UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FAVE: UMA PROPOSTA PARA VERIFICAÇÃO DE EXTRATORES DE DADOS DE PÁGINAS HTML

JOÃO MIGUEL GEHLEN DA SILVA

CHAPECÓ
2018

JOÃO MIGUEL GEHLEN DA SILVA

FAVE: UMA PROPOSTA PARA VERIFICAÇÃO DE EXTRATORES DE DADOS DE PÁGINAS HTML

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.
Orientador: Prof. Dr. Guilherme Dal Bianco

CHAPECÓ
2018

Gehlen da Silva, João Miguel

FAVE: Uma Proposta para Verificação de Extratores de dados de páginas HTML / por João Miguel Gehlen da Silva. – 2018.
51 f.: il.; 30 cm.

Orientador: Guilherme Dal Bianco
Monografia (Graduação) - Universidade Federal da Fronteira Sul, Ciência da Computação, Curso de Ciência da Computação, RS, 2018.

1. Extração de dados. 2. Wrapper. 3. Extração Web. 4. Manutenção de wrappers. 5. Verificação de wrappers. I. Dal Bianco, Guilherme. II. Título.

E-mail: joaogehlen91@gmail.com

JOÃO MIGUEL GEHLEN DA SILVA

FAVE: UMA PROPOSTA PARA VERIFICAÇÃO DE EXTRATORES DE DADOS DE PÁGINAS HTML

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Guilherme Dal Bianco

Aprovado em: 03\07\2018

BANCA EXAMINADORA:

Dr. Guilherme Dal Bianco - UFFS

Dr. Denio Duarte - UFFS

Ma. Andressa Sebben - UFFS

RESUMO

O constante crescimento de serviços online, por exemplo, comparação de preços e produtos, agregadores de conteúdos, entre outros, impulsiona a demanda por soluções para a extração de dados. Para que informações oriundas internet possam ser comparadas ou agrupadas, é necessário extrair os dados relevantes das páginas web em um formato estruturado. As técnicas que providenciam a extração de dados são conhecidas como *wrappers*. Cada *wrapper* é desenvolvido usando como base a página HTML e produz um conjunto de informações estruturadas. Porém quando uma página HTML é modificada, o *wrapper* para de funcionar ou funciona de maneira incorreta. Atualmente já existem diversos estudos para fazer o ajuste automático do sistema de extração de dados, procedimento conhecido como *wrapper maintenance*. Este trabalho apresenta algumas técnicas de *wrapper maintenance* e propõe uma melhoria no método de automação de extratores tomando como base as técnicas apresentadas.

Palavras-chave: Extração de dados. Wrapper. Extração Web. Manutenção de wrappers. Verificação de wrappers. Extração de dados. Wrapper. Extração Web. Manutenção de wrappers. Verificação de wrappers.

ABSTRACT

The constant growth of online services, for example, price and product comparison, content aggregators, among others, drives the demand for solutions for data extraction. In order for information from the Internet to be compared or grouped, it is first necessary to extract relevant data from web pages in a structured format. The techniques that provide data extraction are known as *wrappers*. Each *wrapper* is developed based on the HTML page and produces a set of structured information. But when an HTML page is modified, *wrapper* may stop working or works incorrectly. Currently there are several studies to perform the automatic adjustment of the data extraction system, procedure known as *wrapper maintenance*. This work presents some techniques of *wrapper maintenance* and proposes an improvement in the method of extractor automation based on the presented techniques.

Keywords: Data Extraction. Wrapper. Web Extraction. Wrapper Maintenance. Wrapper Verification.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HTML	*HyperText Markup Language*
XML	*eXtensible Markup Language*
DOM	*Document Object Model*
KNN	*K-nearest Neighbors Classifier*
G	*Gaussian Classifier*
KM	*K-means Classifier*
KC	*K-center Classifier*
ACC	Acurácia
P	Precisão
R	Revocação
F1	*F-Measure*
WS	*Working-Set*
TS	*Training-Set*
UTS	*Unverified Training-Set*
VM	*Verification Model*
MAVE	*Multilevel wrApper Verification systEm*

SUMÁRIO

1 INTRODUÇÃO

1.1 Tema

Este trabalho de conclusão de curso apresenta técnicas e estratégias para a manutenção de extratores de dados de páginas HTML. O objetivo da extração de dados da internet é estruturar e rotular os dados para que possam ser usados posteriormente. A manutenção tem como objetivo automatizar o processo de extração de dados. Este processo tenta evitar a mediação de um programador toda vez que seja necessário algum ajuste no extrator. Esse trabalho tem como foco principal a verificação das informações extraídas, o que se constitui uma etapa fundamental no processo de manutenção, pois garante que os dados extraídos sejam consistentes, alertando ao usuário caso alguma anormalidade seja detectada.

1.2 Contextualização

É perceptível o aumento no número de informações que estão hospedadas na internet, mais especificamente em páginas web. Isso ocorre pelo fato de que pessoas estão comprando cada vez mais através da internet, desde de passagens aéreas, rodoviárias e até mesmo alimentos. Tal fenômeno propicia o surgimento de serviços dependentes de dados, por exemplo, a comparação de preços de um determinado produto em diferentes sites. Independente do objetivo que se tem com os dados da Web, é preciso extraí-los das páginas. Para isso, surgem então os sistemas de extração de dados de páginas HTML.

Sistemas de extração de dados, também chamados de *wrappers*, tem o objetivo de extrair informações de uma determinada página HTML. Por definição, um *wrapper* é um procedimento que transforma uma fonte de dados semi-estruturada (HTML, por exemplo) em uma fonte de dados estruturada (Banco de Dados Relacional, por exemplo), com o objetivo de facilitar o trabalho com os dados. A entrada de uma função *wrapper* será uma fonte de dados semi-estruturada ou desestruturada (texto livre) e a saída será somente a informação que interessa ao usuário em um formato estruturado.

Wrappers são desenvolvidos baseados em diferentes estratégias, (FERRARA et al., 2014), conforme descrito a seguir:

- **Expressões Regulares:** sendo uma das técnicas de extração de dados mais complexas, normalmente são utilizadas em conjunto com outros softwares que são desenvolvidos para

criar as expressões regulares, ou seja, com uma interface mais amigável, o usuário insere a fonte de dados e informa ao software qual informação é relevante. Esse sistema gera uma expressão regular, para ser aplicada. A expressão regular vai funcionar muito bem para determinada fonte de dados, mas qualquer alteração na fonte de dados, por menor que seja, vai fazer com que a expressão regular pare de funcionar e não encontre mais a informação requerida pelo usuário. Assim, o usuário será forçado a reconfigurar, ou seja, informar novamente para o sistema qual informação é importante, para a geração de uma nova expressão regular. Devido ao grande esforço humano, essa estratégia não é muito utilizada para fonte de dados que podem sofrer alterações (FERRARA et al., 2014).

- **Aprendizado de máquina:** apesar de ser usada para extração de dados de páginas HTML, é utilizada mais para dados desestruturadas, como textos livres, por exemplo. Requer um pré-processamento supervisionado para treinamento antes de extrair os dados. Em caso de alteração na fonte de dados, pode ser ineficiente (FERRARA et al., 2014).

- **Baseado na estrutura HTML:** aproveitando a estrutura HTML, conhecida como árvore DOM, essa estratégia atualmente é a mais estudada, pois faz menos suposições que as técnicas baseadas em aprendizado de máquina e também é menos complexa que as expressões regulares. Na ocorrência de qualquer alteração na árvore DOM o extrator para de funcionar, já que a comparação que vai ser feita é na estrutura da árvore como um todo e somente na estrutura, sem levar em conta os dados extraídos (FERRARA et al., 2014).

A mudança na fonte dados, pode ocasionar erros no extrator de dados (*wrappers*) ou, o que é pior, extrair dados incorretos. Muitas abordagens trabalham na questão de manutenção de extratores de dados. A maioria delas utiliza como base a estrutura HTML da página (FERRARA et al., 2014).

O problema de manutenção engloba tanto a tarefa de verificação da informação extraída, como a tarefa de reparação do procedimento de extração. Para realizar a manutenção de um extrator de dados, primeiro é necessário fazer algumas verificações, com o objetivo de identificar mudanças na fonte de dados. E o segundo passo é aplicar as correções necessárias para que a extração continue funcionando apropriadamente.

1.3 Objetivos

1.3.1 Geral

Explorar técnicas de verificação de *wrappers* propondo um aprimoramento em um dos métodos, a fim de identificar com mais precisão as informações extraídas incorretamente das páginas HTML.

1.3.2 Específicos

- Analisar técnicas de verificação de dados de extratores para mostrar que podem ser melhoradas em alguns pontos;
- Selecionar uma fonte de dados, que neste caso, devem ser páginas HTML e implementar um *wrapper* para extrair informações desta base;
- Implementar uma ferramenta de verificação de dados extraídos por um *wrapper* utilizando as técnicas já existentes juntamente com as melhorias feitas ao longo do trabalho.

1.4 Justificativa

Existem muitos tipos de alterações nas páginas HTML, por exemplo, quando é alterado o visual do site, e consequentemente alterações na estrutura da árvore DOM. Quanto mais tipos de alterações sendo identificados e tratados, maior será a eficiência do extrator de dados, já que não precisará ser reprogramado toda vez que ocorre alguma alteração na página. Com a diminuição do esforço humano para reprogramar extratores, mais barato será o custo de manutenção de um extrator de dados, e menor será o tempo que ele ficará inoperante. É visto que existem casos em que o *wrapper* terá que ser reprogramado manualmente, mas quanto menos ocorrer essa possibilidade, melhor.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os principais conceitos envolvendo a extração de dados. Tais conceitos são fundamentais para o entendimento do trabalho. Após uma breve explicação do que é extração de dados da Web (Seção 2.1) e apresentação dos conceitos mais importantes do HTML (Seção 2.2), serão detalhadas as principais técnicas de extração de dados (Seção 2.3). Por último, um explicação sobre o XPath, uma linguagem de consulta utilizada para HTML e XML (Seção 2.4).

2.1 Extração de dados da Web

Extração de dados da Web é o nome do processo de coletar dados em páginas da internet, mais especificamente páginas HTML. Facilmente confundido com *Web Scraping*, a extração em si tem a função de retirar as informações necessárias da página HTML e salvá-las em um banco de dados de forma estrutural. Pode ser encontrado na literatura pelo nome de *wrapper*, que, segundo Ferrara (FERRARA et al., 2014), é definido por um procedimento que pode ser implementado em um único *script* ou então várias classes de algoritmos que possuem a função básica de encontrar uma determinada informação alvo. A entrada de um *wrapper* é uma fonte de dados desestruturada ou semi-estruturada (páginas HTML) e a saída é uma fonte de dados estruturada.

Web Scraping engloba a extração de dados (*wrapper*) e o rastreamento das páginas Web (*crawler*). Para fins de contextualização, um *Web Crawler* tem a função de navegar e capturar páginas HTML.

2.2 HTML

Uma página na web é construída utilizando um padrão de linguagem de marcação de texto, denominado de HTML (*HyperText Markup Language*). Os navegadores interpretam essa linguagem e mostram o conteúdo que está configurado para ser exibido.

Toda e qualquer página HTML e suas variações (XHTML, XML, etc.) possuem um documento de modelos de objetos, denominado DOM (*Document Object Model*) que trata uma estrutura HTML como uma árvore, esta árvore é usada como um facilitador para acessar os dados da página HTML, apenas percorrendo os nós. A Figura 2.1 mostra um exemplo de uma

página HTML visualizada na web (a) e sua respectiva árvore DOM (b).

