UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marcelo Canzian Nunes

Representação de Dados Enriquecida para Extração na Web

Florianópolis
2024

Marcelo Canzian Nunes

Representação de Dados Enriquecida para Extração na Web

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Ciência da Computação.
Orientadora: Profa. Carina Friedrich Dorneles, Dra.

Florianópolis
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Nunes, Marcelo Canzian
Representação de Dados Enriquecida para Extração na Web / Marcelo Canzian Nunes ; orientadora, Carina Friedrich Dorneles, 2024.
71 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Informação, Florianópolis, 2024.

Inclui referências.

1. Ciência da Informação. 2. Extração de Dados. 3. Extração de Dados Semiestruturados da Web. 4. Modelo de Linguagem Bidirecional. I. Dorneles, Carina Friedrich. II. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Ciência da Informação. III. Título.

Marcelo Canzian Nunes

Representação de Dados Enriquecida para Extração na Web

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Renato Fileto, Dr.
Universidade Federal de Santa Catarina - UFSC

Profa. Carmem Satie Hara, Dra.
Universidade Federal do Paraná - UFPR

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Ciência da Computação.

Documento assinado digitalmente
Marcio Bastos Castro
Data: 19/08/2024 18:43:01-0300
CPF: ***.707.720-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de
Pós-Graduação

Documento assinado digitalmente
CARINA FRIEDRICH DORNELES
Data: 19/08/2024 14:02:31-0300
CPF: ***.834.250-**
Verifique as assinaturas em https://v.ufsc.br

Profa. Carina Friedrich Dorneles, Dra.
Orientadora

Florianópolis, 2024.

Dedico este trabalho à minha família.

AGRADECIMENTOS

Gostaria de agradecer, primeiramente, à minha família por todo apoio, incentivo e compreensão durante toda a trajetória do curso. Sem eles não estaria aqui.

Gostaria de agradecer à minha orientadora, Profa. Dra. Carina Friedrich Dorneles, pela parceria, confiança e, principalmente, pela paciência e transmissão de seus conhecimentos, sem os quais este trabalho não seria possível. Estendo meus agradecimentos a todos os professores que fizeram parte de minha educação.

Agradeço também à Universidade Federal de Santa Catarina e ao Programa de Pós-Graduação em Ciência da Computação pela oportunidade de evoluir em mais uma etapa de minha jornada acadêmica.

Gostaria de fazer um agradecimento especial a minha companheira, Gabrieli Roldão da Silva, por todo o suporte durante este projeto, por compreender as horas de dedicação, por me incentivar nos momentos difíceis e compartilhar as alegrias e realizações ao longo deste trabalho.

Por fim, gostaria de agradecer a todos que de maneira direta e indireta contribuíram para a realização deste trabalho.

“A educação tem raízes amargas, mas os seus frutos são doces.”
Aristóteles

RESUMO

A extração de informações de páginas da Web é uma importante tarefa que visa aquisição de dados e informações relevantes de uma vasta gama de fontes, permitindo sua utilização para tomadas de decisões estratégicas, análises estatísticas, previsões de tendências, entre outros. Por definição, as páginas são desenvolvidas visando sua utilização por seres humanos através de navegadores, que realizam a interpretação e exibição do conteúdo em código HTML, CSS e JavaScript através do processo de renderização. Porém, identificar e extrair estas informações apenas por seu código HTML, sem a utilização de renderização, ainda é um grande desafio já que a estrutura das páginas utilizam um modelo semiestruturado, não padronizado e com variações entre diferentes fontes, dificultando o processo de extração. Além disso, as páginas frequentemente possuem conteúdo dinâmico, que se reflete em alterações de sua estrutura, *layout* e marcações HTML, o que torna o processo de extração ainda mais complexo. Para superar isso, este trabalho propõe uma abordagem que utiliza as informações da árvore DOM em conjunto com as informações visuais extraídas em forma de metadados dos elementos HTML da página, para classificar e extrair seus conteúdos relevantes, chamada de *Enhanced Data Representation for Extraction in Web* (EDREW). Para isso, é criado um modelo textual que representa a identidade visual dos elementos de uma página, a fim de emular o seu contexto visual e hierarquia, sem a necessidade de renderização por um navegador. Com este modelo textual, é realizada a classificação dos elementos entre ruído e conteúdo relevante, utilizando o modelo de linguagem bidirecional ELMo para a contextualização e identificação das características individuais de cada tipo de elemento, realizando assim a extração dos conteúdos relevantes. A avaliação do EDREW foi realizada por meio de uma análise experimental utilizando os *datasets* SWDE e N-SWDE, para os quais este trabalho alcançou uma média de extração de dados relevantes utilizando a métrica *F1-score* de 97% no SWDE e 76% no N-SWDE utilizando apenas o domínio de páginas de automóveis.

Palavras-chave: extração semiestruturada web, extração dados web, modelo de linguagem, modelo de linguagem bidirecional, ELMo, documentos HTML, renderização.

ABSTRACT

The extraction of information from Web pages is an important task that aims to acquire relevant data and information from a wide range of sources, allowing its use for strategic decision making, statistical analysis, trend analysis, among others. By definition, pages are developed to be used by human beings through browsers, which perform the interpretation and display of content in HTML, CSS and JavaScript code through the rendering process. However, identifying and extracting information solely through its HTML code, without the use of rendering, is still a major challenge as the structure of the pages uses a semi-structured, non-standardized model with variations between different sources, making the extraction process difficult. Furthermore, pages often have dynamic content, which is reflected in changes to their structure, layout and HTML markup, which makes the extraction process even more complex. To overcome this, this work proposes an approach that uses information from the DOM tree in conjunction with visual information extracted in the form of metadata from the page's HTML elements, to classify and extract their relevant content, called Enhanced Data Representation for Extraction in Web (EDREW). To do this, a textual model is created that represents the visual identity of the elements on a page, in order to emulate their visual context and classification, without the need for rendering by a browser. With this textual model, the elements are classified between noise and relevant content, using the ELMo bidirectional language model for contextualization and identification of the individual characteristics of each type of element, thus extracting the relevant content. The evaluation of EDREW was carried out through an experimental analysis using the SWDE and N-SWDE datasets, where this work achieved an average extraction of relevant data, using the F1-score metric, of 97% in the complete SWDE and 76% in the N-SWDE using only automobile pages domain.

