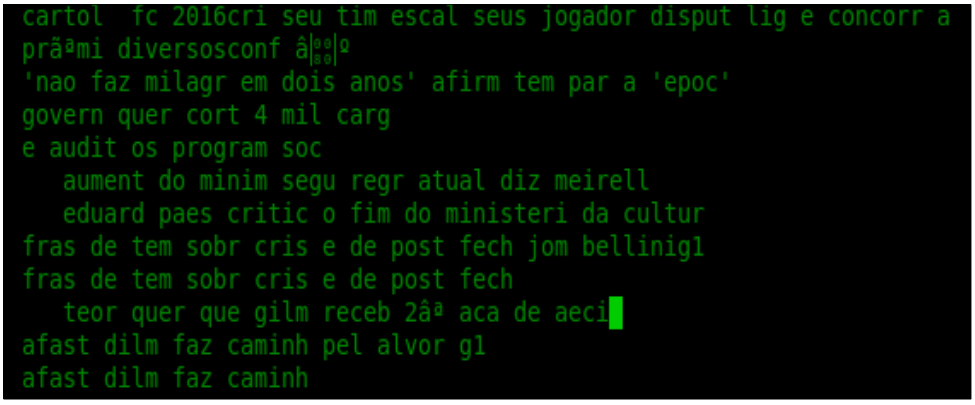
Fonte: Elaborada pelo autor

Na terceira fase é utilizado um Stemmer, (desenvolvido pelo Núcleo Interinstitucional de Linguística Computacional (NILC) da USP) com o objetivo de transformar as palavras em seu radical. Em morfologia linguística e recuperação de informação a stemização (do inglês, stemming) é o processo de reduzir palavras flexionadas (ou às vezes derivadas) ao seu tronco (stem), base ou raiz, geralmente uma forma da palavra escrita. O tronco não precisa ser idêntico à raiz morfológica da palavra; ele geralmente é suficiente que palavras relacionadas sejam mapeadas

para o mesmo tronco, mesmo se este tronco não for ele próprio uma raiz válida (KUMAR, 2011).

A Figura 15 representa o texto após a aplicação do Stemmer (no arquivo.txt que foi gerado na segunda fase).

Figura 15:Texto após aplicação do programa Stemmer



Fonte: Elaborada pelo autor

A quarta fase corresponde à geração automática, na Figura 16 pode-se observar a Shell Script que realiza a criação da base de dados a ser utilizada pela ferramenta WEKA. Para tentar classificar melhor os conteúdos capturados, também foi gerada uma base com Shell Script sem a configuração do Stemmer. A base de dados corresponde a um arquivo no formato ARFF, cuja estrutura é a seguinte:

@relation = nome da base

@attribute = atribui o nome que separa as variáveis {própria, impropria}.

@attribute text String = define o que tipo de arquivo é, sendo, String, NUMERIC…

@data = base aonde os dados vão ser inseridos abaixo.

propria, 'conteúdo do site armazenado'
impropria, 'conteúdo do site armazenado'

O formato ARFF que consiste basicamente de duas partes:

a) A primeira contém uma lista de todos os atributos, onde devemos definir o tipo do atributo ou os valores que ele pode representar. Quando são utilizados os valores estes devem estar entre “{ }“ separados por vírgulas.
b) A segunda parte consiste das instâncias, ou seja, os registros a serem minerados com o valor dos atributos para cada instância separado por vírgula. A ausência de um item em um registro deve ser atribuída pelo símbolo “?”.

Figura 16:Código em Shell Script

```
#!/bin/bash

#CAPTURANDO DADOS DO SITE
lynx -justify -nolist -nomargins -nonumbers -dump www.globo.com > conteudo-do-site.txt

#FILTRANDO OS CARACTERES INVALIDOS
cat conteudo-do-site.txt | sed 's/\/\|\>\|\''\|\-\|\''\|\?\|\<\|\>\|\!\|\.\|\,\|\_\|\-\|\[\|\]\|\*\|\+\|//g' |
sed 'y/ABCDEFGHIJKLMNOPQRSTUVWXYZÀàáéÍíÓóÚÃÕãõç/abcdefghijklmnopqrstuvwxyzaaaeeiiooouaoaoc/' > conteudo-do-site1.txt

#APLICANDO STEMMER NO ARQUIVO GERADO
wine stemmer_arquivo.exe conteudo-do-site1.txt

#CAPUTRANDO ARQUIVO GERADO PELO STEMMER E SALVANDO NOVAMENTE NO MESMO ARQUIVO
cat conteudo-do-site1.txt.stemmed > conteudo-do-site1.txt

#REMOVENDO OUTROS CARACTERES APÓS O STEMMER APLICADO
cat conteudo-do-site1.txt | sed 's/uqe/que/g; s/qeu/que/g; s/euq/que/g; s/unknown//g; s/@card@//g; s/[*()";:%·<>\]//g;' > conteudo-do-site2.txt

#REMOVENDO ESPAÇAMENTOS
cat conteudo-do-site2.txt |  sed '/^\s*$/d' | paste -s -d ' ' > conteudo-do-site3.txt

#SALVANDO CONTEUDO DO ARQUIVO NA VARIAVEL
    propriol=$(cat conteudo-do-site3.txt)
    impropriol=$(cat conteudo-do-site10.txt)

 #-------- SALVANDO A BASE .arff --------#

echo "@relation conteudoimproprios
@attribute palavras {propria,impropria}
@attribute text String


@data
propria, '$propriol'
impropria, '$impropriol' " > BaseDados.arff
```

Fonte: Elaborada pelo autor

A quinta e última fase corresponde à aplicação dos algoritmos de classificação escolhidos para esta investigação: Árvore de decisão, Classificador Bayesiano, ZeroR (usado com baseline). Para isso, o arquivo ARFF gerado na quarta fase é diretamente carregado na ferramenta WEKA e as devidas configurações são feitas para execução dos experimentos e coleta de resultados que permitirão a comparação entre os algoritmos. Os algoritmos foram comparados por meio de medidas comumente utilizadas para avaliar a tarefa em questão: precisão, recall e f-measure (Figura 17).

A recall (cobertura) corresponde a porcentagem de amostras positivas classificadas corretamente sobre o total de amostras positivas. A precisão

corresponde a porcentagem de amostras positivas classificadas corretamente sobre o total de amostras classificadas como positivas. E por fim, a f-measure é uma média ponderada de precisão e recall.

Figura 17: Porcentagem de amostras

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{Positivas}$$

$$Precisão = \frac{TP}{TP + FP}$$

$$F\text{-}measure = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)}$$

Fonte: Elaborada pelo autor.