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Exemplo</title>
5    </head>
6    <body>
7      <table>
8        <tr>
9          <th>Loja</th>
10         <th>Telefone</th>
11       </tr>
12       <tr>
13         <td>Bazar do Pedro</td>
14         <td>9999-8888</td>
15       </tr>
16       <tr>
17         <td>Mercado do Almir</td>
18         <td>5555-5555</td>
19       </tr>
20     </table>
21   </body>
22 </html>
```

Listing 2.1 – Linguagem de Marcação de HiperTexto (HTML) da página web da Figura 2.1.

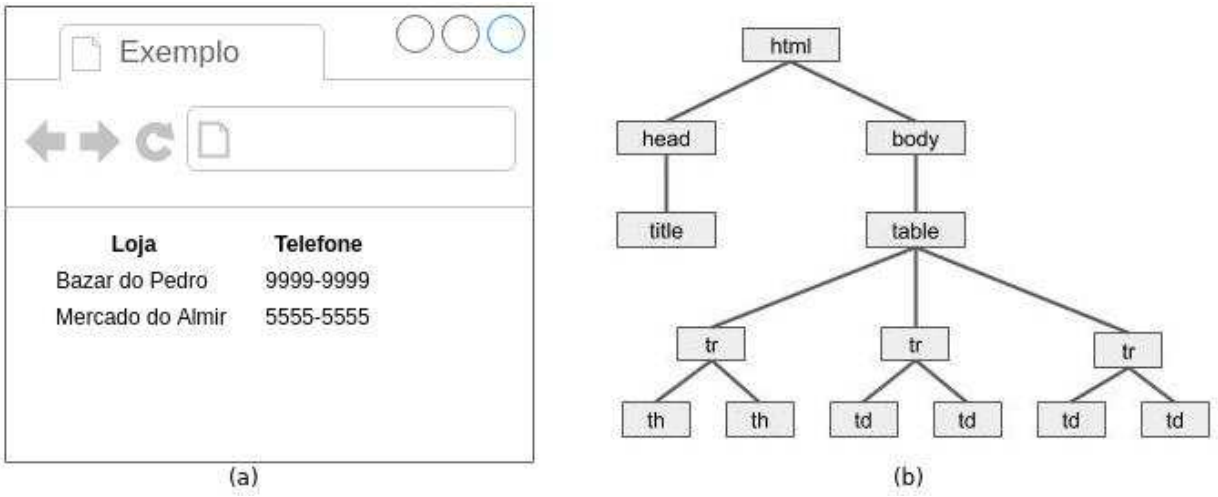


Figura 2.1 – Exemplo de página (a) e sua respectiva árvore DOM (b).

2.3 Técnicas de extração

O processo de extração de dados na internet, através de páginas HTML, é composto de duas etapas, uma faz a captura dos dados da pagina HTML e a outra etapa é responsável por fazer a verificação desses dados.

Existem várias abordagens diferentes utilizadas para captura de dados na web (FER-

RARA et al., 2014). Essas abordagens podem ser divididas em dois grandes grupos: baseada em aprendizado de máquina e baseado em regras.

A abordagem baseada em aprendizado de máquina tenta encontrar padrões para identificar as informações relevantes na base de dados. Existem os subgrupos de aprendizado supervisionado, não supervisionado e semi-supervisionado. Como essa abordagem não será utilizada no processo de extração, não cabe aqui a explicação da extração de dados baseada em aprendizado de máquina.

Outra abordagem de extração de dados é baseada em regras, por exemplo, expressões regulares. Em um primeiro momento, as abordagens baseadas neste método tinham uma complexidade muito alta para serem implementados, pois expressões regulares requerem um grande habilidade de programação e também um grande conhecimento da fonte de dado a ser trabalhada. Mais tarde, na tentativa de diminuir esforços, foram desenvolvidos os métodos baseados em estruturas de árvore, se aproveitando da fonte de dados semi-estruturada do HTML. Tal estrutura possibilita que cada *tag* HTML seja usada como rótulo para cada nó da árvore.

2.3.1 Baseada em árvore

A representação da estrutura de uma página HTML em forma de árvore é conhecida na literatura como "árvore DOM". Devido a tal estrutura ser bem definida, muitas técnicas utilizam árvore DOM para explorar a página HTML, guiando-se pela sua estrutura.

2.3.1.1 XPath

O método de extração por endereçamento é uma das abordagens mais utilizados e mais estudada atualmente (FERRARA et al., 2014). Este método usa como base a estrutura de dados em árvore, para explorá-la e encontrar o valor endereçado pelo usuário que criou o caminho XPath. Adiante será explicado com mais detalhes como funciona o XPath. Um dos pontos fracos do XPath é que, como ele é um caminho exato, definido, da raiz da árvore até o nó onde está a informação, qualquer alteração no caminho implica uma reconfiguração no XPath. Esse problema é o foco principal da área de reparação de extratores.

2.3.1.2 Similaridade entre árvores

Com o objetivo de extrair informações de páginas similares a partir de uma agrupamento de segmento similares em uma árvore, é proposta a técnica de ***tree edit distance matching*** (FERRARA; BAUMGARTNER, 2011).

A tarefa de calcular a distância entre duas árvores é inspirada no problema de comparar a distância entre duas strings. O termo distância aqui neste caso, significa o quanto duas strings ou duas árvores são similares. Por exemplo: considerando as palavras "panela" e "janelas", a distância entre ambas é 2, pois são necessárias duas operações para deixar a palavra "panela" igual a palavra "janelas" ou vice-versa. Primeira operação: troca-se o 'p' por 'j'; segunda operação: acrescenta-se o 's'.

A similaridade ou distância entre duas árvores segue o mesmo princípio usado para as strings. A Figura 2.2 mostra um exemplo de duas árvores cuja distância entre elas é de 3, ou seja, as árvores têm um grau de similaridade calculado em 3. Ainda neste exemplo, para ser mais específico, na Figura 2.2 (b) existe uma inserção de um novo nodo (circulado); remoção de um nodo (pontilhado); alteração do rótulo de um nodo ("span"). Assim, são necessárias 3 operações para deixar as duas árvores equivalentes.

Mais especificamente, o problema de descobrir a distância entre duas árvores consiste em: dadas duas árvores A e B com todos os nodos rotulados, encontrar a correspondência entre as árvores para transformar a árvore A em B, ou vice-versa, com o mínimo de operações.

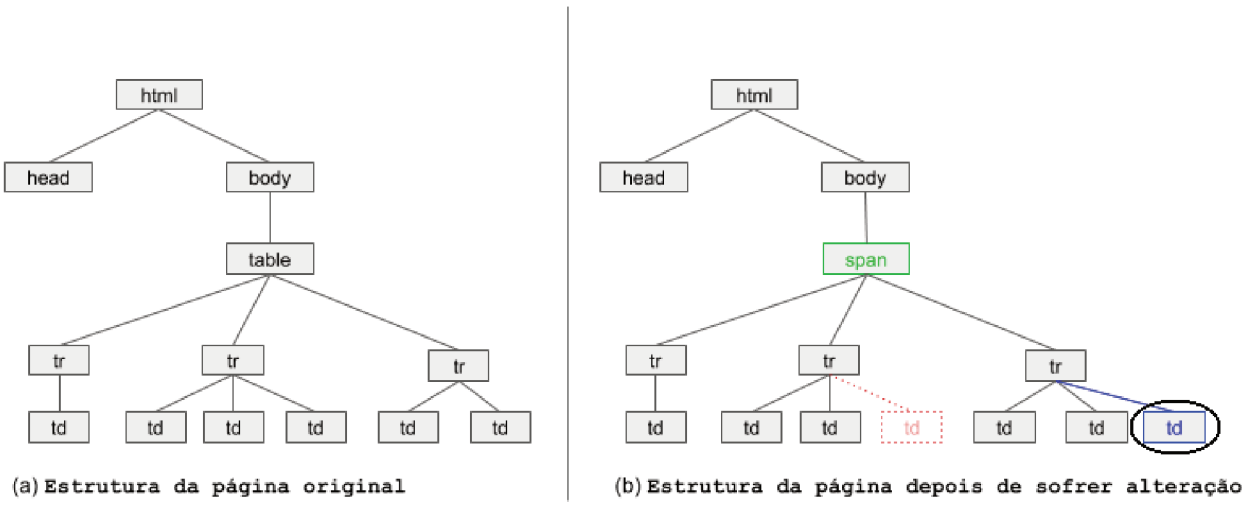


Figura 2.2 – Exemplo de duas estruturas diferentes, dando destaque às suas diferenças.

2.3.1.2.1 Algoritmo: Simple Tree Matching

Selkow (SELKOW, 1977) propôs um algoritmo computacionalmente eficiente para solucionar o problema de computar a distância entre duas árvores rotuladas. Apesar do algoritmo não lidar com a operação de permutação, o *Simple Tree Matching* serviu como base para muitas outras técnicas e variações que serão mostradas ao longo deste trabalho, mais especificamente na seção de trabalhos relacionados. Outra observação que deve ser feita sobre esse algoritmo é que ele não é capaz de fazer uma correspondência entre nodos que estão em diferentes níveis de cada árvore.

Este algoritmo ainda possui um custo computacional de O(n(A)n(B)) onde n(T) é o número de nodos da árvore T. O Algoritmo 1 é apresentado juntamente com a seguinte observação: d(n) representa o grau de um nodo n, ou seja o número de descendentes de primeiro grau de n; T(i) é a i-ésima sub árvore da árvore cuja raiz é T.

Algoritmo 1 SimpleTreeMatching(T', T'')

if T' has the same label of T'' **then**
    $m \leftarrow d(T')$
    $n \leftarrow d(T'')$
    for $i = 0$ to m **do**
        $M[i][0] \leftarrow 0$;
    for $j = 0$ to n **do**
        $M[0][j] \leftarrow 0$;
    for all i such that $1 \leq i \leq m$ **do**
        for all j such that $1 \leq j \leq n$ **do**
            $M[i][j] \leftarrow$ Max($M[i][j-1], M[i-1][j], M[i-1][j-1] + W[i][j]$) where $W[i][j] =$ SimpleTreeMatching($T'(i-1), T''(j-1)$)
    return M[m][n]+1
else
    return 0

2.4 XPath

XPath (*XML Path Language*) é definido atualmente como uma linguagem de consulta. Embora o principal propósito seja endereçar elementos de um arquivo XML, a linguagem pode ser usada também para HTML. XPath também possui algumas ferramentas que podem ser úteis para tratar alguns tipos de dados (strings, números e dados booleano). O foco deste trabalho é na ferramenta que o XPath proporciona para selecionar elementos na página HTML. Ele visualiza

o documento, XML ou HTML, como uma árvore DOM e então navega através da árvore para encontrar os elementos endereçados pelo usuário.