Keywords: semi-structured web extraction, web data extraction, language model, bidirectional language model, ELMo, HTML documents, rendering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery*
AEDI	*Align and Extract Data Items*
AOI	*Area of Interest*
BiLM	*Bidirectional Language Model*
CNN	*Convolutional Neural Networks*
CSS	*Cascading Style Sheets*
CTVS	*Combining Tag and Value Similarity*
DBSCAN	*Density Based Spatial Clustering of Applications with Noise*
DeLa	*Data Extraction and Label Assignmen*
DEPTA	*Data Extraction based Partial Tree Alignment*
DL	*Deep Learning*
DOM	*Document Object Model*
ELMo	*Embeddings from Language Models*
FFT	*Fast Fourier Transform*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
ISBN13	*International Standard Book Number - 13*
LANTERN	*Learning Transferable Node Representations for Attribute Extraction*
LLM	*Large Language Model*
LSTM	*Long Short-Term Memory*
MDR	*Mining Data Record*
ML	*Machine Learning*
MRC	*Machine Reading Comprehension*
MSA	*Multiple Sequence Alignment*
MVC	*Model-View-Controller*
NLP	*Natural Language Processing*
RED	*Redundancy-Driven Data Extraction*
REST	*Representational State Transfer*
SC	Similaridade de Cosseno
SPA	*Single Page Application*
SWDE	*Structured Web Data Extraction Dataset*
TF-IDF	*Term Frequency - Inverse Document Frequency*
TP	*Tag Path*
TPS	*Tag Path Sequence*
VBM	*Visual Block Model*
ViDIE	*Vision-Based Data Item Wrapper*

ViDRE	*Vision-Based Data Record Wrappe*
ViPER	*Visual Perception-based Extraction of Records*
VIPS	*Vision-based Page Segmentation*
W3C	*World Wide Web Consortium*
WT	*Wrapper Tag*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A Web é vista como o maior banco de dados a que se tem acesso, podendo-se obter informações de diversas fontes que abrangem uma variedade de conteúdos, possibilitando assim a criação de *datasets* para tomadas de decisões estratégicas, análises estatísticas, análises de tendências, entre outras finalidades (TANI; FARID; RAHMAN, 2012; TSENG, 2014). Por esse motivo, a tarefa de extrair dados relevantes das páginas Web é essencial, ainda que desafiadora, já que torná-las manipuláveis ainda é um obstáculo (FAYZRAKHMANOV *et al.*, 2018). Os dados são disponibilizados nos mais diversos formatos e estruturas, dificultando a automatização da extração para diferentes páginas. Além disso, as páginas Web são projetadas visando a usabilidade feita por humanos e não por máquinas, podendo apresentar estruturas distintas para os mesmos conteúdos (PARK; NGUYEN; WON, 2015; CRESTAN; PANTEL, 2011; ANDERSON; HONG, 2013a).

Como a maioria das páginas são escritas utilizando código HTML, as soluções existentes são baseadas na análise de seu DOM HTML. No entanto, com o avanço do próprio HTML e das tecnologias utilizadas para o desenvolvimento Web, novas versões de *tags* são introduzidas, necessitando que os trabalhos anteriores necessitem de alterações para serem compatíveis com as novas atualizações (LIU, W.; MENG, X.; MENG, W., 2010a). Para facilitar a utilização do usuário, suas estruturas são construídas utilizando um mesmo padrão de *layout* para os mesmos itens e utilizando de artifícios como estilos e classes para padronizar a interface do usuário. Além disso, junto com as informações relevantes, há muita informação que não traz valor ao conteúdo, poluindo a estrutura do DOM HTML e fazendo com que sejam obtidas informações irrelevantes durante a extração dos dados, como por exemplo anúncios, menus, etc. (LIU, W.; MENG, X.; MENG, W., 2010a; ANDERSON; HONG, 2013a). A Figura 1 apresenta um exemplo de ruído, onde na comparação entre as duas páginas, é acrescentado um bloco de informações adicionais sobre o automóvel na figura à direita, entre a imagem do veículo e o texto com sua descrição, elemento que não existe na figura à esquerda. Este bloco de informações acrescentado acaba dificultando a tarefa de extração de dados, pois isso altera a estrutura do DOM HTML e atrapalha na busca por padrões.

Várias abordagens já foram utilizadas para tentar sanar esse problema, como a extração via texto bruto (DOWNEY *et al.*, 2004; WENINGER; HSU, 2008), a extração via árvore DOM (MEHTA; NARVEKAR, 2015), a extração por meio de informações visuais adquiridas do *layout* da página (ANDERSON; HONG, 2013b) e através da remoção de ruído das páginas (NARWAL, 2013; VELLOSO; DORNELES, Carina F, 2020). Porém, com exceção da extração por meio de informações visuais, todas as abordagens citadas acima enfrentam dificuldades em extrair dados de páginas Web que são desenvolvidas utilizando o conceito de *Single Page Application* (SPA). Uma