7.3 RESULTADOS OBTIDOS

Todos os testes realizados foram feitos com a ferramenta Weka, utilizando os seguintes classificadores: Classificador NaiveBayes, J48 (Árvore de Decisão), ZeroR com opções de Teste: em Percent split de 66%. O Percent split recebe como entrada, a informação referente à percentagem na qual os dados serão divididos em subconjuntos de treinamento e teste, além do algoritmo de mineração a ser usado.

A diferenciação baseou-se basicamente em duas bases de dados uma possuindo 100 sites e outra com 400, também diferenciando dois parâmetros, em uma base que foi aplicada stemização e em outra não.

Com a utilização de um laço de repetição dentro do script não se pode ter a certeza da captura de todos os sites dentro da lista criada. Não obtendo sucesso de captura da quantidade de sites citados, sendo assim foram definidas duas bases que tinham os dados verdadeiros capturados pelo script. A denominada Base1 é responsável pela captura das 100 páginas, no entanto foi capturado a quantidade de 35 instâncias, a denominada Base2 é responsável pela lista de 400 páginas, também não obtendo sucesso de captura, conseguindo gerar 375 instâncias para os testes que serão citados a seguir.

7.4 TESTES COM J48

A realização dos testes com o J48 consistiu em avaliar tempo de execução, tamanho de sua árvore, instâncias corretamente/incorretamente classificadas, e sua matriz de confusão.

7.4.1 Resultados dos testes com a Base1 (J48)

Na Figura 18 os resultados obtidos realizando as configurações de Percent split em 66%, utilizando a base com de 35 instâncias e aplicado a stemização.

Figura 18: Resultados do teste com J48 e aplicado a stemização

```
Number of Leaves  :     8

Size of the tree :      15


Time taken to build model: 0.23 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          26               74.2857 %
Incorrectly Classified Instances         9               25.7143 %
Kappa statistic                          0.4944
Mean absolute error                      0.2267
Root mean squared error                  0.4133
Relative absolute error                 44.2125 %
Root relative squared error             80.0258 %
Total Number of Instances               35

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.667     0.143      0.875     0.667     0.757      0.869    propria
                 0.857     0.333      0.632     0.857     0.727      0.869    impropria
Weighted Avg.    0.743     0.219      0.778     0.743     0.745      0.869

=== Confusion Matrix ===

  a  b   <-- classified as
 14  7 |  a = propria
  2 12 |  b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que foi gerada uma árvore de decisão com 15 nós (size of the tree) que conseguiu uma classificação geral correta (precisão) de cerca de 74,20% dos dados. Nota-se uma diferença na precisão (precision), recall (cobertura) e f-measure obtidas dentro de cada classe. Para a classe “própria” essas medidas foram respectivamente: 87,5%; 66,7% e 75,7%. Já para a classe “imprópria” o desempenho nas medidas de precisão e f-measure obtidas foi inferior, tendo sido alcançados os seguintes valores respectivos: 63,2% e 72,7%. Entretanto, a cobertura alcançou um desempenho superior totalizando 85,7%.

A matriz de confusão obtida mostra que para a classe própria, das 21 instâncias, 14 foram corretamente classificadas e 7 foram incorretamente classificadas. Já para a classe imprópria, das 14 instâncias, 12 foram corretamente classificadas e 2 não.

Na Figura 19 os resultados obtidos sem a aplicação da stemização, também foi utilizado o parâmetro de split em 66% e sem aplicação da stemização.

Figura 19: Resultados do teste com J48 sem stemização

```
Number of Leaves  :     8

Size of the tree :      15


Time taken to build model: 0.23 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          26               74.2857 %
Incorrectly Classified Instances         9               25.7143 %
Kappa statistic                          0.4944
Mean absolute error                      0.2267
Root mean squared error                  0.4133
Relative absolute error                 44.2125 %
Root relative squared error             80.0258 %
Total Number of Instances               35

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.667     0.143      0.875     0.667     0.757      0.869    propria
                 0.857     0.333      0.632     0.857     0.727      0.869    impropria
Weighted Avg.    0.743     0.219      0.778     0.743     0.745      0.869

=== Confusion Matrix ===

  a  b   <-- classified as
 14  7 |  a = propria
  2 12 |  b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que foi gerada uma árvore de decisão do mesmo tamanho da anterior com 15 nós (size of the tree) que conseguiu uma classificação geral correta (precisão) de cerca de 74,28% dos dados. Pela proximidade da precisão geral, o processo de stemização parece não influenciar o comportamento do classificador. Da mesma forma que no teste anterior, nota-se uma diferença na precisão, recall e f-measure obtidas dentro de cada classe. Para a classe “própria” essas medidas foram respectivamente: 87,5%; 66,7% e 75,7%. Já para a classe “imprópria” o desempenho nas medidas de precisão e f-measure obtidas foi inferior, tendo sido alcançados os seguintes valores respectivos: 63,2% e 72,7%. Entretanto, novamente, a cobertura alcançou um desempenho superior totalizando 85,7%.

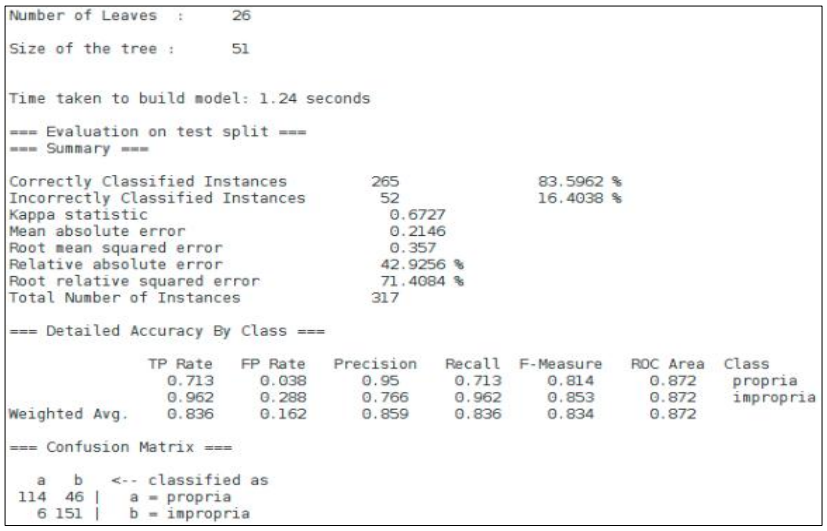
A matriz de confusão obtida mostra que para a classe própria, das 21 instâncias, 14 foram corretamente classificadas e 7 foram incorretamente classificadas. Já para a classe imprópria, das 14 instâncias, 12 foram corretamente classificadas e 2 não. Novamente, o mesmo comportamento do teste anterior.

O comportamento dentro das classes também se mostrou praticamente igual ao teste anterior, reforçando a ideia de que para este algoritmo e esta base de dados, o processo de stemização não gera diferenças significativas.