Para o contexto neste trabalho é importante ressaltar que existem duas possibilidades de seleção de elementos do documento HTML: a primeira forma é selecionar apenas um elemento em cada expressão, e a segunda é selecionar vários elementos com apenas uma expressão. Na Figura 2.3 podemos ver um exemplo que mostra a pequena diferença entre estas duas expressões de consulta. Na Figura 2.3 (a) a expressão XPath seleciona apenas um elemento. Na Figura 2.3 (b) a expressão XPath seleciona os dois elementos "*td*".

Para extrair um determinado elemento de uma página HTML, é necessário criar uma regra de consulta XPath, ou seja um caminho específico para aquele elemento. Se o elemento mudar de lugar ou for removido da página, o caminho configurado não irá encontrá-lo e precisará ser reendereçado.

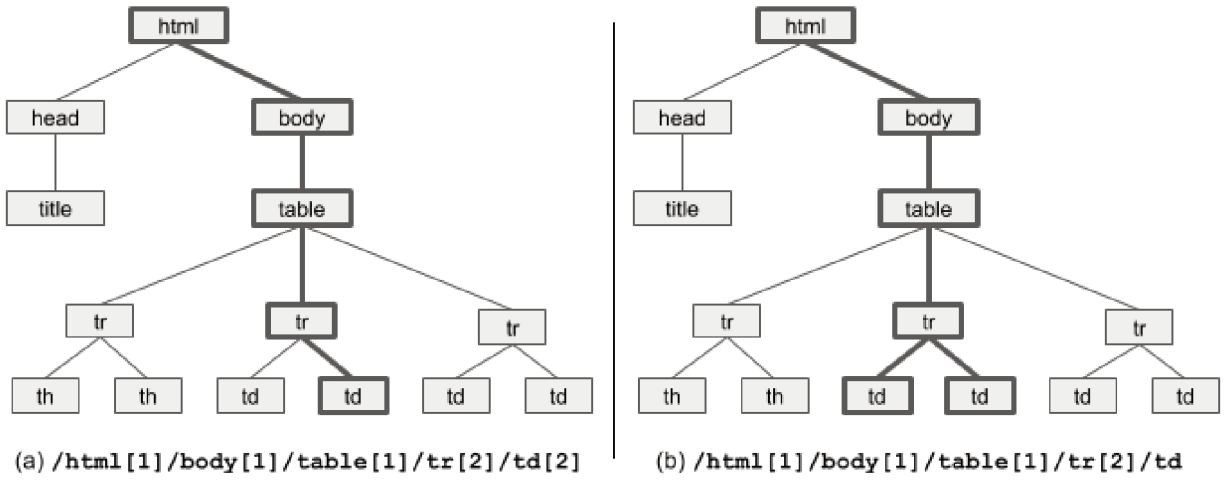


Figura 2.3 – Expressão (a) para selecionar apenas um elemento e (b) para *n* elementos.

3 TRABALHOS RELACIONADOS

Esta seção apresenta uma descrição dos principais trabalhos relacionados à manutenção de extratores de dados de páginas HTML. Vale ressaltar aqui que tal procedimento (manutenção de extratores) engloba a verificação dos dados extraídos e também a reparação do extrator (*wrapper*). Todo o esforço tem o intuito de garantir que os dados extraídos estejam corretos.

3.1 RAPTURE

O processo de manutenção de extratores consiste em basicamente duas etapas: verificação e reparação. Kushmerick (KUSHMERICK, 2000), que é um dos primeiros autores a contribuir para a solução do problema de manutenção de extratores, concentra seus esforços apenas na parte de verificação. Tal etapa é responsável por identificar mudanças na página HTML, verificando os dados extraídos ou a própria estrutura da página. Ou seja, este método não faz a reparação do extrator de dados, apenas identifica se houve alteração ou não.

O método denominado RAPTURE faz uma verificação nos dados extraídos com os dados já existentes, que podem ter sido extraídos por outro sistema ou então informado pelo usuário. Esse método faz a verificação partindo de uma fonte de dados relacional. O autor até sugere uma solução para lidar com dados semi-estruturados, mas essa solução, segundo o autor, não foi homologada. O autor não apresenta detalhes desta solução para dados semi-estruturados.

As primeiras informações extraídas são conferidas visualmente, já que o modelo de verificação será desenvolvido tomando como base a página web correta e conhecida pelo desenvolvedor. O sistema busca fazer uma comparação entre as informações extraídas com as que já existem, levando em conta um conjunto de características pré-definidas. Por exemplo: suponha que um *wrapper* extrai uma *string* “Brasil” de um página que lista países e seus respectivos códigos de DDI, e sabe que esta página, assim como a *string* “Brasil”, estão corretas. Em um próximo passo do *loop* de extração, o *wrapper* extrai a *string* “CLICK HERE! <href="http://www” para o nome de um país, em um página que ainda não foi verificada. Claramente o método irá identificar o dado como inválido.

O algoritmo RAPTURE mede a similaridade das duas strings, usando características como por exemplo, o número de caracteres, o número de palavras, pontuação, letras maiúsculas e minúsculas, etc. Após a verificação, o sistema tem uma medida de quão confiável é essa nova informação para tentar identificar se está correta e deve ser extraída, ou então deve interromper a

extração e tomar medidas, como por exemplo enviar uma mensagem a um sistema de reparação de extratores que esteja integrado com o RAPTURE.

O autor identifica que a grande maioria das vezes em que um *wrapper* para de funcionar, a informação extraída é totalmente diferente da informação esperada. Dificilmente, a *string* errada será parecida com a correta.

3.2 SG-WRAM

Este trabalho propõe uma abordagem de manutenção de *wrappers* baseado em um *schema* XML, na forma de um DTD, criado pelo usuário do sistema de extração. (MENG et al., 2002)

Apesar das mudanças nos documentos HTML, que podem ser de muitos tipos, algumas características principais são mantidas, tais como regras sintáticas, hiperlinks, e até mesmo comentários. Partindo deste princípio o método SG-WRAM funciona seguindo quatro passos básicos:

1. inicialmente obtém as características definidas pelo usuário no *schema* XML, as regras de extração anteriores e os resultados extraídos;
2. identifica os elementos alvos na página modificada com essas características;
3. agrupa os itens de acordo com o *schema* definido pelo usuário;
4. seleciona as instâncias significantes para reconfigurar a nova regra de extração que será usada na nova página.

A Figura 3.1 dá uma visão geral do processo de extração e manutenção do extrator: o *wrapper generator* fornece uma interface para o usuário informar um *schema XML*, um documento HTML e fazer um mapeamento entre eles. O sistema cria regras de extração para o documento HTML, as quais vão extrair os dados e gravar em outro arquivo XML. O arquivo é estruturado conforme definido no *schema XML*. O *wrapper engine* tem o papel de executar as regras criadas. Tendo o HTML e as regras, o *wrapper engine* executa e extrai as informações da página. Caso a regra falhe, ele informa ao *wrapper maintainer*, que automaticamente repara as regras para extrair da página modificada.

A Figura 3.2 mostra um exemplo de como é definido o *schema XML* que vai ser usado para estruturar as informações extraídas da página. Com base nesse *schema* o usuário cria uma

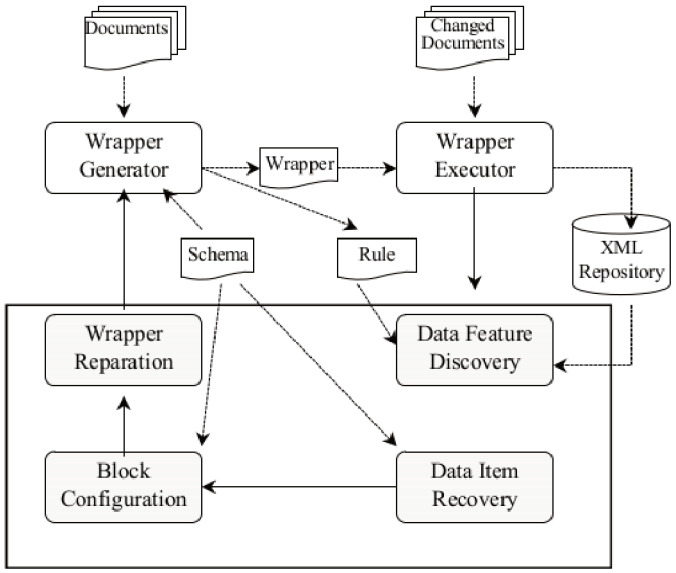


Figura 3.1 – Arquitetura do SG-WRAM (MENG et al., 2002)

correspondência entre a página HTML, através de informações úteis. Com essas informações e a estrutura dada no XML de *schema*, o sistema cria as regras de extrações no formato de *XQuery* (*XPath* por exemplo). O sistema então se mantém em *looping*, repetindo os quatro passos descritos anteriormente.

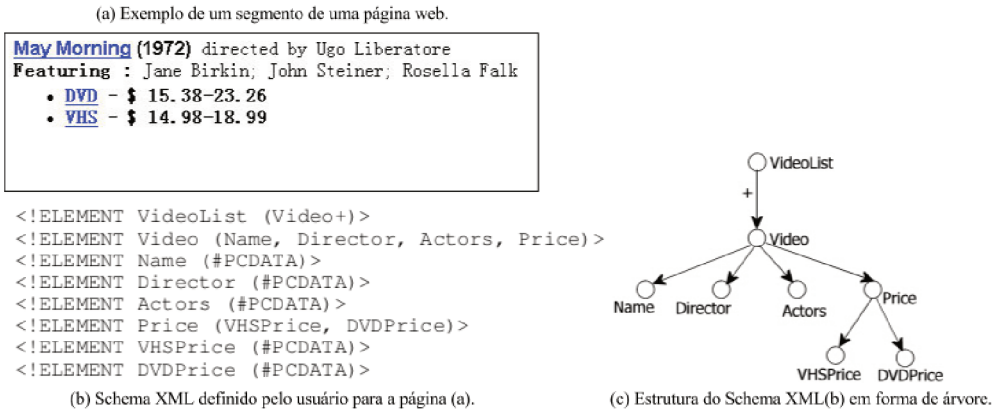


Figura 3.2 – Exemplo de Schema XML definido pelo usuário (MENG et al., 2002)

3.3 Clustered Tree Matching

O método utilizado para manutenção de extratores de dados em páginas HTML chamado de *Clustered Tree Matching*, é baseado na comparação da estrutura da árvore DOM da página HTML. Este método se fundamenta na melhoria do algoritmo *Simple Tree Matching* (Algoritmo 1) - que computa a distância entre duas árvores rotuladas, ou seja, calcular a similaridade entre

elas. Além de fazer uma pré seleção de elementos "candidatos" para posteriormente realizar a verificação de similaridade, com o objetivo de encontrar o mesmo elemento na página HTML modificada (FERRARA; BAUMGARTNER, 2011).

A partir do algoritmo de *Simple Tree Matching*, o autor desenvolve um novo algoritmo chamado *Clustered Tree Matching*. A alteração que ele faz é seguinte:

Algoritmo 2 ClusteredTreeMatching(T', T'')