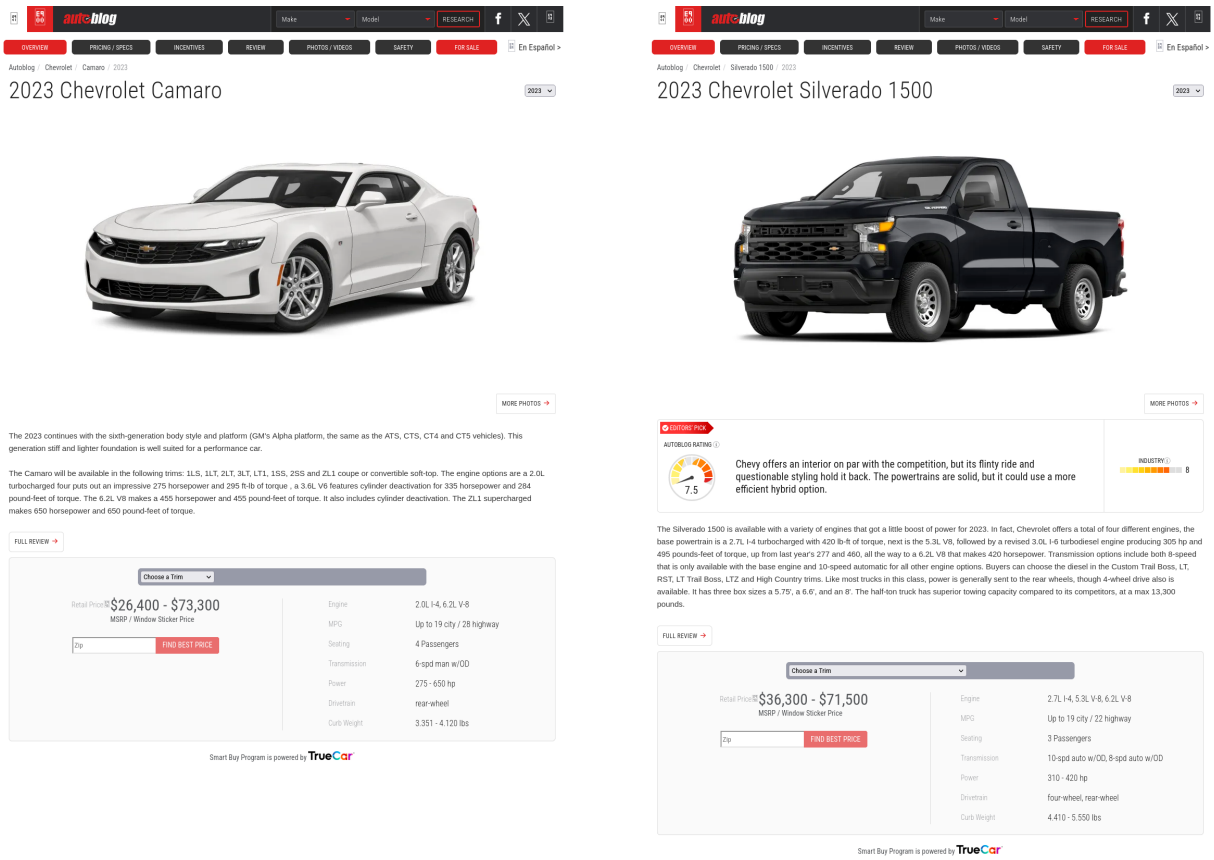

Figura 1 – Apresentação de uma página sem ruído à esquerda e uma página com ruído à direita, que possui um bloco adicional sobre o automóvel, estando localizado entre sua imagem e descrição textual.

SPA concentra suas funcionalidades em uma única página, ao invés de requisitar ao servidor uma nova página a cada operação, e realiza a interação com o usuário através de alterações de seus elementos HTML e sua estrutura, apresentando seu conteúdo de forma dinâmica (HOPPE; SOUZA, 2023). Apesar da abordagem de extração utilizando informações visuais do *layout* da página conseguir ultrapassar esta barreira, o sucesso da proposta depende da renderização da página por um navegador, o que torna o método muito custoso para um grande volume de dados (SIMON; LAUSEN, 2005; FAYZRAKHMANOV *et al.*, 2018).

Através do levantamento dos trabalhos relacionados, foi identificado que o problema de extração de dados, principalmente em páginas que utilizam tecnologias SPA, ainda possui pontos a serem melhorados. A coleta desses dados se torna muito relevante para sistemas que dependem de dados consolidados e informações enriquecidas para realizar análises e tomadas de decisão. Além disso, distinguir conteúdo relevante de ruídos da página auxilia a diminuir a quantidade de dados armazenados e processados, reduzindo assim o custo total da operação em valores computacionais e monetários.

Diante do problema de pesquisa, o objetivo principal deste trabalho é desenvol-

ver uma nova abordagem para extração de dados em ambientes semiestruturados e dinâmicos, sem a utilização de renderização das páginas Web. Para atingir o objetivo principal, foram estabelecidos os seguintes objetivos específicos:

- Coletar páginas da Web de diferentes fontes, de diferentes assuntos e de diferentes estruturas;
- Analisar e selecionar as melhores abordagens para extração de informações visuais do *layout* da página através do texto HTML das páginas;
- Implementar extração de informações visuais;
- Criar um modelo baseado nas características identificadas;
- Realizar um estudo de caso para avaliação da abordagem proposta.

Para avaliar a tarefa de extração dos dados pelo trabalho proposto, são executadas comparações de extração dos dados contra os trabalhos WebKE (XIE *et al.*, 2021) e LANTERN (ZHOU *et al.*, 2022), utilizando o *dataset* público *Structured Web Data Extraction Dataset* (SWDE), que é composto por 124.291 páginas anotadas distribuídas em 8 verticais, onde cada vertical representa um domínio específico dos dados baseado em assunto ou setor. Além disso, também são executadas comparações com o trabalho LANTERN (ZHOU *et al.*, 2022), porém utilizando um novo *dataset* criado para este trabalho, chamado de *New Structured Web Data Extraction Dataset* (N-SWDE). Este novo *dataset* contém as mesmas páginas do SWDE original, mas com as versões atuais das páginas Web, onde são empregadas novas tecnologias que não existiam na época de criação do SWDE original.

Durante a discussão dos resultados, são apresentadas e discutidas as situações em que o trabalho obteve e não obteve sucesso (total ou parcial). São apontadas as causas e sugeridas maneiras de se atacar as limitações identificadas durante a realização dos experimentos. O resultado obtido possui uma média de 97% de extração dos dados relevantes das páginas Web no *dataset* SWDE e 76% no *N-SWDE* utilizando apenas o domínio de páginas de automóveis. Ambos resultados foram avaliados utilizando como métrica de comparação o *F1-score*.

Esta dissertação está organizada da seguinte forma: no Capítulo 2 é descrita a fundamentação teórica do trabalho, para orientar o leitor no entendimento dos assuntos dos capítulos posteriores; no Capítulo 3 é apresentado a revisão da literatura sobre a tarefa de extração de dados de páginas Web, descrevendo os trabalhos relacionados e suas limitações; no Capítulo 4 é descrita a proposta deste trabalho, apresentando suas definições e algoritmos utilizados; no Capítulo 5 são apresentados os resultados obtidos e suas limitações, através de experimentos realizados comparando o trabalho proposto com o LANTERN e WebKE; no Capítulo 5 são apresentadas as conclusões e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 LAYOUTS E COMPOSIÇÕES DE UMA PÁGINA WEB

Um dos objetivos dos *layouts* de uma página é fazer com que os usuários levem o menor tempo possível para entender a disposição dos elementos e buscar as informações que desejam, servindo como um canal entre o usuário e a informação (BEAIRD; WALKER; GEORGE, 2020). Seu bloco de navegação principal deve ser facilmente visível, com *links* descritivos. Recursos como navegação secundária e campos de pesquisa não devem dominar a página, mas sim ser facilmente encontrados e visualmente separados do conteúdo principal. Isso permite que os usuários se concentrem nas informações enquanto sabem onde encontrar outros recursos quando necessário. Visitantes tendem a sair rapidamente de um site se não encontrarem o que procuram, portanto, manter o conteúdo principal como ponto focal é crucial para a usabilidade (BEAIRD; WALKER; GEORGE, 2020). A Figura 2 representa a anatomia de uma página Web, com seus principais blocos destacados. Uma página normalmente é construída dentro de uma estrutura de *container*, que pode ser dividida em três grandes regiões: superior, central e inferior. Na sua região superior, são localizados o logo da página, podendo existir uma estrutura de menus de navegação (*navigation*); na região central, são localizados os conteúdos da página (*content*), podendo ou não ter uma estrutura de navegação; na região inferior, é localizado seu rodapé (*footer*), contendo informações auxiliares da página.