7.4.2 Resultados dos testes com a Base2 (J48)

Para os testes aqui descrito considerou-se uma base de dados maior que a anterior (371 instâncias). Na Figura 20 os resultados dos testes obtidos sem aplicação da stemização, os parâmetros de split também em 66%.

Figura 20: Resultados do teste com J48 sem stemização



```
Number of Leaves  :     26

Size of the tree :      51


Time taken to build model: 1.24 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         265               83.5962 %
Incorrectly Classified Instances        52               16.4038 %
Kappa statistic                          0.6727
Mean absolute error                      0.2146
Root mean squared error                  0.357
Relative absolute error                 42.9256 %
Root relative squared error             71.4084 %
Total Number of Instances              317

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.713     0.038      0.95      0.713     0.814      0.872    propria
                 0.962     0.288      0.766     0.962     0.853      0.872    impropria
Weighted Avg.    0.836     0.162      0.859     0.836     0.834      0.872

=== Confusion Matrix ===

   a   b   <-- classified as
 114  46 |   a = propria
   6 151 |   b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que foi gerada uma árvore de decisão com 51 nós (size of the tree) que conseguiu uma classificação geral correta (precisão) de cerca de 83,60% dos dados. Nota-se uma diferença na precisão (precision), recall (cobertura) e f-measure obtidas dentro de cada classe. Para a classe “própria” essas medidas foram respectivamente: 95,0%; 71,3% e 85,2%. Já para a classe “imprópria” o desempenho nas medidas de precisão e f-measure obtidas foi superior, tendo sido alcançados os seguintes valores respectivos: 76,6% e 72,7%. Entretanto, a cobertura alcançou um desempenho superior totalizando 87,2%.

A matriz de confusão obtida mostra que para a classe própria, das 160 instâncias, 114 foram corretamente classificadas e 46 foram incorretamente

classificadas. Já para a classe imprópria, das 157 instâncias, 151 foram corretamente classificadas e 6 não.

Na Figura 21 os resultados dos testes obtidos com aplicação da stemização, os parâmetros de split também em 66%.

Figura 21: Resultados do teste com J48 e aplicado a stemização

```
Number of Leaves  :     28

Size of the tree :      55


Time taken to build model: 1.11 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         260               82.0189 %
Incorrectly Classified Instances        57               17.9811 %
Kappa statistic                          0.6413
Mean absolute error                      0.2226
Root mean squared error                  0.3736
Relative absolute error                 44.518  %
Root relative squared error             74.7312 %
Total Number of Instances              317

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.688     0.045      0.94      0.688     0.794      0.849    propria
                 0.955     0.313      0.75      0.955     0.84       0.849    impropria
Weighted Avg.    0.82      0.177      0.846     0.82      0.817      0.849

=== Confusion Matrix ===

   a   b   <-- classified as
 110  50 |   a = propria
   7 150 |   b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que foi gerada uma árvore de decisão com 55 nós, uma diferença de 4 nós do teste anterior que conseguiu uma classificação geral correta (precisão) de cerca de 82,01% dos dados. Nessa situação avaliando a precisão geral, o processo de stemização parece influenciar o comportamento do classificador. No entanto Nota-se uma diferença na precisão, recall e f-measure obtidas dentro de cada classe comparada com a anterior. Para a classe “própria” essas medidas foram respectivamente: 94,0%; 68,8% e 79,4%. Já para a classe “imprópria” o desempenho nas medidas de precisão e f-measure obtidas foi inferior, tendo sido alcançados os seguintes valores respectivos: 75,0% e 84,0%. Entretanto, a cobertura alcançou um desempenho inferior totalizando 85,7%.

A matriz de confusão obtida mostra que para a classe própria, das 160 instâncias, 110 foram corretamente classificadas e 50 foram incorretamente classificadas. Já para a classe imprópria, das 157 instâncias, 150 foram corretamente classificadas e 7 não. O comportamento dentro das classes mostrou um resultado diferenciado ao teste anterior, reforçando a ideia de que para este

algoritmo, o aumento da base de dados, e o processo de stemização geram diferenças significativas.

7.5 TESTES COM NAIVEBAYES

A realização dos testes com oNAIVEBAYES consistiu em avaliar tempo de execução, instâncias corretamente/incorretamente classificadas, e sua matriz de confusão.

7.5.1 Resultados dos testes com a Base1(NAIVEBAYES)

Na Figura 22 os resultados dos testes obtidos sem aplicação da stemização, os parâmetros de split também em 66%.

Figura 22: Resultado do teste com NaiveBayes sem aplicação da stemização

```
Time taken to build model: 0.1 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          31               88.5714 %
Incorrectly Classified Instances         4               11.4286 %
Kappa statistic                          0.75
Mean absolute error                      0.1143
Root mean squared error                  0.3381
Relative absolute error                 22.292  %
Root relative squared error             65.4621 %
Total Number of Instances               35

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.286      0.84      1         0.913      0.857    propria
                 0.714     0          1         0.714     0.833      0.934    impropria
Weighted Avg.    0.886     0.171      0.904     0.886     0.881      0.888

=== Confusion Matrix ===

  a  b   <-- classified as
 21  0 |  a = propria
  4 10 |  b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que conseguiu uma classificação geral correta (precisão) de cerca de 88,57% dos dados. Nota-se uma diferença na precisão (precision), recall (cobertura) e f-measure obtidas dentro de cada classe. Para a classe “própria” essas medidas foram respectivamente: 84,0%; 100% e 91,3%. Já para a classe “imprópria” o desempenho nas medidas se “equilibraram” com a precisão maior e f-measure inferior, tendo sido alcançados os seguintes valores respectivos: 100% e 83,3%. A

cobertura alcançou um desempenho superior dos testes realizados no J48, totalizando 85,7%.

A matriz de confusão obtida mostra que para a classe própria, das 21 instâncias, 21 foram corretamente classificadas e 0 foram incorretamente classificadas. Já para a classe imprópria, das 14 instâncias, 10 foram corretamente classificadas e 4 não.

Na Figura 23 os resultados dos testes obtidos com aplicação da stemização, os parâmetros de split também em 66%.

Figura 23: Resultado do teste com NaiveBayes com aplicação da stemização

```
Time taken to build model: 0.05 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          31               88.5714 %
Incorrectly Classified Instances         4               11.4286 %
Kappa statistic                          0.75
Mean absolute error                      0.1143
Root mean squared error                  0.3381
Relative absolute error                 22.292  %
Root relative squared error             65.4621 %
Total Number of Instances               35

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.286      0.84       1         0.913      0.857    propria
                 0.714     0          1          0.714     0.833      0.934    impropria
Weighted Avg.    0.886     0.171      0.904      0.886     0.881      0.888

=== Confusion Matrix ===

  a  b   <-- classified as
 21  0 |  a = propria
  4 10 |  b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que também conseguiu uma classificação geral correta (precisão) de cerca de 88,57% dos dados. Nota-se o mesmo comportamento na precisão (precision), recall (cobertura) e f-measure obtidas dentro de cada classe. Para a classe “própria” essas medidas foram respectivamente: 84,0%; 100% e 91,3%. Já para a classe “imprópria” o desempenho nas medidas se “equilibraram” com a precisão maior e f-measure inferior, tendo sido alcançados os seguintes valores respectivos: 100% e 83,3%. A cobertura também alcançou um desempenho superior dos testes realizados no J48,porém idêntico ao anterior totalizando 85,7%.