```
1: {Change line 11 with the following code}
2: if m > 0 AND n > 0 then
3:     return M[m][n] * 1 / Max(t(T'), t(T''))
4: else
5:     return M[m][n] + 1 / Max(t(T'), t(T''))
```

Essa alteração faz com que o resultado da execução do algoritmo retorne um valor que seja mais parecido com o percentual de similaridade. Em outras palavras, o algoritmo vai retornar um número entre 0 e 1. Quanto mais perto de 1, maior é a similaridade das árvores comparadas. Nesta técnica o usuário define um limiar de similaridade, assim quando o sistema encontrar um valor que extrapole esse limiar, uma mensagem de erro é mostrada, e o sistema de extração para, evitando a extração de dados errados.

O autor usa esse algoritmo para fazer comparações entre sub árvores das árvores em questão, ao invés da árvore inteira. Para selecionar essas sub árvores o autor utiliza a estratégia descrita a seguir.

A ideia principal é comparar algumas informações úteis armazenadas pela extração da versão original da página web e realizar uma busca por similaridades na nova página, independente do método usado para extração (*XPath* por exemplo). As informações consideradas úteis pelo autor são atributos das *tags* HTML, a própria *tag* HTML, sub-árvores e que sejam descendentes da mesma raiz. Esse elementos serão chamados de candidatos. Dentre esses, o que apresentar o maior grau de similaridade provavelmente representa a nova versão do elemento original que estava sendo extraído. Depois desta verificação feita, pode-se simplesmente reconfigurar o XPath (ou o método usado para extração), fazendo apontar para este novo elemento.

Vale ressaltar que a comparação de sub-árvores do elemento, e a comparação do nodo raiz deste mesmo elemento são uteis para caso a alteração seja maior na estrutura da página, por exemplo, onde o elemento <*table*> é substituído por <*div*>.

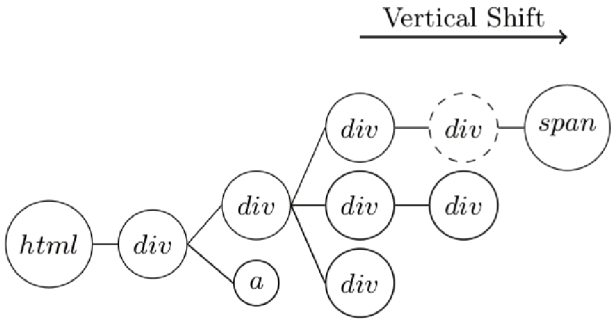
O autor não entra em muitos detalhes de como é feita a seleção dos candidatos, pois sua pesquisa foi desenvolvida em cima de uma software comercial de extração de dados.

3.4 XTPath

Este método utiliza uma técnica para identificar mudanças no HTML da página que armazena a estrutura contextual ao elemento alvo a ser extraído. Com essa estrutura armazenada, quando a extração para de funcionar, o algoritmo é ativado. E então ele faz uma varredura em cada item armazenado anteriormente comparando com a estrutura da nova página HTML (COHEN; DING; BAGHERJEIRAN, 2015).

Primeiramente o autor define e categoriza os principais tipos de mudança que podem ocorrer em estrutura HTML. Segundo o autor, quando o código HTML é modificado, a sua árvore estrutural também é alterada, o que é chamado de *shift*. Esta mudança ocorre devido a uma inclusão alteração ou remoção de algum elemento na estrutura DOM em questão. As mudanças que podem ocorrer na estrutura da árvore podem fazer com que o *wrapper* pare de funcionar, ou então extraia dados incorretos. O autor ainda separa as mudanças (*shift*) em duas categorias, são elas: *horizontal shift* e *vertical shift*.

Vertical Shift ocorre quando uma mudança causa uma alteração no tamanho do caminho da raiz da árvore até o elemento relevante a ser extraído. Isso acontece quando é feita uma inserção ou remoção na árvore. Para ajustar o *XPath* neste caso, é necessária uma alteração no tamanho do *Path*, incluindo ou removendo o elemento em questão. A Figura 3.3 dá um exemplo de *vertical shift* e um destaque para a alteração necessária no *XPath*.

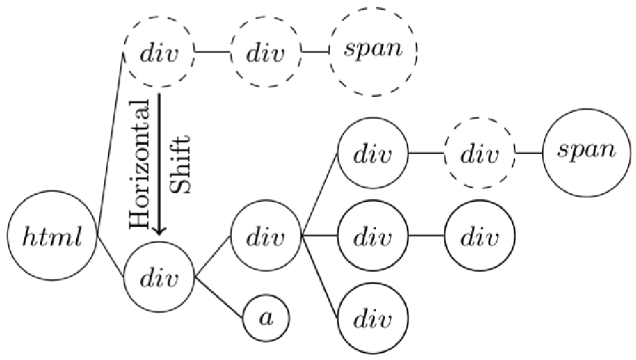


Target	Figure	XPath
Name	Before	/html/div[1]/div[1]/div[1]/span
	After	/html/div[1]/div[1]/div[1]/div/span

Figura 3.3 – Exemplo de *vertical shift* e a alteração necessária no XPath. (COHEN; DING; BAGHERJEIRAN, 2015)

Horizontal shift ocorre quando um elemento irmão de algum elemento no caminho é inserido ou removido. Este tipo de alteração pede uma correção do caminho do *XPath*, em que o índice de algum ou alguns elementos sejam alterados. A Figura 3.4 mostra um exemplo de

horizontal shift e um destaque para a alteração necessária no XPath.



Target	Figure	XPath
Name	Before	/html/div[1]/div[1]/div[1]/div/span
	After	/html/div[2]/div[1]/div[1]/div/span

Figura 3.4 – Exemplo de *horizontal shift* e a alteração necessária no XPath. (COHEN; DING; BAGHERJEIRAN, 2015)

3.4.1 O Algoritmo **XTPath** (XPath + Tree Paths)

Ao invés de consultar um conjunto de treinamento grande e complexo para identificar as mudanças na página, este método armazena a estrutura da árvore que está ao redor do elemento, ou seja, a estrutura contextual ao elemento alvo do *wrapper*.

Para facilitar o entendimento do funcionamento do método XTPath, algumas definições devem ser feitas. Uma delas é sobre o funcionamento e a maneira como o autor define a sua árvore chamada de Tree Path. Outra é sobre o LCA.

LCA (*Last Common Ancestor*) é um elemento que existe em cada caminho da raiz da árvore DOM até o elemento alvo. O LCA é único para cada elemento alvo.

Tree Path é uma sequência de árvores DOM, que estão contidas em um HTML, e que cada elemento dessa sequência é um caminho do LCA até o elemento alvo. Uma Tree Path é uma extensão do XPath, no XPath cada elemento do caminho é o nome da tag HTML, enquanto que no Tree Path cada elemento do caminho é a subárvore que vai de cada elemento, partindo da raiz, até o elemento alvo. A Figura 3.5 mostra um exemplo de uma Tree Path de tamanho 4 fazendo uma relação com o XPath normal.

Dadas as definições, o algoritmo, basicamente, funciona da seguinte maneira: O primeiro passo do algoritmo é a tentativa de extrair a informação utilizando o caminho original, ou seja, navegar na árvore usando a *XPath* original para chegar no elemento a ser extraído. Como esse passo pode resultar em um erro, pois a estrutura da página HTML, e consequentemente

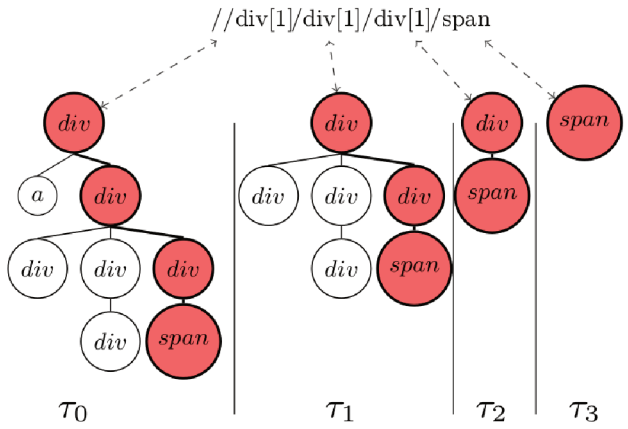


Figura 3.5 – Representação gráfica de um *Tree Path* em comparação com um XPath normal. (COHEN; DING; BAGHERJEIRAN, 2015)

o caminho até o elemento, foram alterados, então o recuperador de *wrapper* é acionado. A recuperação faz uma navegação na lista de subárvores (*Tree Path*), comparando cada elemento do HTML DOM com cada subárvore da lista. Para calcular essa similaridade, é usado o algoritmo *html_tree_match*, que é baseado no *SimpleTreeMatching*. O resultado do algoritmo é um número inteiro que indica o quão similar é determinado elemento. Então o mais similar é substituído pelo antigo, desta forma ajustando o XPath para que volte a funcionar e extrair os dados corretos.

Este método de reparação de extrator de dados, serve como um complemento aos métodos que utilizam estratégias baseadas em XPath. O XTPath não substitui qualquer outra estratégia de reparação que já esteja sendo utilizada, ele auxilia, entrando em ação quando um XPath falha.

3.5 MAVE (Multilevel wrApper Verification systEm)

O objetivo principal do MAVE é verificar os dados extraídos pelo *wrapper*. Para tal, o MAVE usa modelos de predição para validar os dados extraídos. O modelo de predição é gerado a partir dos dados inicialmente extraídos e validados manualmente, ou seja um conjunto de dados validados por um humano. Utilizando os modelos de predição, o MAVE calcula a similaridade entre as informações e consequentemente, alerta ao usuário se os dados extraídos diferem do esperado.

O processo de verificação de dados segue um fluxograma padrão, conforme mostrado na Figura 3.6. As alterações que o MAVE propõe, em relação a outros métodos de verificação

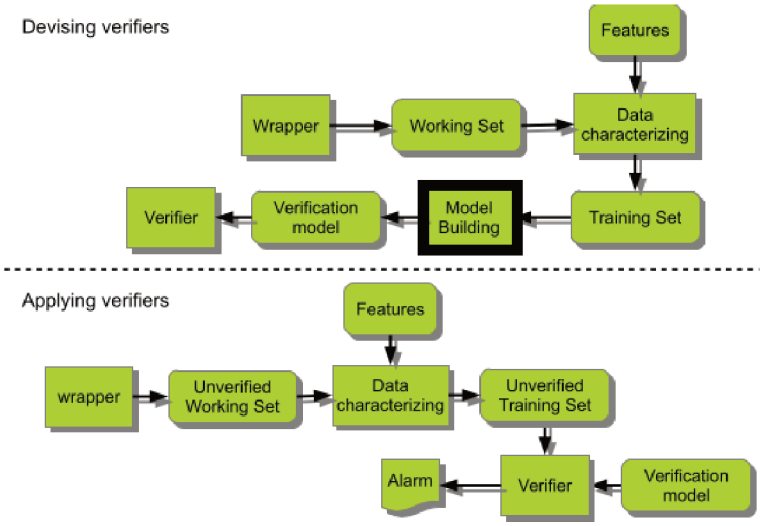


Figura 3.6 – Fluxograma do processo geral de verificação (VIANA et al., 2016).

de dados extraídos, estão concentradas na parte da criação do modelo de verificação (*Model Building*), destacado na Figura 3.6, e na aplicação desse modelo.

Com o objetivo de mostrar como funciona o processo de verificação de dados, será explicado cada passo apresentado no fluxograma da Figura 3.6.

A parte de cima da linha pontilhada apresenta o fluxograma do processo de criação do modelo de predição (*Devising-Verifiers*). A seguir será detalhado cada parte deste processo.

- ***Wrapper***: esse item é responsável por extrair os dados da pagina HTML e salvá-los de forma estruturada. Gerando o *Working Set*;

- ***Working Set***: esse item é o conjunto de dados estruturado gerado pelo *Wrapper*. nesta etapa, que é o processo de criação do verificador, esse conjunto de dados é verificado por um usuário. Esses dados são enviado para o processo de caracterização;

- ***Data Characterizing:*** os dados verificados são então transformados em vetores de valores, em que cada valor é a representação de cada característica do conjunto de características (***Features***). Esse conjunto de vetores é chamado de ***Training Set***;

- ***Model Building***: aqui é o onde o modelo de predição é criado, utilizando os valores que estão no *Training-Set*. Por fim é gerado um modelo de verificação (***Verification Model***), que por sua vez resulta em um verificador (***Verifier***).

A parte de baixo da linha pontilhada é onde o processo de verificação de novos dados acontece, ou seja, onde é aplicado o verificador (*Applying Verifiers*). A seguir será detalhado

cada passo deste processo:

- ***Wrapper***: da mesma forma que no processo de criação do verificador, o *wrapper* é responsável por extrair os novos dados e gravar de forma estruturada, gerando o *Working Set*, desta vez não verificado por um humano, pois será aplicado o verificador criado anteriormente;

- ***Data Characterizing***: da mesma forma como é feito no processo de criação do verificador, este passo é responsável por gerar os valores das características de cada dados extraído. Esse conjunto de valores que representa o conjunto de dados é chamado de ***Unverified Training-Set*** ou *Test-set*. Neste ponto, o *Training-Set* ainda não foi verificado;

- ***Verifier***: este passo é responsável por aplicar o modelo de verificação no ***Unverified Training-Set***. Caso os dados do conjunto sejam classificados como incorretos, então um alerta (***Alarm***) é mostrado ao usuário.

O processo de criação do modelo de verificação e a aplicação desse modelo, é feito em dois níveis. A seguir será explicado o primeiro nível do MAVE. Nessa subseção é explanado o processo de criação do verificador e também como este é aplicado para fazer a verificação.

3.5.1 Primeiro nível: Características Não-Numéricas

O processo de criação de um verificador de *wrapper* começa obtendo um conjunto de dados extraídos. Esse primeiro conjunto de dados é previamente verificado e validado por um usuário. Isso é feito para garantir a validade dos dados e para poder ensinar o algoritmo o que é um dado esperado. Os dados brutos extraídos pelo *wrapper* são chamados pelo autor de *working-set* (WS). A partir do WS é gerado um outro conjunto de dados, chamado de *training-set* (TS). O TS é o conjunto onde estarão os valores das características dos dados do WS.