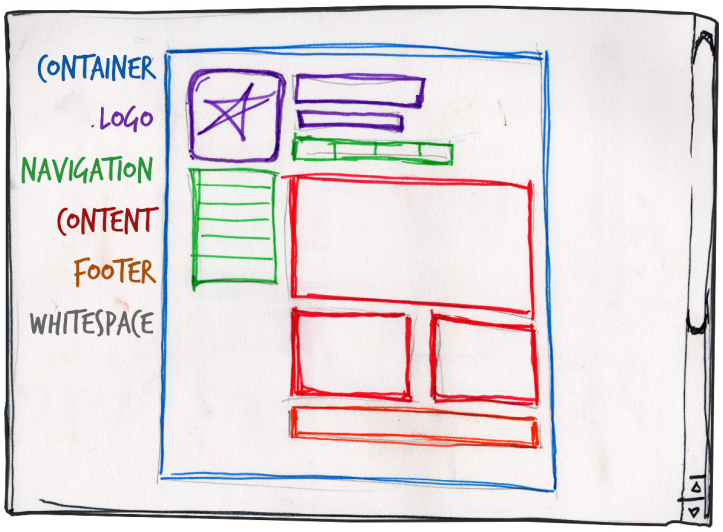

Figura 2 – Anatomia de uma página Web, com seus principais blocos destacados (BEAIRD; WALKER; GEORGE, 2020).

Para o alinhamento e disposição dos blocos e elementos de uma interface, a utilização de grades é uma técnica bem difundida (MOHAMED *et al.*, 2014; SIU; CHAPARRO, 2014; NAMOUN, 2018), onde seu uso vai além do alinhamento dos elementos,

envolvendo também proporção. A teoria do uso de grades remonta a Piet Mondrian e aos pitagóricos, que estabeleceram a proporção áurea como um padrão matemático na natureza (ILIĆ; STEFANOVIĆ; SADIKOVIĆ, 2018; MARPLES; WILLIAMS, 2022). A proporção áurea é uma base fundamental para a organização estética dos elementos em uma página, onde ela é representada pelo número 1,61803. Este número pode ser obtido através da razão entre duas partes de um objeto, onde um objeto é igual à razão entre a soma das duas partes maiores e a parte maior. Outra forma de se obter a proporção áurea é através da utilização da série de Fibonacci, onde ao dividir um número da série por seu anterior o resultado será cada vez mais próximo de 1,618 (BEAIRD; WALKER; GEORGE, 2020; PROKOPAKIS *et al.*, 2013).

Composições divididas por linhas proporcionais à proporção áurea são usadas como uma forma de dividir a tela e posicionar os pontos focais, além de serem consideradas esteticamente agradáveis para os usuários, fazendo com que os *layouts* das páginas tendam a adotar esta composição para seus blocos e elementos. Também é possível apresentar as informações através da regra dos terços, onde duas ou mais linhas verticais e horizontais imaginárias são traçadas sobre o *layout* da página, formando uma grade (BEAIRD; WALKER; GEORGE, 2020). Com esta grade pode-se organizar os blocos de conteúdo e seu posicionamento na página de forma harmônica, conforme apresentado na Figura 3.

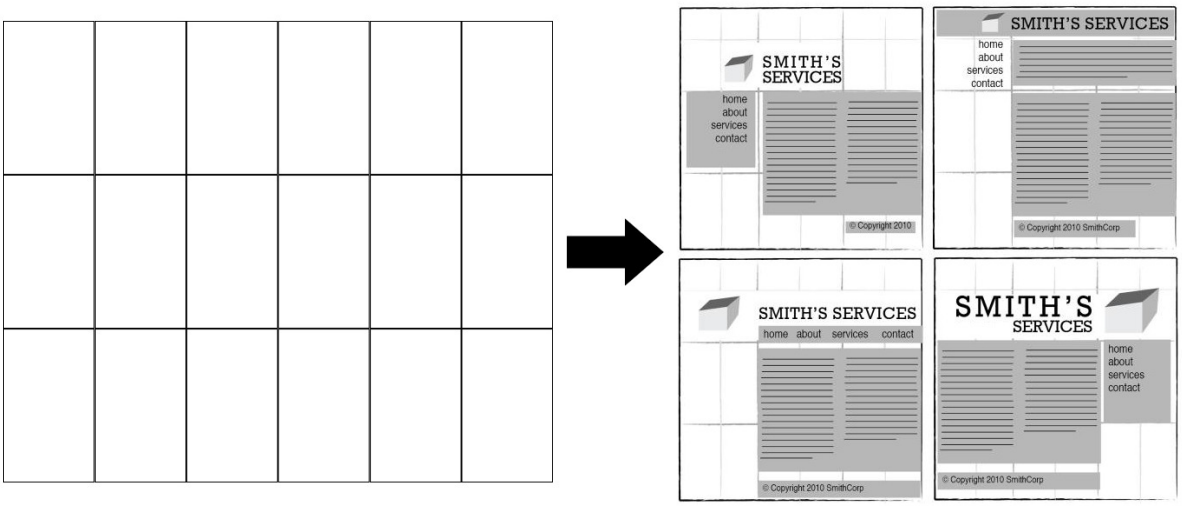

Figura 3 – Exemplo do uso da regra dos terços para montar o *layout* de uma página Web (BEAIRD; WALKER; GEORGE, 2020).

Um *layout* unificado é aquele que funciona como um todo de forma harmônica e a proximidade entre os elementos de um mesmo tipo é uma estratégia poderosa para criar essa unidade. Isto cria pontos focais na página e destaca os grupos de elementos, onde elementos semelhantes podem possuir regras de estilo *Cascading Style Sheets* (CSS) semelhantes, espaçamentos semelhantes, entre outros, formando assim uma unidade na página Web (BEAIRD; WALKER; GEORGE, 2020).

Durante a evolução das páginas HTML, o modo de utilização dos elementos também foram modificando. No início dos anos 2000, o uso de tabelas HTML era a técnica predominante para criar *layouts* de páginas, onde eram utilizadas para alinhar elementos e criar estruturas complexas. Com o desenvolvimento e adoção do CSS, a *World Wide Web Consortium* (W3C) criou diretrizes para incentivar a separação do conteúdo (HTML) da apresentação (CSS) através de um movimento chamado de *tabless*, permitindo assim uma maior flexibilidade no *design* e na manutenção do código (W3C, 2002, 2021).