A matriz de confusão obtida mostra que para a classe própria, das 21 instâncias, 21 foram corretamente classificadas e 0 foram incorretamente

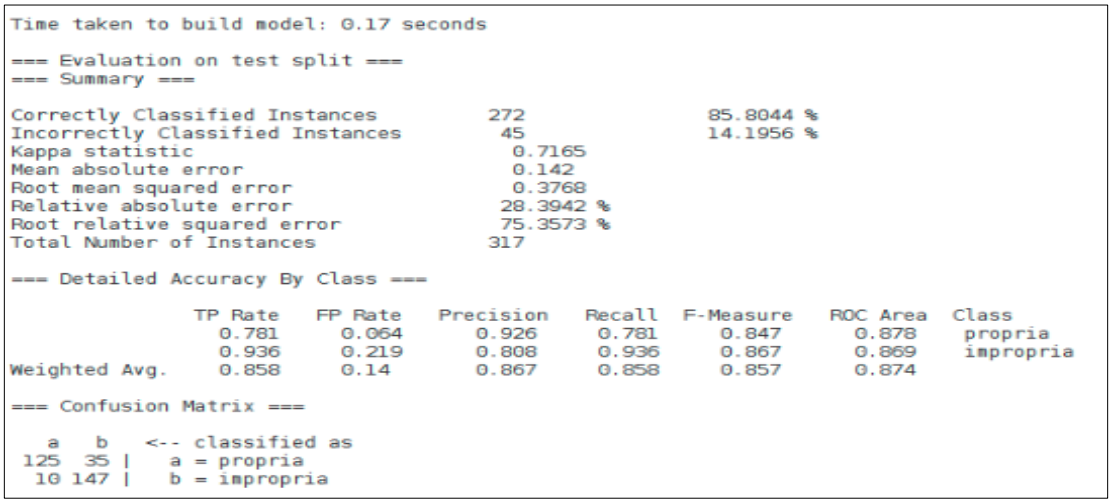
classificadas. Já para a classe imprópria, das 14 instâncias, 10 foram corretamente classificadas e 4 não.

O comportamento dentro das classes também se mostrou praticamente igual ao teste anterior, reforçando a ideia de que para este algoritmo e esta base de dados, o processo de stemização não gera diferenças significativas.

7.5.2 Resultados dos testes com a Base2(NAIVEBAYES)

Para os testes aqui descritos considerou-se uma base de dados maior que a anterior (371 instâncias). Na Figura 24 os resultados dos testes obtidos sem aplicação da stemização, os parâmetros de split também em 66%.

Figura 24: Resultado do teste com NaiveBayes sem aplicação da stemização



```
Time taken to build model: 0.17 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         272               85.8044 %
Incorrectly Classified Instances        45               14.1956 %
Kappa statistic                          0.7165
Mean absolute error                      0.142
Root mean squared error                  0.3768
Relative absolute error                 28.3942 %
Root relative squared error             75.3573 %
Total Number of Instances              317

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.781     0.064      0.926     0.781     0.847      0.878    propria
                 0.936     0.219      0.808     0.936     0.867      0.869    impropria
Weighted Avg.    0.858     0.14       0.867     0.858     0.857      0.874

=== Confusion Matrix ===

   a   b   <-- classified as
 125  35 |   a = propria
  10 147 |   b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que conseguiu uma classificação geral correta (precisão) de cerca de 85,83% dos dados. Nota-se uma diferença na precisão (precision), recall (cobertura) e f-measure obtidas dentro de cada classe. Para a classe “própria” essas medidas foram respectivamente: 92,6%; 78,1% e 84,7%. Já para a classe “imprópria” o desempenho da precisão foi inferior e com maior e f-measure superior, tendo sido alcançados os seguintes valores respectivos: 80,8% e 86,7%. A cobertura alcançou um desempenho superior dos testes realizados no J48 e no teste anterior, totalizando 87,4%.

A matriz de confusão obtida mostra que para a classe própria, das 160 instâncias, 125 foram corretamente classificadas e 35 foram incorretamente classificadas. Já para a classe imprópria, das 157 instâncias, 147 foram corretamente classificadas e 10 não.

Na Figura 25 os resultados dos testes obtidos com aplicação da stemização, os parâmetros de split também em 66%.

Figura 25: Resultado do teste com NaiveBayes com aplicação da stemização

```
Time taken to build model: 0.24 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         195               61.5142 %
Incorrectly Classified Instances       122               38.4858 %
Kappa statistic                          0.2253
Mean absolute error                      0.3847
Root mean squared error                  0.6202
Relative absolute error                 76.9569 %
Root relative squared error            124.0429 %
Total Number of Instances              317

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.95      0.726      0.571     0.95      0.714      0.618    propria
                 0.274     0.05       0.843     0.274     0.413      0.862    impropria
Weighted Avg.    0.615     0.391      0.706     0.615     0.565      0.739

=== Confusion Matrix ===

   a   b   <-- classified as
 152   8 |   a = propria
 114  43 |   b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que conseguiu uma classificação geral correta (precisão) de cerca de 61,51% dos dados, resultado bem inferior de todos os testes já realizado até o presente momento. Nota-se a que nos comportamentos como na precisão (precision), recall (cobertura) e f-measure obtidas dentro de cada classe. Para a classe “própria” essas medidas foram respectivamente: 57,1%; 95% e 71,4%. Já para a classe “imprópria” o desempenho nas medidas de precisão e f-measure obtidas foi inferior, tendo sido alcançados os seguintes valores respectivos: 84,3% e 41,3%. A cobertura alcançou um desempenho inferiordos testes realizados totalizando 73,9%.

A matriz de confusão obtida mostra que para a classe própria, das 160 instâncias, 152 foram corretamente classificadas e 8 foram incorretamente classificadas. Já para a classe imprópria, das 157 instâncias, 43 foram corretamente classificadas e 114 não.

O comportamento dentro das classes também se inferior, reforçando a ideia de que para este algoritmo de que a base de dados com maior instâncias, o processo de stemização influenciou para resultados inferiores.

7.6 TESTES COM ZEROR

Aplicação do método nos testes, utilizando os parâmetros de split em 66% para verificar seus resultados e comparar com demais métodos.