A imagem 3.7 mostra um exemplo de um WS e o seu referido TS. Nesta imagem, cada coluna f_i representa uma característica do *Attribute(a)(a informação extraída)* do *Slot(s)*. No exemplo da figura, a coluna f_1 significa a quantidade de palavras, f_2 a quantidade de palavras que começam com letra maiúscula. f_3 diz se a *string* tem pontuação (*true*) ou não (*false*), f_4 indica a cor da *string* e f_5 vai ser *true* quando a primeira letra começa com maiúscula, caso contrário será *false*. Neste exemplo, tem-se três características não numéricas: f_3, f_4 e f_5, e duas características numéricas: f_1 e f_2.

	Slot (s_i)	
i	**Attribute (a_i)**	**Label (l_i)**
1	*Distributed Reasoning for Multiagent Simple Temporal Problems*	*Title*
2	*The Arcade Learning Environment: An Evaluation Platform for General Agents*	*Title*
3	*Learning by Observation of Agent Software Images*	*Title*
4	*Networks of Influence Diagrams: A Formalism for Representing Agents*	*Title*

(a) List of slots contained in the working set *ws:WS*

	Slot (s_i)					
i	$\vec{x}_i$					**Label (l_i)**
	f_1	f_2	f_3	f_4	f_5	
1	7	6	false	blue	true	Title
2	10	9	true	black	true	Title
3	7	5	false	black	true	Title
4	9	7	true	green	true	Title

(b) Training set obtained from the working set *ws:WS*

Figura 3.7 – Conjunto de trabalho (a) e conjunto de treinamento (b) (VIANA et al., 2016).

Neste primeiro nível de verificação, o MAVE utiliza apenas as características não numéricas, a partir das quais o método gera uma assinatura. Esta assinatura serve para decidir qual característica não numérica é relevante para fazer a verificação. Ela é obtida fazendo uma comparação de cada características não numérica (f_3, f_4, f_5) entre todos os *slots* do TS. Por exemplo, cada *slot* de f_3 (ou seja, linha) é comparada com os outros valores de f_3 dos demais *slots*. Se todos os *slots* tem o mesmo valor, então a assinatura para o f_3 é o próprio valor da característica. Se caso os valores não forem todos iguais, então o valor da posição correspondente na assinatura será -1. Outro exemplo, é o *slot* f_5, em que todos são *true*, logo a posição correspondente ao f_5 na assinatura vai ser *true*. Na Figura 3.7(b) a assinatura do TS é a seguinte: [-1, -1, *true*].

Uma vez obtida a assinatura, a verificação dos próximos WS analisados pelo verificador, é feita da seguinte forma: faz-se uma comparação entre a característica não numérica correspondente à posição da assinatura. Por exemplo, na Figura 3.8 as características não numéricas do primeiro *slot* são: [*false*, *black*, *false*] que será comparada a assinatura [-1, -1, *true*]. Como as duas primeiras posições da assinatura são -1, então não é feita a comparação destas mesmas características com o *slot* 1, já a comparação da terceira característica vai retornar false pois "*true* = *false*" é falso. No segundo *slot* temos: [*true*, *blue*, *true*], nesse caso a última característica retorna *true* pois "*true* = *true*" portanto o *slot* 2 é classificado com válido nesse primeiro nível. Mas para o WS ser válido, todos os *slots* precisam ser válidos. Neste caso, o WS será invalidado e reportado ao usuário.

Note que é comum existir dois dados que contenham o mesmo tipo, por exemplo o título de um livro e o autor do livro, ambos os campos são uma *string*. Se aplicar uma verificação à informação extraída do tipo *string* e o verificador retornar *false*, assume-se que o dado não é nem um título e nem um autor. No entanto, ao aplicar a verificação e retornar *true*, não se sabe se o dado é um título ou um autor. Neste caso, é criado um modelo de verificação mais complexo, com outras características e que possibilite saber o que é título e o que é autor, neste exemplo. Por isso um segundo nível de verificação é adicionado, neste, características numéricas, por exemplo o tamanho da *string*, são analisadas. Para exemplificar, no trabalho são definidas duas características numéricas: a quantidade de palavras na *string* e a quantidade de palavras que começam com letra maiúscula.

	Slot (s_i)	
i	**Attribute** (a_i)	**Label** (l_i)
1	*a Survey of Multi-objective Sequential decision-making*	*Title*
2	*AI Methods in Algorithmic Composition: A Comprehensive Survey*	*Title*

(a) List of slots contained in an unverified working set *ws:WS*

	Slot (s_i)					
i	$\vec{x}_i$					**Label** (l_i)
	f_1	f_2	f_3	f_4	f_5	
1	6	3	false	black	false	Title
2	8	7	true	blue	true	Title

(b) Unverified training set

Figura 3.8 – Conjunto de trabalho(a) e conjunto de treinamento(b) (VIANA et al., 2016).

Na subseção a seguir será explanado o segundo nível do MAVE. Apresentando a forma como o modelo de verificação é gerado e aplicado.

3.5.2 Segundo nível: Características Numéricas

Os conjuntos de dados que passaram pelo primeiro nível, ou seja pela verificação baseada na assinatura, serão verificados através de um algoritmo de classificação baseado em uma classe. Esse tipo de algoritmo de classificação utiliza a estratégia não-supervisionada para fazer o treinamento. Somente as características numéricas serão utilizadas para a verificação neste nível. No exemplo da Figura 3.8, se considerarmos o *slot* 2 = [8, 7, *true*, *blue*, *true*], serão utilizadas somente as características das duas primeiras posições [8,7]. Se o modelo de predição rotular esse *slot* como sendo da classe alvo, então ele é válido, caso contrário será invalidado.

Para simplificar o processo de verificação, pode-se definir em uma função que retorna 1 para o conjunto de treinamento válido e 0 caso contrário. Esta função verifica se o conjunto de treinamento, que não foi validado manualmente (*Unverified Training Set*), se assemelha com o modelo de verificação (*Verifier Model*) criado pelo algoritmo. O nível de semelhança, ou seja, o limiar para definir o que vai ser considerado como um TS válido é definido pelo usuário. A função pode ser escrita como:

$$V(uts, vm) : TS \times VM \rightarrow \{0, 1\}$$

onde uts é o conjunto não verificado e vm é o modelo de verificação. As funções mais populares são baseadas na distância Euclidiana, distribuição Gaussiana, e distribuição Gaussiana normalizada.

Cada TS será visualizado como um conjunto de pontos igualmente rotulados, em um espaço de dimensão P. A ideia do MAVE é treinar um algoritmo classificador de uma classe para calcular um limite ao redor dos pontos do TS, tal que aceite o maior número de *slots* similares ao TS. Bem como, minimize a chance de aceitar *slots* incorretos.

Os algoritmos classificadores chamados de *One-Class Classifier* são utilizados para os casos em que não se tem quase nenhum exemplo negativo (exemplos raros) para utilizar no seu treinamento. No entanto, no teste, exemplos negativos podem estar presente no *dataset*.

Quando o *wrapper* está em execução, o verificador MAVE é responsável por analisar se todos os objetos extraídos estão devidamente corretos. Se caso os dados se assemelham, respeitando o limiar definido pelo usuário, os dados são validados e armazenados, caso contrário, um alerta é exibido ao usuário.

3.5.3 Experimentos e resultados

A base de dados utilizada pelo MAVE, para executar os experimentos, possui 23.416 páginas de 27 sites diferentes, com um total de 290.000 *slots* foram extraídos. O processo de teste foi dividido em três fases:

- Fase 1: é destinada a testar se as variações no tratamento dos dados, do algoritmo MAVE melhoram as técnicas de classificação de uma classe. Isto é, o autor verifica se a ideia de lidar com características não numéricas e numéricas independentemente melhoram o processo de classificação dos dados, consequentemente melhorando o processo de verificação. A Figura 3.9 lista os algoritmos utilizados;

- Fase 2: depois de verificado que a performance das variações do MAVE melhoram os resultado, foi identificado qual a melhor variação;
- Fase 3: por último, técnicas tradicionais de verificação foram aplicadas, e os resultados foram comparados com os resultados alcançados através das variações do MAVE.

Algorithm	Description
G	Normal Gaussian One Class Classifier density
P	Parzen One Class Classifier
KC	K-center One Class Classifier
KNN	Nearest neighbour One Class Classifier
KM	K-means One Class Classifier
SVM	Support Vector Machine One Class Classifier
PCA	Principal Component Analysis One Class Classifier
NN	Auto-encoder One Class Classifier

Figura 3.9 – Resultados de todos os algoritmos estudados (VIANA et al., 2016)

Ao total foram testados 19 algoritmos e cada um deles foi submetido a verificar 290.000 *slots*. Para cada conjunto de treinamento, a *label* que representa o *slot* é chamado de classe alvo, enquanto as demais *labels* são classificadas como *outlier*.

Para avaliar a performance de cada abordagem, o autor utilizou a técnica de validação cruzada com 10-fold. Além disso, em cada teste foram incluídos os demais *slots* de outras *labels*, ou seja, incluído alguns *outliers* extras.

Por fim, o MAVE conquistou os melhores resultados com os seguintes algoritmos de classificação: KNN, G, KM e KC, conforme mostra a tabela da Figura 3.10

Alg	AUC	ACC	R	P	Alg	AUC	ACC	R	P	Alg	AUC	ACC	R	P
G	0.93	0.92	0.97	0.82	**M-G**	0.95	0.94	0.97	0.87	**RAP**	0.84	0.90	0.77	0.82
P	0.73	0.88	0.57	0.81	**M-P**	0.78	0.90	0.64	0.86	**LER**	0.54	0.41	0.77	0.24
KC	0.92	0.90	0.97	0.76	**M-KC**	0.94	0.94	0.97	0.84	**MCC**	0.55	0.32	0.90	0.25
KNN	0.92	0.89	0.80	0.74	**M-KNN**	0.95	0.93	0.87	0.83					
KM	0.92	0.90	0.98	0.77	**M-KM**	0.95	0.94	0.98	0.85					
SVM	0.82	0.91	0.72	0.87	**M-SVM**	0.90	0.92	0.75	0.89					
PCA	0.92	0.90	0.84	0.80	**M-PCA**	0.92	0.90	0.84	0.80					
NN	0.92	0.91	0.85	0.82	**M-NN**	0.92	0.89	0.83	0.79					

Figura 3.10 – Resultados de todos os algoritmos estudados (VIANA et al., 2016)

4 MÉTODO PROPOSTO

Conforme descrito neste trabalho, o processo de manutenção de extratores de dados de páginas web é dividido em duas partes. A primeira etapa tem como objetivo a verificação de erros ou inconsistência. Na segunda etapa, é realizada a reparação do extrator, quando possível, para a continuação da extração dos dados.

A identificação de erros pode ser feita de várias formas. Uma das formas é a verificação dos dados extraídos das páginas HTML. O foco deste trabalho é no método de verificação utilizando os dados extraídos pelo *wrapper*.

A primeira etapa do estudo sobre *wrapper verification* foi identificar e encontrar soluções para verificação de dados extraídos. No capítulo 3 (Trabalhos Relacionados) são apresentadas algumas destas soluções. Diversas propostas foram identificadas na literatura, porém, as mais relevantes foram descritas neste trabalho. Algumas das soluções são: RAPTURE (KUSHMERICK, 2003) MAVE (VIANA et al., 2016).

Entre os trabalhos estudados, foi identificado o potencial da proposta MAVE (Seção 3.5) por estar entre os métodos no estado-da-arte da extração e verificação de dados. Devido a isto, a seções a seguir descrevem as principais melhorias que foram propostas utilizando como base o método MAVE.

4.1 FAVE (Feature wrApper Verification systEm)

Esta seção tem por objetivo descrever as mudanças que foram feitas no método MAVE (explicado na Seção 3.5). O método proposto, chamado de FAVE, tem como objetivo aprimorar os resultados já alcançados pela proposta original. Note que a principal melhoria do MAVE, em relação a outros trabalhos de extração de dados, é a separação da verificação das características dos dados. Enquanto um nível faz a verificação somente de características não numéricas, outro nível faz a verificação de características numéricas. A seguir será descrito o funcionamento do MAVE e as melhorias propostas.