2.2 SINGLE PAGE APPLICATION

A construção de aplicações Web que centralizam suas funcionalidades em uma única página é chamada de *Single Page Application* (SPA). Diferentemente de uma aplicação baseada no padrão *Model-View-Controller* (MVC), a lógica de montagem dos elementos da interface Web em uma aplicação SPA é transferida do servidor para o cliente, através do *download* de todos seus códigos necessários para executar a página Web em sua primeira chamada. A partir disso, não há a necessidade de realizar novas requisições para o servidor sempre que houver mudança nas páginas do cliente, fazendo com que o servidor se preocupe apenas com a troca de informações entre cliente/servidor. A página e seus elementos são atualizados através de manipulações do HTML e CSS por meio de código JavaScript (SCOTT JR, 2015; MOLIN, 2016).

SPAs são geralmente executadas através da utilização de *frameworks*, que se concentram na interface Web do *frontend* (cliente), solicitando dados e atualizações para o *backend* através de requisições (servidor). Estas requisições normalmente se dão através de chamadas HTTP utilizando o modelo de arquitetura *Representational State Transfer* (REST) (SCOTT JR, 2015). Duas implementações bastante utilizadas do SPA são os *frameworks* Angular e ReactJS.

O AngularJS foi desenvolvido pela Google e estende o vocabulário HTML implementando o conceito de *ng-tags*, que vinculam a visualização a um ou mais modelos de dados (GOOGLE, 2021). As atualizações dos dados no AngularJS são implementadas sem fazer uma previsão sobre as dependências do componente (verificação suja), não sendo atualizado imediatamente mas sim quando o AngularJS fizer a verificação do valor (GOOGLE, 2020). Sua nova versão, chamada de Angular 2, também utiliza ng-tags mas com uma sintaxe diferente. Ele não utiliza verificação suja como seu antecessor, mas árvores de componentes onde os pais estão em um nível mais alto na árvore do que seus filhos. Cada componente possui uma classe detectora de alterações, onde o pai pode atualizar seus filhos se uma alteração for detectada. Ele também pode carregar código dinamicamente durante a execução, fornecendo assim um melhor desempenho que o AngularJS (GOOGLE, 2024).

O ReactJS, criado pelo Facebook, é um *framework* que se baseia no conceito

de Virtual DOM, permitindo a criação de novas árvores DOM por meio de JavaScript. As atualizações de dados no ReactJS são realizadas utilizando um algoritmo de comparação, o qual desencadeia uma nova renderização da página sempre que um componente é modificado. Para detectar essas mudanças no Virtual DOM, o ReactJS compara as árvores DOM em todos os níveis e realiza uma nova renderização quando uma diferença é identificada. Além disso, o React possui sua própria extensão de JavaScript chamada JSX, que adiciona uma sintaxe semelhante ao XML, proporcionando uma abordagem mais declarativa e intuitiva para a construção de interfaces (FACEBOOK, 2024).

2.3 TPS

Um *Tag Path* (TP) é a representação compacta em texto da localização de uma *tag* em uma página Web, descrevendo seu caminho absoluto desde a raiz até sua folha na árvore DOM. O caminho de uma *tag* pode ser denotado como uma sequência ordenada de seus nós ancestrais, separados por um separador (como por exemplo o caractere "/"), e representa um caminho exclusivo de seu nó raiz até ela mesma (MIAO *et al.*, 2009; FANG *et al.*, 2018). A Tabela 1 apresenta uma comparação entre um código HTML simples e sua representação em TP.

Código HTML	**Tag Path**
<html>	html
<head>	html/head
<title>Título da página</title>	html/head/title
</head>	-
<body>	html/body
Título do texto	html/body/h1
 	html/body/br
Primeiro parágrafo	html/body/p
Segundo parágrafo	html/body/p
</body>	-
</html>	-

Tabela 1 – Tabela comparativa entre código HTML e sua representação em TP.

Já um *Tag Path Sequence* (TPS) consiste em uma sequência de símbolos (*strings*), em que cada simbolo representa um caminho de *tag* diferente. Um código numérico ascendente é atribuído a cada *string* de caminho da *tag* encontrada na árvore DOM, em ordem de aparência, onde os caminhos são construídos em profundidade de primeira ordem (VELLOSO; DORNELES, Carina F, 2013). A Figura 4 apresenta os três passos para a montagem de um TPS: (1) a árvore DOM é construída através do código HTML; (2) a árvore DOM é percorrida da raiz até cada um de seus nós,

construindo um TP; (3) o TPS é construído através de uma *string* que contém apenas os códigos dos TP.

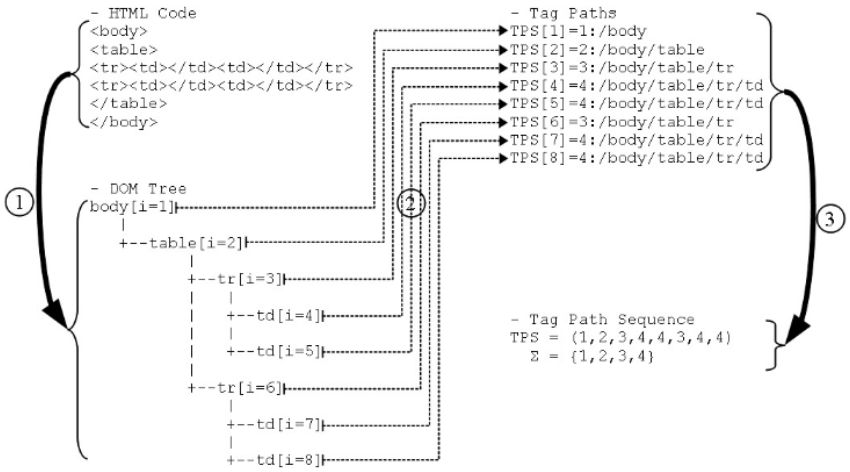

Figura 4 – Exemplo de montagem de TPS através de um código HTML (VELLOSO; DORNELES, Carina F, 2013).