7.6.1 Resultados dos testes com a Base1(ZEROR)

Na Figura 26 os resultados do teste obtido sem aplicação da stemização e com parâmetros de split 66%.

Figura 26: Resultado do teste com ZeroR sem aplicação da stemização

```
ZeroR predicts class value: impropria

Time taken to build model: 0 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          14               40      %
Incorrectly Classified Instances        21               60      %
Kappa statistic                          0
Mean absolute error                      0.5127
Root mean squared error                  0.5164
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances               35

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0         0          0         0         0          0.5      propria
                 1         1          0.4       1         0.571      0.5      impropria
Weighted Avg.    0.4       0.4        0.16      0.4       0.229      0.5

=== Confusion Matrix ===

  a  b   <-- classified as
  0 21 |  a = propria
  0 14 |  b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que conseguiu uma classificação geral correta (precisão) de cerca de 40% dos dados. Nota-se por tratar de um algoritmo de baseline, o seu teste selecionou apenas uma classe definindo seus parâmetros. O algoritmo ZeroR possui uma diferença na precisão (precision), recall (cobertura) e f-measure obtidas dentro de cada classe. Para a classe “própria” essas medidas foram respectivamente: 0%; 0% e 0%. Já para a classe “imprópria”, a qual foi “escolhida” pelo algoritmo tem-se o

desempenho nas medidas de precisão e f-measure, tendo sido alcançados os seguintes valores respectivos: 4% e 57,1%. Sua cobertura alcançou um desempenho totalizando 5%.

A matriz de confusão obtida mostra que para a classe própria, das 21 instâncias, 0 foram corretamente classificadas e 21 foram incorretamente classificadas. Já para a classe imprópria, das 14 instâncias, 14 foram corretamente classificadas.

Na Figura 27 os resultados do teste obtido sem aplicação da stemização e com parâmetros de split 66%.

Figura 27: Resultado do teste com ZeroR com aplicação da stemização

```
=== Classifier model (full training set) ===

ZeroR predicts class value: impropria

Time taken to build model: 0 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          14               40      %
Incorrectly Classified Instances        21               60      %
Kappa statistic                          0
Mean absolute error                      0.5127
Root mean squared error                  0.5164
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances               35

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0         0          0         0         0          0.5      propria
                 1         1          0.4       1         0.571      0.5      impropria
Weighted Avg.    0.4       0.4        0.16      0.4       0.229      0.5

=== Confusion Matrix ===

  a  b   <-- classified as
  0 21 |  a = propria
  0 14 |  b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que conseguiu uma classificação geral correta (precisão) de cerca de 40% dos dados. Nota-se que os resultados foram os mesmos, o seu teste selecionou apenas uma classe definindo seus parâmetros. O algoritmo ZeroR possui uma diferença na precisão (precision), recall (cobertura) e f-measure obtidas dentro de cada classe. Para a classe “própria” essas medidas foram respectivamente: 0%; 0% e 0%. Já para a classe “imprópria”, a qual foi “escolhida” pelo algoritmo tem-se o desempenho nas medidas de precisão e f-measure, tendo sido alcançados os seguintes valores respectivos: 4% e 57,1%. Sua cobertura alcançou um desempenho totalizando 5%.

A matriz de confusão obtida mostra que para a classe própria, das 21 instâncias, 0 foram corretamente classificadas e 21 foram incorretamente

classificadas. Já para a classe imprópria, das 14 instâncias, 14 foram corretamente classificadas.

O comportamento dentro das classes também sãoinferior a outros testes, reforçando a ideia de que para este algoritmo por se tratar de ser baseline, é desconsiderando completamente o processo de stemização e sua base dados.

7.6.2 Resultados dos testes com a Base2 (ZEROR)

Na Figura 28 os resultados do teste obtido sem aplicação da stemização e com parâmetros de split 66% e 400 páginas.

Figura 28: Resultado do teste com ZeroR sem aplicação da stemização

```
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         160               50.4732 %
Incorrectly Classified Instances       157               49.5268 %
Kappa statistic                          0
Mean absolute error                      0.4999
Root mean squared error                  0.5
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              317

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         1          0.505     1         0.671      0.5      propria
                 0         0          0         0         0          0.5      impropria
Weighted Avg.    0.505     0.505      0.255     0.505     0.339      0.5

=== Confusion Matrix ===

   a   b   <-- classified as
 160   0 |   a = propria
 157   0 |   b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que conseguiu uma classificação geral correta (precisão) de cerca de 50% dos dados. Nota-se por tratar de um algoritmo de baseline, o seu teste selecionou apenas uma classe definindo seus parâmetros. O algoritmo ZeroR possui uma diferença na precisão (precision), recall (cobertura) e f-measure obtidas dentro de cada classe. Para a classe “própria” a qual foi selecionada pelo algortimosuas medidas foram respectivamente: 50,5%; 100% e 67,1%. Já para a classe “imprópria”, tem-se o desempenho inferior em todas medidas, tendo sido alcançados os seguintes valores respectivos: 0% e 0%. Porém sua cobertura também alcançou um desempenho totalizando 5%.

A matriz de confusão obtida mostra que para a classe própria, das 160 instâncias, 160 foram corretamente classificadas e 0 foram incorretamente classificadas. Já para a classe imprópria, das 157 instâncias, 0 foram corretamente classificadas e 157 classificadas incorretamente.

Na Figura 29 os resultados do teste obtido com aplicação da stemização e com parâmetros de split 66%.

Figura 29: Resultado do teste com ZeroR com aplicação da stemização

```
ZeroR predicts class value: propria

Time taken to build model: 0 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         160               50.4732 %
Incorrectly Classified Instances       157               49.5268 %
Kappa statistic                          0
Mean absolute error                      0.4999
Root mean squared error                  0.5
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              317

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         1          0.505     1         0.671      0.5      propria
                 0         0          0         0         0          0.5      impropria
Weighted Avg.    0.505     0.505      0.255     0.505     0.339      0.5

=== Confusion Matrix ===

   a   b   <-- classified as
 160   0 |   a = propria
 157   0 |   b = impropria
```

Fonte: Elaborada pelo autor.

Observa-se que os testes realizados com aplicação da stemização, também obteve os mesmos resultados do teste anterior, incluindo suas medidas e matriz de confusão. O comportamento dentre as classes demonstrou que utilizando o algoritmo ZeroR, não importando a quantidade de instâncias e se esta aplicado a stemização, os resultados foram inferiores a qualquer outro algoritmos aqui testados.

7.7 AMBIENTE DE TESTE E APLICAÇÃO

Com os resultados obtidos pode-se definir o algoritmo que classifica melhor os conteúdos na base de dados, ou seja, o Naive-Bayes (sem stemização).

No ambiente de teste foram utilizados um computador com as seguintes configurações: Core 2 Duo 2.4Ghz, 4Gb Memória RAM e Disco Rígido de 80GB, para o servidor com o S.O Debian 8, configurado com servidor SQUID com Proxy e

gerador de relatórios SARG para efetivar os resultados dos testes. Para aplicar o teste como cliente, foi utilizado um computador Core i7 2.5Ghz, 8Gb de memória RAM e Disco rígido de 750GB, com S.O Ubuntu 16.04.

A aplicação foi desenvolvida no servidor, utilizando um script em Shell, o qual utiliza os resultados dos testes realizados no Weka, para melhor utilização para esse fim, foram utilizados os testes adquiridos pelo NaiveBayes utilizando a Base1 e sem aplicação da stemização, a qual obteve uma classificação correta de 88,57%, Na Figura 30 demonstra uma parte do retorno do teste realizado com NaiveBayes.

Figura 30: Resultado do teste com NaiveBayes.