No primeiro nível do processo de verificação do MAVE, é utilizada a assinatura que foi gerada no processo de construção do modelo de verificação. O processo de construção da assinatura faz com que os *slots* sejam analisados levando em conta inicialmente somente características não-numéricas. No processo de criação da assinatura (apresentado na Seção

3.5.1), somente são verificados os campos que possuem valores iguais para todos os *slots*. Na etapa de verificação, o *slot*, que não obedecer a assinatura será considerado um *slot* incorreto.

Com o intuito de melhorar o processo de verificação, algumas mudanças foram feita,s partindo da ideia apresentada no MAVE.

A principal mudança no FAVE é a que nenhuma característica não-numérica é descartada. Desta forma todas as características são utilizadas para criar o modelo de predição e também todas são utilizadas no processo de verificação dos dados. Com isso, evita-se que um *slot* seja classificado com incorreto por apresentar apenas uma característica não-numérica diferente dos demais *slots*.

No segundo nível do processo de verificação, o MAVE utiliza apenas as características numéricas para treinar o algoritmo de classificação que construirá o modelo de predição. No FAVE, foi definido utilizar todas as características (numéricas e não-numéricas) no processo de treinamento do algoritmo de classificação. As características não-numéricas foram quantificadas para 0 (*false*) e 1 (*true*). Nessa etapa, duas técnicas de classificação de dados foram avaliadas: utilizando algoritmos de classificação de apenas uma classe e também algoritmos de classificação de duas ou mais classes.

As características utilizadas pelo MAVE foram propostas em outros trabalhos [(BRONZI et al., 2013), (CHIDLOVSKII; ROUSTANT; BRETTE, 2006), (KUSHMERICK, 2000), (CRESCENZI; MECCA, 2004), (CRESCENZI et al., 2001), (VIANA et al., 2011), (LERMAN; MINTON; KNOBLOCK, 2003)]. Ou seja, o objetivo do MAVE não foi propor novas características mas sim reutilizar de outros trabalhos. O FAVE segue o mesmo modelo, replicando as mesmas características.

Unindo as ideias de características encontradas em outros autores, com características selecionadas neste trabalho, um total de 24 características são utilizadas: 12 não numéricas, e 12 numéricas. A Tabela 4.1 apresenta todas as características separadas pelo seu tipo. Já a Tabela 4.2 apresenta exemplos de possíveis valores para cada característica.

ID	Característica
C1	Contém pontuação (.,;:!?)
C2	Primeiro caractere em maiúsculo
C3	Todas as palavras começam com maiúsculo
C4	Contém ponto (.)
C5	Contém vírgula (,)
C6	Contém ponto e vírgula (;)
C7	Contém dois pontos (:)
C8	Contém ponto de interrogação (?)
C9	Contém ponto de exclamação (!)
C10	Contém caracteres maiúsculos
C11	Contém dígitos
C12	Contém caracteres diferente de dígitos
N1	Quantidade de dígitos
N2	Quantidade de caracteres diferente de dígitos
N3	Quantidade de caracteres
N4	Quantidade de palavras
N5	Quantidade de palavras que começam com letra maiúscula
N6	Relação de palavra por caractere
N7	Relação de dígito por caractere
N8	Relação de pontuação por caractere
N9	Relação de letras por caractere
N10	Relação de letras maiúsculas por caractere
N11	Relação de letras minúsculas por caractere
N12	Tamanho médio de palavras

Tabela 4.1 – Características utilizadas. *IDs* precedidos da letra 'C' representam características não numéricas. *IDs* precedidos pela letra 'N' representam características numéricas

Característica	Valor	Característica	Valor
C1	1 (*true*)	N1	1
C2	1 (*true*)	N2	40
C3	0 (*false*)	N3	41
C4	1 (*true*)	N4	9
C5	0 (*false*)	N5	5
C6	0 (*false*)	N6	0.2195
C7	0 (*false*)	N7	0.0243
C8	0 (*false*)	N8	0.0243
C9	0 (*false*)	N9	0.7317
C10	1 (*true*)	N10	0.1219
C11	1 (*true*)	N11	0.6097
C12	1 (*true*)	N12	3.6666

Tabela 4.2 – Exemplos de valores utilizando as características no seguinte exemplo de um título de livro: '*Harry Potter e a Pedra Filosofal - Vol 1.*'

4.2 Código-Fonte

Esta seção apresenta as ferramentas e bibliotecas que foram utilizadas no desenvolvimento do código fonte.

Todos as etapas do processo foram codificadas utilizando a linguagem de programação Python. Para trabalhar com os algoritmos de classificação (OneSVM, KNN, IF, G) foi utilizado a biblioteca **scikit-learn**[1]. Esta possui uma grande quantidade de ferramentas para algoritmos de aprendizado de maquina, alem de possuir todas as ferramentas necessárias para fazer a leitura dos arquivos da base de dados.

Todos os códigos das ferramentas utilizadas para os testes estão disponível publicamente.[2]

[1] http://scikit-learn.org
[2] https://github.com/joaogehlen91/FAVE

5 EXPERIMENTOS E RESULTADOS

Neste capítulo, serão apresentados os resultados obtidos de acordo com os cenários que foram montados para cada experimento. Serão detalhados os algoritmos que foram utilizados para fazer a classificação dos dados, conforme proposto pelo método de verificação. A Seção 5.1 tem como objetivo mostrar a base de dados que foi utilizada para fazer os testes. Já as Seções 5.3.1 e 5.3.2 apresentam os resultados de cada técnica de classificação de dados.

5.1 Base de dados

A base de dados proposta por (HAO et al., 2011) e que pode ser obtida por *download* através do site (HAO, 2011), foi utilizada para fazer a simulação dos dados extraídos por um *wrapper*.

A base é composta por uma coleção de páginas reais da web, que estão separadas em oito assuntos diferentes. Em cada um destes 8 domínios existe um conjunto de 10 *websites*. De cada *website* foram coletadas em média 1800 *webpages*, todas em formato de arquivo '*.htm*'. De cada página HTML foram extraídos em média 4 atributos, por exemplo: nome, telefone, modelo, marca, etc. dependendo do domínio do site, conforme apresenta a Tabela 5.1.

Assunto	**Sites**	**Páginas**	**Nº Atributos**	**Atributos**
Auto	10	17923	4	modelo, preço, motor, consumo
Book	10	20000	5	título, autor, ISBN, editora, data publicação
Câmera	10	5258	3	modelo, preço, fabricante
Job	10	20000	4	título, empresa, local, data postagem
Movie	10	20000	4	título, direto, gênero, classificação
NBA Player	10	4405	4	nome, time, altura, peso
Restaurant	10	20000	4	nome, endereço, telefone, cozinha
University	10	16705	4	nome, telefone, website, tipo

Tabela 5.1 – Base de dados.

A base de dados utilizada neste trabalho possui um gabarito separado por assunto onde consta a informação correta extraída de cada página, relacionado com cada atributo. Cada assunto no gabarito possui em média 40 arquivos do tipo '.txt', pois existe um arquivo para cada atributo de cada página. Por exemplo, a parte do gabarito em que se trata de *NBA Players* possui 40 arquivos, um arquivo para cada atributo de cada site. Por exemplo o arquivo 'nbaplayer-espn-height.txt' possui todas as alturas extraídas das páginas da ESPN. Cada linha do arquivo possui

as informações de identificação da página, a quantidade de atributos extraídos, e o valor do atributo, que neste exemplo é a altura do jogador.

5.2 Métricas Utilizadas

Para avaliar a eficácia de cada abordagem, foram selecionadas três métricas mais tradicionais: acurácia, precisão, revocação e *F-measure*(MANNING et al., 2008).

A acurácia (ACC) é a medida mais utilizada para avaliar qual a frequência em que o classificador está correto, ou seja, mostra o percentual de exemplos que foram classificados corretamente.

Precisão (P) é a porcentagem de amostras positivas classificadas corretamente, sobre o total de amostras classificadas como positivas, ou seja, dos exemplos classificados como positivos, quantos, de fato, eram.

Revocação (R) é a porcentagem de amostras positivas classificadas corretamente, sobre o total de amostras positivas, ou seja, quando o exemplo é positivo com que frequência é classificado como positivo.

Foi utilizado, ainda, a medida *F-Measure* (F1), que representa a média harmônica entre precisão e revocação.

Para validar as comparações entre os métodos, foi utilizado o teste *Student's T-Test* (*t-test*) na métrica F1. Esse teste é feito sobre dois conjuntos de dados e o seu resultado é um número, entre 0 e 1, que mede a confiança de uma afirmação. Neste trabalho, as afirmações que são passíveis de validação são de que uma proposta obteve um melhor resultado do que a outra em determinado teste. Assim, a Equação 5.1 traz as hipóteses que serão levantadas nas seções dos experimentos. Na hipótese H_1, desta equação, tem-se que as duas propostas comparadas são iguais caso o resultado do *t-test* for $\alpha > 0,05$. E, na hipótese H_2 desta mesma equação, pode-se afirmar que uma proposta foi melhor ou pior que a outra em um determinado aspecto.

$$\begin{cases} H_1 : MAVE = FAVE, & \text{se } \alpha > 0,05 \\ H_2 : MAVE \neq FAVE, & \text{se } \alpha < 0,05 \end{cases} \tag{5.1}$$

Em resumo, como o valor de α foi definido em $0,05$, ao validar-se a hipótese H_2 da Equação 5.1, pode-se afirmar com 95% de confiança que a FAVE obteve um resultado médio diferente de MAVE.

5.3 Execução dos experimentos

Foi adotada a mesma metodologia do MAVE (VIANA et al., 2016), onde o autor separa os testes por diferentes assuntos. Isto é, em cada domínio de conteúdo é escolhido um determinado tipo de informação para ser classificada. Por exemplo, tomando como base todos os sites de câmeras fotográficas, o objetivo pode ser identificar o que é um modelo de câmera. Como o gabarito da base de dados já está separado por tipo de informação (preço, fabricante, etc.), o MAVE rotula os dados que são modelo de câmeras com 'positivo' e os demais dados, dos mesmos sites de câmeras (preço, fabricante) são rotulado como dados 'negativos'. Isso é feito para treinar o algoritmo de classificação. Com essa base de dados preparada os testes são executados.

Neste trabalho além de utilizar apenas os demais dados do mesmo assunto, foram utilizados dados de outros domínios (outros sites parecidos com os dados alvo do classificador). No exemplo de modelos de câmeras, ao invés de rotular o preço e o fabricante como 'negativo' mais dados foram incluídos, como por exemplo, títulos de livros, que neste caso é mais parecido com modelos de câmeras do que o preço ou fabricante.

A Subseção 5.3.1 apresenta cenários de teste em que utilizou-se algoritmos de classificação de duas ou mais classes, enquanto que na Subseção 5.3.2 são cenários em que utilizou-se algoritmos de classificação de uma classe. Tendo isso definido, cada algoritmo foi executado 10 vezes para se obter um valor médio de acurácia (ACC), precisão (P), revocação (R) e F-Measure (F1).

Todos os resultados dos experimentos e discussões sobre os mesmos serão apresentados na Seção 5.4.

5.3.1 Algoritmos de Classificação de duas ou mais classes

Essa subseção tem o objetivo de apresentar os resultados obtidos utilizando algoritmos de classificação de duas ou mais classes. Esses algoritmos utilizam o método supervisionado para o aprendizado e criação do modelo de predição.