2.4 *WORD EMBEDDINGS*

Word embeddings são representações vetoriais de palavras geradas por modelos de linguagem que mapeiam palavras para vetores em um espaço dimensional contínuo, onde palavras semanticamente semelhantes são localizadas próximas umas das outras. Esses vetores capturam o significado, sua semântica e contexto em relação a outras palavras no corpus de texto usado para treinar o modelo. São amplamente utilizados em tarefas de *Natural Language Processing* (NLP), como para classificação de texto, devido à sua capacidade de capturar nuances semânticas e contextuais das palavras a partir de grandes corpus de texto durante o treinamento dos modelos de linguagem (ULČAR; ROBNIK-ŠIKONJA, 2019), como por exemplo o Word2Vec (MIKOLOV; LE; SUTSKEVER, 2013), GloVe (PENNINGTON; SOCHER; MANNING, 2014), ELMo (PETERS *et al.*, 2018), BERT (DEVLIN *et al.*, 2018), etc.

2.4.1 ELMo

O *Embeddings from Language Models* (ELMo) é um método que usa um *Bidirectional Language Model* (BiLM) para derivar representações de palavras baseadas no contexto completo de uma frase (PETERS *et al.*, 2018). Para isso, sua implementação utiliza de três camadas de um modelo de linguagem, onde a primeira camada calcula a representação de uma palavra não contextualizada, com base nos caracteres da palavra, através de uma *Convolutional Neural Networks* (CNN). Em seguida, duas camadas do tipo *Long Short-Term Memory* (LSTM) bidirecionais são encarregadas de identificar o contexto da palavra na frase. Por fim, sua saída é integrada em tarefas

específicas da CNN. A Figura 5 apresenta a arquitetura do ELMo, utilizando as três camadas descritas (REIMERS; GUREVYCH, 2019).

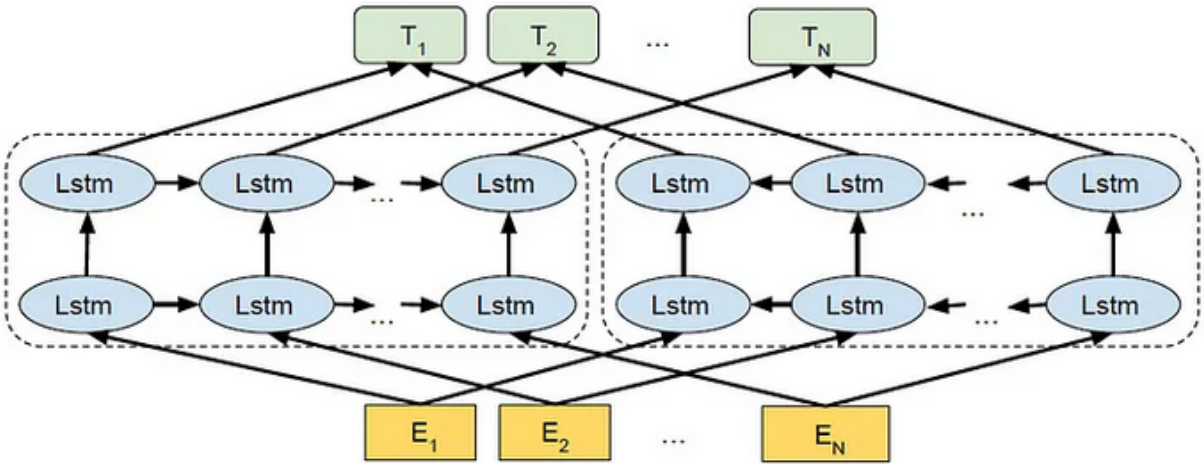

Figura 5 – Apresentação gráfica de uma arquitetura ELMo com suas três camadas (TSANG, 2022).

O ELMo supera os *word embeddings* anteriores, como Word2Vec e GloVe, em muitas tarefas de NLP e se mostra útil contra recentes modelos mais complexos, como o BERT. Isso se deve a sua capacidade de se adaptar a tarefas específicas, além de possuir uma rede neural com apenas três camadas, o que permite um treinamento mais rápido em comparação com modelos mais complexos (ULČAR; ROBNIK-ŠIKONJA, 2019).

2.5 MÉTRICAS DE AVALIAÇÃO PARA CLASSIFICAÇÃO DE TEXTOS

A avaliação da eficácia de modelos de classificação de texto geralmente envolvem uma série de métricas e técnicas que são utilizadas para comparar a performance do modelo contra outros algoritmos. Para diminuir o risco de viés nos dados utilizados e garantir que as observações são independentes umas das outras, é comum dividir de forma randômica os dados em conjuntos de treinamento, teste e validação. O conjunto de treinamento é usado para treinar o modelo e ajustar seus parâmetros e hiper parâmetros; o conjunto de teste é usado para avaliar sua eficácia durante o treinamento; o conjunto de validação serve para avaliar sua eficácia em dados desconhecidos, se aproximando ao caso real (SALAZAR *et al.*, 2022).

Para a avaliação de performance entre diferentes modelos de classificação de textos as seguintes métricas são comumente utilizadas (KANG *et al.*, 2020):

Matriz de Confusão: É uma tabela que descreve a performance do modelo de classificação em um conjunto de dados de teste, permitindo a visualização dos resultados verdadeiros positivos (VP) e verdadeiros negativos (VN), assim como seus falsos Positivos (FP) e falsos negativos (FN) durante a classificação. É usada como base para os cálculos das métricas *accuracy*, *precision*, *recall* e *F1-score*.

		Predito	
		Positivo	**Negativo**
Real	**Positivo**	Verdadeiro Positivo (VP)	Verdadeiro Negativo (VN)
	Negativo	Falso Positivo (FP)	Falso Negativo (FN)

Acurácia (*Accuracy*): É a proporção de dados classificados corretamente pelo modelo em relação ao total de amostras. No entanto, a precisão sozinha pode ser enganosa se as classes estiverem desbalanceadas.

$$Acurcia = \frac{\text{VP + VN}}{\text{Total de amostras}}$$

Precisão (*Precision*): É a proporção de dados positivos identificados corretamente pelo modelo em relação ao total de amostras identificadas como positivas. Serve para medir quantas previsões positivas estão corretas, minimizando falsos positivos.

$$Precision = \frac{\text{VP}}{\text{VP + FP}}$$

Revocação (*Recall*): É a proporção de dados positivos que foram corretamente identificados pelo modelo em relação ao total de amostras positivas. Serve para medir quantos casos positivos estão corretos, sendo útil para identificar falsos negativos.

$$Recall = \frac{\text{VP}}{\text{VP + FN}}$$

F1-score: É a média harmônica entre *precision* e *recall*. É uma métrica útil quando há um desequilíbrio entre as classes por ser sensível tanto aos Falsos Positivos quanto aos Falsos Negativos, permitindo avaliar o desempenho do modelo de forma equilibrada.

$$F1\text{-}score = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision + recall}} = \frac{2 \times \text{VP}}{\text{Total de amostras + VP - VN}}$$

3 REVISÃO DA LITERATURA

Este capítulo apresenta os trabalhos relacionados ao assunto desta dissertação obtidos a partir de uma revisão bibliográfica da literatura que apresentam trabalhos relacionados à extração de dados de páginas Web. Consultas nas bases *Association for Computing Machinery* (ACM), *Institute of Electrical and Electronics Engineers* (IEEE), Springer entre outras resultaram em aproximadamente 32.500 artigos para o período de 2010 a 2024. Com isso, obteve-se o total de 15 resultados após a aplicação de critérios de exclusão e critérios de qualidade, utilizando como critério de filtragem apenas trabalhos que apresentam propostas para extração de dados semi-estruturados e conteúdos dinâmicos e que possuam um número alto de citações em comparação aos trabalhos do mesmo ano de publicação.