```
                                Class
Attribute                     propria impropria
                               (0.49)    (0.51)
=================================================
zona
  mean                         0.1147    0.0652
  std. dev.                    0.3544    0.3305
  weight sum                       51        53
  precision                     0.266     0.266
```

Fonte: Elaborada pelo autor.

A aplicação desenvolvida em shell script teve como objetivo gerar uma lista de palavras consideradas impróprias gerada pelos testes classificados no Weka. A Figura 31 demonstra como trata o arquivo para criar a lista de palavras impróprias.

Figura 31: Script gerador de lista para o Squid

```
#!/bin/bash


#SEPARAÇÃO DO CABEÇALHO

sed '/^===/,/^=\+$/d' TestWekaNaiveBayes |
awk 'BEGIN{FS="\n";RS="\n\n"} {print $1,$2}'|
awk '$3<$4 {print $1}' >> palavrasimproprias

#COPIANDO AS PALAVRAS PARA PASTA DO SQUID
sudo cp palavrasimproprias /etc/squid/palavrasimproprias
```

Fonte: Elaborada pelo autor.

A primeira linha do código “*sed /^===/,/^=\+$/d' TestWekaNaiveBayes |*” remove todo o topo do arquivo de resultado, no qual possui apenas informações que não serão necessárias para os fins, seleciona somente o início dos resultado como pode-se observar na Figura 30. A segunda linha do código correspondente à “*awk 'BEGIN{FS="\n";RS="\n\n"} {print $1,$2}'|*”, é onde trata cada “bloco” como um registro, capturando os campos “01” e “02”. A Figura 32 demonstra como ficou o código.

Figura 32:Código configurado para capturar a 1ª Coluna

	PROPRIA	IMPROPRIA
youtube mean	0.2117	0
your mean	0.1428	0.2274
youporn mean	0	0.1263
young mean	0.0698	0.293
you mean	0.1825	0.2073
york mean	0.0767	0
years mean	0.1898	0.0422
year mean	0.1526	0.1272

Fonte: Elaborada pelo autor.

Na terceira linha do código, “*awk '$3<$4 {print $1}'>> palavrasimproprias”*, realiza uma comparação das colunas 3 (representada pela coluna PROPRIA) e 4 (representada pela coluna IMPROPRIA), no qual se a coluna 4 for menor que a coluna 3, então captura a palavra da coluna 1 salvando o conteúdo capturado para o arquivo palavrasimproprias, que terá uma utilização no Squid no qual possui um configura um arquivo que utiliza em suas configurações, definido como palavrasimproprias. Foi criado uma ACL que bloqueia todas as palavras contidas nesse arquivo. Na Figura 33 o exemplo de configuração do squid utilizando ACL para bloquear as palavras contidas nesse arquivo.

Figura 33: Bloqueio de palavras utilizando ACL