De acordo com os resultados do MAVE (VIANA et al., 2016) e apresentados na Seção 3.5.3, foram escolhido os algoritmos KNN (*K-Nearest Neighbour*) e G (*Normal Gaussian*). Ambos partem do princípio que os dados de treinamento serão rotulados e terão mais de uma classe, por exemplo a classe alvo e os *outliers*. A classe alvo é utilizada para apontar qual dado

é correto, ou seja, dados positivos, já a classe *outlier* é utilizada para dizer para o algoritmo o que é um dado incorreto, ou seja, dados negativos. A seguir será apresentado alguns cenários de testes.

5.3.1.1 Cenário 1: Identificação de modelos de câmeras

Neste cenário de teste os registros foram obtidos da base de dados e rotulados da seguinte forma:

- Dados positivos: são todos os registros de modelo de câmera extraídos dos sites cujo assunto é Câmera.
- Dados negativos: são todos os registros dos sites cujo assunto é Carro, ou seja, modelo de carro, preço motor e consumo.

Com um total de 76.950 registros, 5.258 são dados positivos e 71.692 são dados negativos. Foram feitos testes com diferentes tamanhos do conjunto de treinamento. A Tabela 5.2 apresenta os resultados destes testes.

		MAVE				FAVE				
Treinamento	**Algoritmo**	**ACC**	**P**	**R**	**F1**	**ACC**	**P**	**R**	**F1**	**T-Test (F1)**
1%	G	94,59	56,36	93,42	70,28	92,83	52,91	97,79	67,44	3,94E-01
	KNN	98,17	88,49	84,62	86,31	98,17	88,49	84,62	86,31	4,96E-01
10%	G	94,8	57,32	94,08	71,23	94,87	57,99	98,45	72,77	3,96E-01
	KNN	99,06	95,89	90,21	92,96	99,19	97,18	90,81	93,89	2,41E-05
40%	G	94,84	57,37	93,96	71,24	95,69	61,73	98,11	75,72	3,25E-04
	KNN	99,40	98,17	92,94	95,48	99,57	99,10	94,59	96,79	8,23E-06
70%	G	94,86	57,60	93,69	71,33	96,26	64,96	98,00	78,10	5,55E-07
	KNN	99,45	98,49	93,43	95,89	99,68	99,45	95,83	97,61	8,65E-07
99%	G	94,62	57,48	92,69	70,79	96,44	66,49	97,69	78,89	1,72E-02
	KNN	99,48	97,96	94,88	96,36	99,73	99,04	96,92	97,95	1,41E-01

Tabela 5.2 – Resultados obtidos para o Cenário 1, comparando os dois métodos.

5.3.1.2 Cenário 2: Identificação de modelos de câmeras

Neste cenário de teste os registros foram obtidos da base de dados e rotulados da seguinte forma:

- Dados positivos: são todos os registros de modelo de câmera extraídos dos sites cujo assunto é Câmera.

- Dados negativos: são registros de títulos de livros e autores do site *'abebooks'*; modelos de carro do site *'aol'*; modelos de motor de carro do site *'autobytel'*.

Com um total de 13.258 registros, 5.258 são dados positivos e 8.000 são dados negativos. Conforme mostra a Tabela 5.3, foram feitos testes com diferentes tamanhos do conjunto de treinamento.

		MAVE				FAVE				
Treinamento	**Algoritmo**	**ACC**	**P**	**R**	**F1**	**ACC**	**P**	**R**	**F1**	**T-Test (F1)**
0,1%	G	89,90	90,24	84,95	86,45	91,82	98,37	80,81	88,48	5,28E-01
	KNN	89,90	86,97	88,21	86,85	89,93	87,63	87,15	87,04	9,46E-01
1%	G	93,56	90,78	93,27	91,99	95,92	98,15	91,43	94,62	1,02E-02
	KNN	92,94	87,51	95,93	91,51	93,86	89,11	96,40	92,58	1,30E-02
10%	G	93,61	90,49	93,74	92,08	97,44	97,76	95,73	96,72	1,59E-06
	KNN	97,30	96,04	97,22	96,62	97,70	96,65	97,57	97,11	1,20E-02
40%	G	93,63	90,63	93,69	92,13	97,60	97,73	96,18	96,94	2,26E-08
	KNN	98,21	97,52	97,98	97,75	98,47	97,67	98,49	98,08	1,70E-03
70%	G	93,56	90,57	93,64	92,08	97,76	97,60	96,74	97,17	6,19E-10
	KNN	98,51	97,77	98,54	98,15	98,61	97,83	98,69	98,26	2,75E-01
99%	G	94,14	90,91	95,04	92,89	97,29	96,74	96,34	96,51	1,24E-02
	KNN	98,57	97,74	98,72	98,22	98,35	96,97	98,84	97,87	4,86E-01
99,9%	G	95,00	92,13	96,33	93,92	99,29	98,33	100,00	99,09	9,23E-01
	KNN	97,14	93,57	93,33	93,32	98,57	95,24	100,00	97,23	3,81E-01

Tabela 5.3 – Resultados obtidos para o Cenário de testes número 2, comparando os dois métodos.

5.3.1.3 Cenário 3: Identificação de títulos de livros

Neste cenário de teste os registros foram obtidos da base de dados e rotulados da seguinte forma:

- Dados positivos: são todos os registros de títulos de livros extraídos dos sites cujo assunto é livros.

- Dados negativos: são todos os registros de títulos de filmes extraídos dos sites cujo assunto é filmes; registros de anúncios de emprego extraídos de sites cujo assunto é empregos.

Com um total de 60.000 registros, 20.000 são dados positivos e 40.000 são dados negativos. Conforme mostra a Tabela 5.4, foram feitos testes com diferentes tamanhos do conjunto de treinamento.

		MAVE				FAVE				
Treinamento	**Algoritmo**	**ACC**	**P**	**R**	**F1**	**ACC**	**P**	**R**	**F1**	**T-Test (F1)**
1%	G	63,11	47,24	68,28	54,93	58,65	48,97	71,77	52,10	2,64E-01
	KNN	66,64	50,13	38,04	43,10	66,91	50,67	39,42	44,17	1,29E-01
10%	G	65,36	48,85	65,92	55,85	65,51	52,44	64,56	54,95	5,19E-01
	KNN	69,03	54,48	44,32	48,82	70,20	56,51	46,95	51,24	1,16E-04
40%	G	64,53	47,89	69,63	56,71	66,88	50,74	69,92	58,43	8,90E-04
	KNN	70,65	57,08	49,85	53,10	73,18	62,06	50,61	55,73	1,36E-05
70%	G	64,36	47,56	70,29	56,73	68,15	52,13	66,25	58,08	5,67E-04
	KNN	71,46	58,03	51,57	54,57	73,34	61,71	53,19	57,10	4,71E-06
99%	G	63,75	46,87	69,58	55,98	71,98	58,98	50,62	54,45	2,48E-01
	KNN	66,62	49,91	68,57	57,68	73,28	60,87	54,10	57,23	2,37E-01

Tabela 5.4 – Resultados obtidos para o cenário de testes número 3, comparando os dois métodos.

5.3.1.4 Cenário 4: Identificação de modelos de carros

Este cenário de teste foi montado com base na ideia dos testes que são executados no MAVE. O autor separa os testes por assunto. A informação alvo é rotulada como positivo, e as informações negativas (*outliers*) são as demais informações do mesmo assunto. A seguir será explicado quais são os dados positivos e negativos.

- Dados positivos: são todos os registros de modelos de carros extraídos dos sites cujo assunto é Carros.
- Dados negativos: são todos os registros diferentes de modelos de carros, extraídos dos sites, cujo assunto é Carros, neste caso são: modelo do motor, economia de combustível e preço.

Com um total de 71.692 registros, 17.923 são dados positivos e 53.769 são dados negativos. Conforme mostra a Tabela 5.5, foram feitos testes com diferentes tamanhos do conjunto de treinamento.

5.3.1.5 Cenário 5: Identificação de títulos de livros

Este cenário foi montado com o objetivo de fazer a comparação dos resultados com o MAVE, apesar das características não serem as mesmas utilizadas no MAVE.

- Dados positivos: são todos os registros de títulos de livros extraídos dos sites cujo assunto é livros.
- Dados negativos: são todos os registros diferentes de títulos de livros, extraídos dos sites cujo assunto é livros: autor, código isbn, data de publicação e editora.

		MAVE				FAVE				
Treinamento	**Algoritmo**	**ACC**	**P**	**R**	**F1**	**ACC**	**P**	**R**	**F1**	**T-Test (F1)**
0,1%	G	94,48	89,70	88,69	88,80	88,87	69,87	98,45	81,65	5,88E-04
	KNN	83,05	68,19	64,38	64,59	82,65	66,13	65,60	64,84	9,44E-01
1%	G	95,42	88,45	94,03	91,13	89,97	71,66	99,03	83,15	4,65E-08
	KNN	95,04	86,08	95,69	90,61	96,25	87,91	98,62	92,94	4,77E-05
10%	G	95,69	89,28	94,07	91,61	89,96	71,64	99,03	83,14	1,63E-16
	KNN	98,89	96,79	98,82	97,80	99,53	98,41	99,74	99,07	1,60E-07
40%	G	95,68	89,31	93,99	91,59	89,92	71,57	99,02	83,09	3,13E-14
	KNN	99,58	98,81	99,50	99,15	99,84	99,54	99,84	99,69	1,64E-10
70%	G	95,62	89,27	93,72	91,44	89,88	71,55	98,97	83,05	1,71E-14
	KNN	99,77	99,36	99,72	99,54	99,91	99,75	99,88	99,82	1,53E-06
99%	G	95,73	89,64	94,04	91,77	90,10	71,48	99,15	83,06	3,04E-07
	KNN	99,80	99,46	99,78	99,62	99,93	99,71	100,00	99,85	5,31E-02

Tabela 5.5 – Resultados obtidos para o cenário de testes número 4, comparando os dois métodos.

Com um total de 100.000 registros, 20.000 são dados positivos e 80.000 são dados negativos.

A Tabela 5.6 apresenta os resultados obtidos.

		MAVE				FAVE				
Treinamento	**Algoritmo**	**ACC**	**P**	**R**	**F1**	**ACC**	**P**	**R**	**F1**	**T-Test (F1)**
0,1%	G	66,48	36,21	87,75	51,21	66,86	37,14	87,70	51,39	8,32E-01
	KNN	82,78	62,45	38,59	46,96	81,43	57,25	37,52	43,53	3,47E-01
1%	G	64,87	35,47	92,28	51,23	67,40	37,53	94,42	53,69	6,96E-05
	KNN	86,48	71,40	54,69	61,78	87,16	72,27	58,48	64,55	1,02E-03
10%	G	64,75	35,40	92,61	51,22	68,04	37,92	93,61	53,94	2,37E-06
	KNN	88,73	76,65	63,05	69,13	89,34	78,22	64,69	70,79	3,53E-03
40%	G	64,73	35,46	92,74	51,30	67,66	37,77	94,65	53,99	4,89E-08
	KNN	90,05	80,55	66,57	72,85	90,93	82,98	69,03	75,31	3,90E-03
70%	G	64,78	35,50	92,89	51,37	67,85	37,84	94,60	54,06	5,41E-10
	KNN	90,40	80,99	68,26	74,01	91,06	81,84	71,23	76,13	1,39E-03
99%	G	64,84	35,50	92,26	51,25	68,54	38,47	93,75	54,53	2,78E-03
	KNN	90,41	81,12	68,41	74,06	91,22	83,37	70,80	76,46	1,24E-01

Tabela 5.6 – Resultados obtidos para o cenário de testes número 5, comparando os dois métodos.

5.3.2 Algoritmos de Classificação de Uma Classe

Essa subseção tem o objetivo de apresentar os resultados obtidos utilizando algoritmos de classificação de Uma Classe. Esses algoritmos utilizam o método não supervisionado para o aprendizado e criação do modelo de predição.

Os algoritmos selecionados para esses testes foram: OneClassSVM (OneSVM) (SCHÖLKOPF et al., 2001) e IsolationForest (IF) (LIU; TING; ZHOU, 2008).

5.3.2.1 Cenário 1: Identificação de Modelos de Câmeras

Neste cenário, os dados utilizados para rodar o algoritmo são os mesmos dados do Cenário 1 da Seção 5.3.1. A diferença é que utilizando algoritmos de Classificação de uma Classe, os dados não serão rotulados com *labels* 'positivo' e 'negativo'. O conjunto de dados para treinamento, no entanto, serão os mesmos registros, porém, apenas positivos. O conjunto de dados para teste será todos os registros do Cenário da seção 5.3.1, os positivos e negativos. A principal diferença é que apenas dados positivos serão usados para o treinamento. Na Tabela 5.7 a coluna 'Treinamento' é em relação ao conjunto de dados de treinamento (5.258 registros).

- Quantidade total de registros para treinamento: 5.258
- Quantidade total de registros para teste: 76.950

		MAVE				FAVE				
Treinamento	**Algoritmo**	**ACC**	**P**	**R**	**F1**	**ACC**	**P**	**R**	**F1**	**T-Test (F1)**
10%	OneSVM	93,66	94,46	99,05	96,70	94,10	94,69	99,27	96,93	9,40E-21
	IF	90,33	96,87	92,69	94,73	88,07	96,99	90,07	93,40	3,60E-05
40%	OneSVM	95,34	96,88	98,31	97,59	96,39	97,05	99,26	98,14	1,09E-24
	IF	87,53	98,90	87,96	93,10	91,34	98,78	92,10	95,31	5,40E-04
70%	OneSVM	96,50	98,25	98,17	98,21	97,66	98,41	99,21	98,81	2,07E-20
	IF	87,19	99,42	87,41	93,02	91,30	99,63	91,45	95,35	1,19E-03
95%	OneSVM	98,04	99,76	98,27	99,01	99,14	99,75	99,39	99,57	2,70E-24
	IF	85,20	99,99	85,15	91,96	92,37	99,98	92,36	96,01	1,39E-04

Tabela 5.7 – Resultados obtidos para o cenário de testes número 1, utilizando Classificadores de Uma Classe, comparando-se os dois métodos.

5.3.2.2 Cenário 2: Identificação de Modelos de Câmeras

Partindo da mesma ideia do cenário anterior, este cenário irá utilizar os mesmos dados do Cenário 2 da Seção 5.3.1. Os dados serão tratados da mesma forma como foram preparados no Cenário 1 desta seção, já que os algoritmos em questão nesta etapa dos testes são os algoritmos de Classificação de Uma Classe. Neste cenário os testes foram executados de uma forma ligeiramente diferente dos demais casos. Os registros positivos do conjunto de treinamento que não foram utilizados no processo de treinamento foram incluídos no conjunto de teste. A Tabela 5.8 apresenta os resultados.