Os trabalhos estão divididos em 2 grupos de extração de dados que cobrem as mais relevantes propostas de solução: extração sem uso de renderização da página Web e extração com uso de renderização da página Web. A Seção 3.1 apresenta os métodos que extraem informações sem utilizar renderização da página Web, Seção 3.2 descreve os métodos que dependem da renderização da página Web para realizar a extração dos dados e por fim a Seção 3.3 apresenta uma tabela comparativa e discussão entre as abordagens.

3.1 EXTRAÇÃO SEM USO DE RENDERIZAÇÃO DA PÁGINA WEB

Para a extração de dados de páginas Web sem uso de renderização, heurísticas podem ser aplicadas através de regras em cima conjuntos de código HTML ou texto, usando exemplos e contra-exemplos. Em conjunto a isso, também podem ser utilizadas técnicas de extração baseadas em estruturas de árvores DOM HTML, como a utilização de *Data Regions* e *Data Records*, para identificar informações por meio de alinhamento de árvores, ou *tandem repeat* para identificar informações de *substrings* frequentes de *tags* que são adjacentes diretamente umas às outras. Algoritmos que buscam detectar padrões de elementos para remover ruídos das páginas e implementações redes neurais artificiais para identificar e classificar padrões dentro de uma página Web também são utilizadas através do uso de processamento de linguagem natural e transformadores para realizar a tarefa de extração de conteúdos.

(YANG; ZHANG, 2001) apresenta uma abordagem baseada em dicas visuais para extração da estrutura semântica de documentos HTML. Por observação, sua abordagem se baseia nos estilos de layout de páginas da Web onde, em sua maioria, agregam elementos de mesma categoria de conteúdo em estilos de layout semelhantes. Com base em suas observações, foi notado durante experimentos que é bastante comum dividir os conteúdos em categorias. Cada categoria contém registros de legendas relacionadas e os registros de uma categoria normalmente são organizados

de forma a ter um estilo de *layout* visual consistente. As divisões entre as diferentes categorias são marcadas com estilos visuais ou por separadores. Porém, dois objetos HTML que parecem semelhantes visualmente não necessariamente indicam que utilizam as mesmas combinações de *tags* HTML, já que *tags* diferentes e em diferentes ordens podem gerar o mesmo resultado. Para contornar isso, o autor extrai parâmetros de renderização das *tags* que afetam atributos visuais como: tipo de fonte, estilos, tamanho e cor. Com isso, ele define a similaridade visual dos objetos aplicando algumas regras de comparação *fuzzy*. Então são criados grupos de objetos como conteúdos considerados relacionados e, a partir disso, é utilizado uma das duas medidas de similaridade visual: Similaridade Aproximada e Similaridade Paralela. A Similaridade Aproximada realiza a comparação de incompatibilidades ponderadas e omissões entre duas strings enquanto a Similaridade Paralela realiza a comparação de apenas as incompatibilidades ponderadas entre duas strings. Após isso, é utilizado um método de detecção de padrão baseado em árvores de sufixos e então são expandidos os padrões de acordo com a similaridade aproximada. Junto das árvores de sufixos também são utilizadas algumas heurísticas para localizar possíveis padrões como: similaridade de *string* obedecendo um *threshold*, ter um mínimo de repetições, não haver sobreposição de padrões, alinhamento dos padrões, utilização de parágrafos, desvios padrões, tamanho da cobertura dos padrões, substituições de sub-padrões e similaridade de prefixos entre os elementos. Em seguida, os elementos são agrupados baseados nas suas medidas de similaridade e são atribuídos identificadores únicos a eles. Para a utilização dos agrupamentos foi utilizado uma implementação baseada no algoritmo DBSCAN (ESTER *et al.*, 1996). A contagem dos padrões encontrados é feita através da árvore de sufixos e a seleção dos padrões é feita com base nas regras da heurística. Por fim, os objetos de contêiner são mesclados para os que tiverem o mesmo pai e forem semelhantes até que a execução chegue até o elemento `<body>` da árvore. Os objetos `<h1>...<h6>`, `<form>`, `<address>`, `<table>`, `<col>`, `<ul>` e `<dl>` são mantidos sem alteração, pois são tratados pelo autor como estruturas importantes.

(LIU, B.; GROSSMAN; ZHAI, 2003) propõem um algoritmo para minerar dados contínuos e não contínuos em páginas Web. Ele utiliza o conceito de nós generalizados e regiões regulares. Nós generalizados são conjuntos de nós que possuem o mesmo pai, o mesmo tipo e são adjacentes. Regiões regulares são conjuntos de nós generalizados e suas subárvores. Originalmente, o algoritmo *Mining Data Record* (MDR) foi implementado para extrair dados tabulares de páginas Web. Sua utilização foi delimitada ao uso das *tags* de tabelas como `<table>`, `<form>`, `<tr>`, `<td>`, etc. Dada uma página Web, o algoritmo funciona em três etapas: (a) constrói uma árvore de *tags* HTML baseada na página Web; (b) minera as regiões de dados da página usando a árvore de *tags* HTML criada anteriormente; (c) identifica os registros de dados de cada região de dados. O algoritmo identifica regiões de dados procurando por múlti-

plos nós de generalização usando similaridade de distância de edição onde os nós de generalização são uma combinação fixa de vários nós filhos e suas sub-árvores correspondentes. O MDR não identifica os registros de dados mais relevantes, mas informa cada sub-região repetitiva contida em uma página da Web. Ele se baseia na hipótese de que uma região de dados contém uma estrutura repetitiva em um documento, que cada estrutura repetitiva dentro de uma região de dados é um registro de dados e que os registros de dados geralmente são renderizados dentro de tabelas e formulários.