```
#bloqueando palavras definidas pelo script
acl palavrasimproprias url_regex -i "/etc/squid/palavrasimproprias"
http_access deny palavrasimproprias
```

Fonte: Elaborada pelo autor.

Com essa configuração pode-se gerar relatórios de acessos para verificar a eficácia dessa lista de palavras bloqueadas. A Figura 34 mostra o relatório gerado com a ferramenta Sarg, mostrando o que foi liberado antes da aplicação da ACL.

Figura 34: Relatório gerado antes da aplicação da lista de bloqueio

ACCESSED SITE	CONNECT	BYTES	%BYTES	IN-CACHE-OUT		ELAPSED TIME	MILLISEC	%TIME
www.msn.com	15	1.13M	11,44%	0,00%	100,00%	00:00:30	30,079	0,41%
media.trafficfactory.biz	1	827.00K	8,36%	0,00%	100,00%	00:00:21	21,474	0,29%
s.glbimg.com	58	819.99K	8,29%	0,00%	100,00%	00:00:27	27,450	0,37%
img-s-msn-com.akamaized.net	85	811.10K	8,20%	0,00%	100,00%	00:00:53	53,919	0,73%
s3.glbimg.com:443	21	797.97K	8,07%	100,00%	0,00%	00:01:57	117,712	1,60%
img-l3.xvideos.com	45	612.15K	6,19%	0,00%	100,00%	00:02:03	123,550	1,67%
ssl.gstatic.com:443	7	556.85K	5,63%	100,00%	0,00%	00:03:49	229,460	3,11%
akfs.nspmotion.com	21	404.27K	4,09%	0,10%	99,90%	00:00:12	12,843	0,17%
www.bing.com	14	339.90K	3,44%	0,00%	100,00%	00:00:11	11,716	0,16%
static-hp-eus-s-msn-com.akamaized.net	8	328.46K	3,32%	0,00%	100,00%	00:00:09	9,380	0,13%
www.google.com.br:443	10	282.53K	2,86%	100,00%	0,00%	00:04:29	269,781	3,66%
static-hw.xvideos.com	13	263.73K	2,67%	0,00%	100,00%	00:00:04	4,776	0,06%

Fonte: Elaborada pelo autor.

Observando-se o relatório gerado pelo Sarg antes da aplicação do script, verificamos que todos os sites acessados, não receberam o bloqueio. Na Figura 35 demonstra o relatório depois da aplicação da ACL com as palavras impróprias.

Figura 35: Relatório gerado após aplicação da lista de bloqueio (DENIED)

ACCESSED SITE	CONNECT	BYTES	%BYTES	IN-CACHE-OUT		ELAPSED TIME	MILLISEC	%TIME	
s.glbimg.com	171	737.52K	32,11%	100,00%	0,00%	00:00:00	4	12,50%	DENIED
s2.glbimg.com	122	577.07K	25,12%	100,00%	0,00%	00:00:00	20	62,50%	DENIED
s3.glbimg.com	67	288.00K	12,54%	100,00%	0,00%	00:00:00	8	25,00%	DENIED
servidor-tcc:3128	21	95.74K	4,17%	100,00%	0,00%	00:00:00	0	0,00%	DENIED
ep01.epimg.net	21	91.63K	3,99%	100,00%	0,00%	00:00:00	0	0,00%	DENIED
b.scorecardresearch.com	13	57.67K	2,51%	100,00%	0,00%	00:00:00	0	0,00%	DENIED
s04.video.glbimg.com	12	51.30K	2,23%	100,00%	0,00%	00:00:00	0	0,00%	DENIED
tags.globo.com	12	51.09K	2,22%	100,00%	0,00%	00:00:00	0	0,00%	DENIED
www.google.com.br	10	45.53K	1,98%	100,00%	0,00%	00:00:00	0	0,00%	DENIED

Fonte: Elaborada pelo autor.

Pode-se observar que após aplicação da regra com palavras impróprias, todas as páginas de testes foram bloqueadas (DENIED). Ou seja, aplicação do método de bloqueio por palavras realizou de certa forma o bloqueio de conteúdos impróprios, porém a quantidade de palavras adquiridas foi extensa gerando certos problemas

para acessar qualquer outro tipo de página, não só aquelas consideradas como impróprias. A Figura 36 ilustra os resultados finais obtidos e suas comparações.

Figura 36: Resultados Finais dos Testes

Algoritmos	Base de Dados	Precision	Recall	F-Measure	ROC Área	Classe	Matriz de Confusão		Corretamente Class.	Incorretamente Class.	A = própria B = impropria
							A	B			
J48	Base1	87,50%	66,70%	75,70%	86,90%	própria	14	7	74,28%	25,71%	
		63,20%	85,70%	72,70%	86,90%	impropria	2	12			
		77,8%%	74,50%	74,50%	86,90%						
Naive-Bayes		84,00%	100,00%	91,30%	85,70%	própria	21	0	88,57%	11,42%	
		100,00%	71,40%	83,30%	93,40%	impropria	4	10			
		90,40%	88,60%	88,10%	88,80%						
ZeroR		0	0	0	50,00%	própria	0	21	40,00%	60,00%	
		40,00%	100,00%	57,10%	50,00%	impropria	0	14			
		16,00%	40,00%	22,90%	50,00%						
J48	Base2	94,00%	68,80%	79,40%	86,90%	própria	110	50	83,59%	16,40%	
		75,00%	95,50%	84,00%	86,90%	impropria	7	150			
		84,60%	82,00%	81,70%	86,90%						
Naive-Bayes		57,10%	95,00%	71,40%	61,80%	própria	152	8	61,51%	38,48%	
		84,30%	27,40%	41,30%	86,20%	impropria	114	43			
		70,60%	61,50%	56,50%	73,90%						
ZeroR		0	0	0	50,00%	própria	160	0	50,47%	49,52%	
		40,00%	100,00%	57,10%	50,00%	impropria	157	0			
		16,00%	40,00%	22,90%	50,00%						
J48	Base1	87,50%	66,70%	75,70%	86,90%	própria	14	7	74,28%	25,71%	
		63,20%	85,70%	72,70%	86,90%	impropria	2	12			
		77,70%	77,80%	74,50%	86,90%						
Naive-Bayes		84,00%	100,00%	91,30%	85,70%	própria	21	0	88,57%	11,42%	
		100,00%	71,40%	83,30%	93,40%	impropria	4	10			
		90,40%	88,60%	88,10%	88,80%						
ZeroR		0	0	0	50,00%	própria	0	21	40,00%	60,00%	
		40,00%	100,00%	57,10%	50,00%	impropria	0	14			
		16,00%	40,00%	22,90%	50,00%						
J48	Base2	95,00%	71,30%	81,40%	87,20%	própria	114	46	83,59%	16,40%	
		76,60%	96,20%	85,30%	87,20%	impropria	6	151			
		85,90%	83,60%	83,40%	87,20%						
Naive-Bayes		92,60%	78,10%	84,70%	87,80%	própria	125	35	85,80%	14,19%	
		80,80%	93,60%	85,70%	86,90%	impropria	10	147			
		86,70%	85,80%	85,70%	85,70%						
ZeroR		0	0	0	50,00%	própria	160	0	50,47%	49,52%	
		40,00%	100,00%	57,10%	50,00%	impropria	157	0			
		16,00%	40,00%	22,90%	50,00%						

Fonte: Elaborada pelo autor.

8 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo a implementação de uma aplicação de forma que criasse uma lista de bloqueio para a ferramenta SQUID, utilizando um algoritmo que fosse possível classificar as palavras, considerando impróprias e próprias de uma forma automática e em tempo real.

Com testes realizados na ferramenta Weka, pode-se obter resultados positivos. A captura e testes de forma automática foram concluídos com êxito, os testes realizados tanto com NaiveBayes e J48 obtiveram uma classificação positiva, em média 80% das classificações foram corretas.

Porém, mesmo com resultados positivos o método da implementação de uma lista de palavras impróprias automática, não se tornou eficaz, pois a aquisição de muitas palavras acabou bloqueando boa parte dos sites acessados, mesmo aqueles que não possuíam palavras consideradas impróprias, tornando a navegação internet inviável. Atualmente os métodos manuais ainda mesmo que sejam trabalhosos, são mais eficientes, pois o administrador da rede consegue definir as palavras que exatamente representa uma palavra imprópria. Existem alguns pontos que corrigidos na aplicação consiga se tornar um método mais eficiente, como por exemplo integrar com a API da ferramenta Weka e unificar os códigos em shell script, realizando uma pré filtragem dos conteúdos e definindo um limite de palavras específicas para ser capturadas, outro ponto a ser melhorado, seria utilizar a base de acesso gerada pelo próprio Squid, tendo como aquisição base do qual usuários logados no proxy acessaram páginas web.