- Quantidade total de registros para treinamento: 5.258
- Quantidade total de registros para teste: 13.258

		MAVE				FAVE				
Treinamento	**Algoritmo**	**ACC**	**P**	**R**	**F1**	**ACC**	**P**	**R**	**F1**	**T-Test (F1)**
10%	OneSVM	67,22	65,74	99,89	79,29	68,66	66,73	99,96	80,03	2,06E-18
	IF	80,84	78,01	96,79	86,38	83,04	79,78	97,83	87,88	3,43E-03
40%	OneSVM	79,46	77,86	99,73	87,44	80,52	78,68	99,91	88,04	9,16E-17
	IF	86,29	91,31	89,38	90,30	91,06	90,22	98,18	94,03	5,72E-04
70%	OneSVM	86,65	86,44	99,65	92,57	87,89	87,41	99,89	93,24	2,21E-19
	IF	87,36	95,17	89,40	92,16	93,83	96,53	96,07	96,29	1,85E-03
95%	OneSVM	97,57	97,89	99,64	98,76	97,73	97,80	99,90	98,84	8,66E-07
	IF	84,36	99,95	83,89	91,16	95,83	99,83	95,85	97,79	1,26E-04

Tabela 5.8 – Resultados obtidos para o cenário de testes número 2, utilizando Classificadores de uma Classe, comparando os dois métodos.

5.3.2.3 Cenário 3: Identificação de Títulos de Livros

Partindo da mesma ideia dos cenários anteriores, este cenário irá utilizar os mesmos dados do Cenário 3 da Seção 3.5.1. Os dados serão tratados da mesma forma como foram preparados no cenário anterior, já que os algoritmos em questão, nesta etapa dos testes, são os algoritmos de Classificação de Uma Classe. A Tabela 5.9 apresenta os resultados.

- Quantidade total de registros para treinamento: 20.000
- Quantidade total de registros para teste: 60.000

		MAVE				FAVE				
Treinamento	**Algoritmo**	**ACC**	**P**	**R**	**F1**	**ACC**	**P**	**R**	**F1**	**T-Test (F1)**
10%	OneSVM	58,78	74,19	61,70	67,36	57,89	74,96	58,47	65,69	2,44E-03
	IF	44,22	85,66	22,97	36,20	42,90	87,41	20,11	32,66	8,40E-03
40%	OneSVM	57,08	79,42	59,68	68,14	55,50	79,51	56,79	66,24	6,63E-04
	IF	37,02	90,54	20,24	33,08	35,77	90,73	18,37	30,53	9,47E-03
70%	OneSVM	59,92	88,52	61,94	72,88	56,55	88,39	57,59	69,74	3,71E-06
	IF	30,84	95,19	21,55	35,13	31,06	95,00	21,87	35,54	4,26E-01
95%	OneSVM	60,75	98,18	60,90	75,15	57,76	98,29	57,71	72,71	1,52E-03
	IF	23,40	99,32	21,63	35,52	24,04	99,54	22,25	36,35	3,18E-01

Tabela 5.9 – Resultados obtidos para o cenário de testes número 3, utilizando Classificadores de Uma Classe, comparando os dois métodos.

5.3.2.4 Cenário 4: Identificação de modelos de carros

Partindo da mesma ideia dos cenários anteriores, este cenário irá utilizar os mesmos dados do Cenário 4 da Seção 3.5.1. Os dados serão tratados da mesma forma como foram preparados no cenário anterior, já que os algoritmos em questão, nesta etapa dos testes, são os algoritmos de Classificação de Uma Classe. A Tabela 5.10 apresenta os resultados.

- Quantidade total de registros para treinamento: 17.923
- Quantidade total de registros para teste: 71.692

		MAVE				FAVE				
Treinamento	**Algoritmo**	**ACC**	**P**	**R**	**F1**	**ACC**	**P**	**R**	**F1**	**T-Test (F1)**
1%	OneSVM	76,72	76,38	99,95	86,59	77,13	76,68	100,00	86,80	1,51E-03
	IF	74,15	78,67	90,01	83,92	62,67	75,44	73,80	74,22	9,33E-03
10%	OneSVM	82,03	81,15	99,82	89,52	82,32	81,32	100,00	89,70	5,95E-02
	IF	82,20	86,15	91,57	88,77	81,81	86,21	90,90	88,48	5,52E-01
40%	OneSVM	88,82	88,44	99,59	93,69	89,46	88,97	99,72	94,04	1,16E-04
	IF	92,16	94,52	96,15	95,29	94,52	95,26	98,31	96,75	8,50E-02
70%	OneSVM	92,33	92,51	99,63	95,94	92,98	92,90	99,92	96,28	2,79E-07
	IF	93,91	95,28	98,16	96,69	94,02	95,57	97,97	96,75	9,21E-01
95%	OneSVM	98,39	98,79	99,58	99,18	98,69	98,79	99,89	99,34	1,33E-18
	IF	94,66	99,38	95,16	97,21	97,03	99,43	97,54	98,46	8,34E-02

Tabela 5.10 – Resultados obtidos para o cenário de testes número 4, utilizando Classificadores de Uma Classe, comparando os dois métodos.

5.4 Discussão sobre os resultados

O objetivo desta seção é apresentar um resumo dos resultados obtidos após experimentos feitos em 5 cenários diferentes. Conforme já mencionado nas seções anteriores, duas estratégias diferentes foram escolhidas para executar os testes, uma delas é utilizar algoritmos de classificação de duas ou mais classes e em que o aprendizado é supervisionado, e outra estratégia foi utilizar algoritmos de classificação baseados em uma classe em que o aprendizado é não-supervisionado.

Os algoritmos de classificação de uma classe são: OneClassSVM (OneSVM) e Isolation Forest (IF). Os algoritmos de classificação de duas ou mais classes são: K-Nearest Neighbour (KNN) e Normal Gaussian (G).

Conforme pode ser observado, nos algoritmos de classificação de duas ou mais classes (KNN e G), o algoritmo que se saiu melhor na maioria dos experimentos foi o KNN. O que pode ser observado é que o KNN obtém resultados ainda melhores que o G nos casos de teste onde a base de treinamento é menor que 70%. Quanto menor a base de treinamento mais o KNN se sobressai em relação ao G. Ao ponto que o conjunto de treinamento aumenta, ambos algoritmos alcançam resultados parecidos.

Em relação aos algoritmos de classificação de uma classe (OneSVM e IF), os resultados variam conforme a relação entre dados positivos e negativos. Essa análise pode ajudar na escolha do algoritmo em cada caso.

O que pode ser notado é que nos cenários em que a base de treinamento é incluída junto na base de teste, o algoritmo IF obtém melhores resultados. Isso pode ser observado no Cenário 2 da Seção 5.3.2. Ou seja, o IF consegue identificar com mais precisão os exemplos positivos que foram usados para criar o modelo de predição, enquanto que o OneSVM não atinge bons resultados com esses exemplos. Mas no contexto geral o OneSVM é superior ao IF ao que se trata de identificar *outliers*.

Analisando os resultados dos algoritmos de classificação de uma classe (OneSVM e IF) e os algoritmos de classificação de duas classes (KNN e G), podemos concluir que o KNN obtém melhores resultados contra os algoritmos de classificação de uma classe. Apesar de o IF e OneSVM não apresentarem os melhores resultados se comparados com o KNN e G, eles tiveram resultados mais satisfatórios se comparados quando utilizado a estratégia do MAVE, conforme pode ser observado no Cenário 1 da Seção 5.3.2.

Um caso em que nenhum algoritmo obteve bons resultados é no Cenário 3. Esse cenário apresenta um dos piores casos em classificação de dados, pois os conjuntos de dados positivos e negativos apresentam menor discrepância entre eles. Apesar disso, o algoritmo que obteve melhores resultados nesse caso é o KNN.

Em alguns cenários, o tamanho do treinamento, em alguns casos, foi reduzido para até 0,1% e em outros casos, foi aumentado até 99%. Tal decisão foi motivada pelo fato de que com os valores padrões (10%, 40% e 70%) os resultados não foram significativamente diferentes. Então, optou-se por elevar ao extremo os testes para analisar o impacto disto no resultado. E, como pode ser observado, mesmo utilizando o treinamento com tamanho 0,1%, alguns casos tiveram bons resultados.

6 CONCLUSÃO

Esse trabalho de conclusão de curso teve como objetivo analisar técnicas de verificação de *wrappers* e propor um novo método, com base nas técnicas estudadas durante o desenvolvimento do trabalho.

Foi utilizado como fundamentação o método MAVE. Outros métodos de verificação de dados foram estudados durante o desenvolvimento do trabalho, porém, o MAVE apresentou os melhores resultados. Com base nessa ferramenta, foi proposto um novo método de verificação de dados, com o objetivo de melhorar os resultados já alcançados pelo método estudado. A nova proposta, intitulada como FAVE, tem como principal diferença a ideia de manter todas as características (numéricas e não-numéricas) para fazer o treinamento dos algoritmos de classificação.

Observando os experimentos, pode-se dizer que a proposta do FAVE obteve resultados promissores na maioria dos cenários em que foi experimentado. Comparado com o método no estado da arte, obteve um ganho superficial utilizando o mesmo cenário de teste.

Como trabalho futuro, pode-se testar mais casos com diferentes bases de dados, combinando com diferentes valores nos parâmetros de treinamento dos algoritmos. Além disso, testes estatísticos devem ser feitos para validar o ganho real da abordagem proposta.

REFERÊNCIAS

BRONZI, M. et al. Extraction and integration of partially overlapping web sources. **Proceedings of the VLDB Endowment**, [S.l.], v.6, n.10, p.805–816, 2013.

CHIDLOVSKII, B.; ROUSTANT, B.; BRETTE, M. Documentum eci self-repairing wrappers: performance analysis. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2006. **Proceedings...** [S.l.: s.n.], 2006. p.708–717.

COHEN, J. P.; DING, W.; BAGHERJEIRAN, A. Semi-Supervised Web Wrapper Repair via Recursive Tree Matching. **arXiv preprint arXiv:1505.01303**, [S.l.], 2015.

CRESCENZI, V. et al. Roadrunner: towards automatic data extraction from large web sites. In: VLDB. **Anais...** [S.l.: s.n.], 2001. v.1, p.109–118.

CRESCENZI, V.; MECCA, G. Automatic information extraction from large websites. **Journal of the ACM (JACM)**, [S.l.], v.51, n.5, p.731–779, 2004.

FERRARA, E.; BAUMGARTNER, R. Automatic wrapper adaptation by tree edit distance matching. In: **Combinations of Intelligent Methods and Applications**. [S.l.]: Springer, 2011. p.41–54.

FERRARA, E. et al. Web data extraction, applications and techniques: a survey. **Knowledge-Based Systems**, [S.l.], v.70, p.301–323, 2014.

HAO, Q. **A dataset for structured data extraction from web pages**. [Online; acessado em 20/06/2018], https://archive.codeplex.com/?p=swde.

HAO, Q. et al. From one tree to a forest: a unified solution for structured web data extraction. In: ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 34. **Proceedings...** [S.l.: s.n.], 2011. p.775–784.

KUSHMERICK, N. Wrapper verification. **World Wide Web**, [S.l.], v.3, n.2, p.79–94, 2000.

KUSHMERICK, N. Finite-state approaches to web information extraction. In: **Information Extraction in the Web Era**. [S.l.]: Springer, 2003. p.77–91.

LERMAN, K.; MINTON, S. N.; KNOBLOCK, C. A. Wrapper maintenance: a machine learning approach. **Journal of artificial intelligence research**, [S.l.], v.18, p.149–181, 2003.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation forest. In: DATA MINING, 2008. ICDM'08. EIGHTH IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2008. p.413–422.

MANNING, C. D. et al. **Introduction to information retrieval**. [S.l.]: Cambridge university press Cambridge, 2008. v.1, n.1.

MENG, X. et al. SG-WRAP: a schema-guided wrapper generator. In: DATA ENGINEERING, 2002. PROCEEDINGS. 18TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2002. p.331–332.

SCHÖLKOPF, B. et al. Estimating the support of a high-dimensional distribution. **Neural computation**, [S.l.], v.13, n.7, p.1443–1471, 2001.

SELKOW, S. M. The tree-to-tree editing problem. **Information processing letters**, [S.l.], v.6, n.6, p.184–186, 1977.

VIANA, I. F. de et al. Toward one class classifier techniques applied to verifier information. In: INFORMATION SYSTEMS AND TECHNOLOGIES (CISTI), 2011 6TH IBERIAN CONFERENCE ON. **Anais...** [S.l.: s.n.], 2011. p.1–7.

VIANA, I. F. de et al. MAVE: multilevel wrapper verification system. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.28, n.9, p.2393–2406, 2016.