Desta forma, conclui-se que o presente trabalho contribuiu para o conhecimento de Mineração de Dados e aprendizagem de máquinas, explorando uma forma com potencial de implementação de bloqueio de conteúdos impróprios acessados em uma rede de computadores. Além disso, apesar do método precisar de melhorias, ainda pode ser utilizado para obtenção de conteúdos de páginas web, como textos, avaliações e outras informações necessárias para o fim de seu uso. A qualidade e o tamanho das bases de teste utilizadas deverão ser revistas em trabalhos futuros.

REFERÊNCIAS

ANTONÍN, M. A. M. **Tecnologías del texto y del habla**. Espanha: Universitat Barcelona, 2004.

AUGUSTO, JOSÉ B. **Extração de Conhecimento & Mineração de Dados**. Disponível em: <http://dcm.ffclrp.usp.br/~augusto/teaching/ami/AM-I-KDD-DM.pdf>. Acesso em: 20 maio. 2016.

ARARIBÓA, G. **Inteligência Artificial** – Rio de Janeiro, 3 Ed. Livro Técnico e Cientifico Editora Ltda, 1988.

ÁVILA B.C. **Data Mining**. Dissertação (Mestrado em informática Aplicada). Disponível em: < http://www.ler.esalq.usp.br/download/TEC%202002.11.PDF>. Acesso em: 20 maio. 2016.

BRACHNAD, R.J. & ANAND, **T. The process of knowledge discovery in databases**. In: FAYYAD, U.M. et al. Advances in Knowledge Discovery in Data Mining. Menlo Park: AAAI Press, 1996.

BRAGA, L.P.V.B., **Introdução à Mineração de Dados 2Ed: Edição ampliada e revisada**. Editora E-Papers, 2005.

COPPIN, BEN. **Inteligência Artificial**: tradução e revisão técnica Jorge Duarte Pires. Rio de Janeiro: LTC, 2013.

BARROS, SABINO DAVI. **Como instalar e configurar o SARG.** Disponível em <http://www.vidati.com.br/2011/04/08/como-instalar-e-configurar-o-sarg/>. Acesso em 10 ago 2016.

EMMANUEL PASSOS; RONALDO GOLDSCHMIDT. **Data mining: um guia prático** Rio de Janeiro, Elsevier, 2005.

FERNANDES. **Inteligência Artificial**: Noções Gerais – 2 imp. 2005. Florianópolis: VisualBooks.

FIGUEIRA, R. M. A. - **Miner: um Software de Interferência de Dependência Funcional**. 1 imp. 1998. Rio de Janeiro.

FORMICE, C.R – **Linux Configurações de Serviços de Rede Apostila Técnica** – Clube de Autores, 2013.

GONGORA, A. D. **O que é Inteligência Artificial?** UFSC, 2007. Disponível em: <http://www.egov.ufsc.br/portal/sites/default/files/anexos/6515-6514-1-PB.pdf>>. Acesso em: 25 mar. 2016.

GROTH. **Data mining a Hands-On for Business Professional** New Jersey: Prentice Hall PTR, 1998.

HAYKIN. **Redes neurais**: princípios e práticas / Simon Haykin. Trad. Paulo Martins Engel. 2.ed. 2001. Porto Alegre: Bookman.

HERTZOG, R. **The Debian Administrator's Handbook, Debiaj Jessie from Discovery to Mastery.** - 2015 – Freexian.<https://debian-handbook.info/download/stable/debian-handbook.pdf>. Acesso em 29 Set 2016

IBM, **Visão geral sobre o Linux.** Disponível em: <http://www.ibm.com/developerworks/br/library/l-linuxuniversal/>. Acesso em: 27 Abr. 2016.

KUMAR, E. - **Natural Language Processing** – 2011. Disponível em: < https://books.google.com.br/books/about/Artificial_Intelligence.html?id=rNmAY-RcGKYC&redir_esc=y >. Acesso em: 29 Abr. 2016

KUROSE, JAMES F. ROSS, KEITH W. **Redes Computadores e a Internet:** Uma abordagem top-down. 3 ed. São Paulo: Pearson Addison Wesley, 2006.

LUGER, George F. I**nteligência Artificial**: 6ª Edição. São Paulo: Daniel Vieira, 2014.

LUÍS, ANA R. **Estudos de Linguística** – Coimbra, 2011.

MATTAR, F. N. **Pesquisa de Marketing, edição compacta**. 5. ed. Rio de Janeiro: Elsevier, 2012.

MARIMOTO, C. E. **Linux - Guia prático**. Porto Alegre: Sul Editores, 2009.

MARCELO, A. **Configurando o Proxy para Linux**. 5ª ed ampl. Rio de Janeiro: Brasport, 2006.

MCCABE, JAMES D. **Network Analysis, Architecture, and Design.** 3 ed. Burlington: Morgan Kaufmann Publishers, 2007.

NIKOLOPOULOS, C. **Expert systems.New York:** Marcel Dekker, Inc., 1997.

PITANGA, MARCOS. **Construindo Super Computadores com Linux**: 3ª Edição. Rio de Janeiro: Brasport, 2008. <https://goo.gl/ttA3nN> Acesso em: 20 mai. 2016.

REVISTABW. Classificadores Bayesianos.**Revista Brasileira de Web**: Tecnologia. Disponível em: <http://www.revistabw.com.br/revistabw/classificadores-bayesianos/>. Criado em: 14/02/2015. Última atualização: 24/07/2015. Acesso em:08 ago. 2016.

RUSSEL, D.; NORVIG, P. **Inteligência Artificial**: tradução da segunda edição. Rio de Janeiro: Elsevier, 2004.

RICH, ELAINE. - **Inteligência Artificial**: tradução Newton Vasconcellos. Revisão técnica Nizam Omar. - São Paulo: McGraw-Hill, 1998.

REZENDE, S. OLIVEIRA. **Sistemas inteligentes e aplicações** – Barueri, 2005.

RICCI, Bruno; MENDONÇA, Nelson. **Squid – Solução Definitiva.** Rio de Janeiro. Ciência Moderna. 2006.

SANTOS, G. M. **A realidade virtual**. Campo Grande: UCDB, 2001.

ISO 50, BLOG. O ColumbiasDec PDP-7. Disponível em:<http://blog.iso50.com/3446/columbias-dec-pdp-7>. Acesso em 09 ago. 2016.

SHIELDS, R. **Cultures of Internet**, Virtual Spaces, Real Histories, Living Bodies. London: SAGE Publications, 1996.

SILBERSCHATZ, ABRAHAM, **- Fundamentos de Sistemas Operacionais,** 8ª Edição – Rio de Janeiro, 2010.

SOARES, WALACE, et al. **Linux: Fundamentos –** 1ªed. - Erica, São Paulo, 2010.

SAYAD, SAYAD. **An Introdution to Data Minig.** Disponível em: <http://www.saedsayad.com/data_mining_map.htm>. Acesso em 9 Nov 2016.

TANENBAUM, A. S, et al. **Operation System Design and Implemetation**, Third Edition Disponível em: <https://goo.gl/cvBpFy>. Acesso em: 20 mai. 2016.

UNIMEP, **Conceitos e Aplicações de Data Mining**. Disponível em: <http://www.unimep.br/phpg/editora/revistaspdf/rct22art02.pdf/> Acesso em: 27 Abr. 2016.

WITTEN, H. IAN. FRANK, EIBE. **Data mining**: practical machine learning tools and techniques / – 2nd ed. 2005. (pg. 398-424).

ZEMBRZUSKI, M. C. **Classificadores Bayesianos**. Disponível em:<http://www.inf.ufrgs.br/~alvares/CMP259DCBD/classificadores-bayseanos.pdf>. Acesso em: 20 mai. 2016.