(ZHAI; LIU, B., 2005) propõem um algoritmo nomeado de *Data Extraction based Partial Tree Alignment* (DEPTA) que combina as técnicas do MDR e *Align and Extract Data Items* (AEDI) para a extração de informação de registros contíguos e não contínuos. O algoritmo consiste em primeiro identificar registros na página Web e em seguida alinhar e extrair os dados dos registros identificados. Para segmentação e identificação dos dados, o DEPTA utiliza uma melhoria do algoritmo MDR, chamado de MDR-2, para encontrar todos os dados que são formados por *tags* de tabelas e formulários (`<table>`, `<form>`, `<tr>`, `<td>`, etc) através de alinhamento parcial da árvore. Ele opera em uma árvore de *tags* construída de acordo com informações de renderização visual, onde as subárvores de cada registro são rearranjadas em uma única árvore e são alinhados utilizando o novo algoritmo. Além disso, é mencionado que as informações de lacunas são incorporadas para eliminar combinações de falsos nós, mas nada é dito a respeito de sua realização. Para o alinhamento dos dados é utilizado um novo algoritmo de alinhamento parcial da árvore, que é executado na saída do MDR-2, otimizando assim o processamento. Após todos os registros serem identificados, as subárvores de cada registro são rearranjadas em uma única árvore, onde são alinhados utilizando o novo algoritmo.

(SIMON; LAUSEN, 2005) propõem uma ferramenta chamada *Visual Perception-based Extraction of Records* (ViPER) para realizar a extração de informações de forma totalmente automática, sendo capaz de extrair e discriminar, através da percepção visual do usuário da página da Web, a relevância dos conteúdos que contenham informações repetitivas na página. Ele busca melhorar a técnica de extração realizada no algoritmo MDR pela identificação de *substrings* frequentes de *tags* que são adjacentes diretamente umas às outras (*tandem repeat*) e informações de contexto visual, relatando apenas os *data records* relevantes de acordo com a percepção visual de uma página da Web. O ViPER primeiro identifica e classifica potenciais padrões repetitivos com relação à percepção visual do usuário da página da Web, baseado na localização e no tamanho dos elementos correspondentes na página Web. Em seguida, é utilizado a técnica *Multiple Sequence Alignment* (MSA) para alinhar as subsequências correspondentes do padrão de maior peso. Inspirado pelo trabalho do DeLa (WANG; LOCHOVSKY, 2003), que utiliza da detecção de repetições contínuas em múltiplos níveis através de árvores de sufixo, o ViPER utiliza de *tandem repeat* para lidar com

elementos repetitivos, que frequentemente descartam possíveis correspondências.

(VELLOSO *et al.*, 2014) apresenta um algoritmo para segmentação e remoção de ruído de páginas Web através da utilização de TPS como representação da página Web. Seu objetivo é localizar a região principal de uma página Web com conteúdo estruturado e eliminar o restante da página, que é considerado ruído, para elevar a precisão da extração dos dados. Para isso, o algoritmo busca por posições no TPS onde é possível dividi-la em duas regiões, de modo que suas regiões tenham conjuntos de TPS completamente distintos e, portanto, representando regiões diferentes da página. Para isso o algoritmo se baseia em duas premissas: diferentes regiões de uma página Web são descritas por diferentes TPS e sites com conteúdo estruturado possuem sua principal região representada de forma mais densa que as demais regiões. Para isso, o algoritmo recebe como entrada a página HTML, extrai sua árvore DOM e a converte em uma representação TPS. Possuindo o TPS do HTML, são executadas buscas pela região principal da página, filtrando do alfabeto os símbolos com frequência inferior a um *threshold*, tornando o algoritmo de busca mais robusto e resistente ao ruído. Por fim, são removidos da árvore DOM original os ruídos encontrados, deixando apenas o conteúdo da região principal.

(PANDARGE; CHAKKARWAR, 2017) propõem uma técnica para extrair automaticamente dados baseados na Web e alinhar essas informações em forma de tabela usando o algoritmo *Combining Tag and Value Similarity* (CTVS) (SU *et al.*, 2012). Para isso, ele primeiro constrói uma árvore de *tags* HTML para descobrir os valores dos objetos, identificam as regiões de dados que incluem vários registros, segmenta os registros, alinha os campos de dados correspondentes de vários registros e os coloca em uma tabela de banco de dados. Após a montagem da árvore de *tags* HTML, a identificação é executada de cima para baixo na árvore de *tags* HTML para encontrar regiões de dados em que começam a partir do nó inicial. Em seguida, ele detecta os dados acumulando e combinando sequências de *tags* adjacentes de um elemento específico. Se a semelhança média de duas regiões de dados for maior ou igual a 0.6 essas duas regiões são mescladas em uma única região de dados. Para alinhar os registros são utilizados alinhamento de pares e alinhamento de dados holístico. No alinhamento de pares o método CTVS depende da percepção de que os itens correspondentes ao mesmo campo devem ter o mesmo tipo de dados e conter a mesma *string*, pois os resultados correspondem a uma mesma consulta. No alinhamento de dados holísticos, são organizados os dados de resultados da consulta em uma tabela global, o que significa que todos os itens de atributos semelhantes são organizados em um mesmo campo da tabela.

(WAI *et al.*, 2017) apresenta um *framework* para extração de registros de dados da Web com estrutura HTML variável utilizando um classificador binário para a identificação inicial dos nós de dados e ruídos em uma página da Web. Aproveitando o uso

de *Deep Learning* (DL) e utilizando rótulos para separar dados de ruídos e melhorar o processo automatizado de extração de dados estruturados, foi construída uma rede neural para aprender postagens de fóruns, integrando com a técnica de MDR existente no DEPTA. Sua estrutura aprende as características dos nós HTML localizados nas regiões que contém os registros de dados de interesse nas páginas. Primeiro, é utilizado o uso de aprendizado profundo para identificar se cada nó HTML em uma página da Web pertence ou não à área de interesse. Os rótulos dos nós são então incorporados ao MDR para melhorar seu processo de extração da região de dados. O *framework* não captura a semântica do conteúdo, mas sim suas propriedades físicas, como o posicionamento relativo e as propriedades visuais dos nós. Ao total, são identificados 17 propriedades físicas: localização absoluta e relativa, tamanho absoluto/relativo, distância até o nó pai, distância média para nós irmãos, distância média até nós filhos, *tag* HTML, fonte do texto pertencente ao nó, cor do texto pertencente ao nó, opacidade do texto pertencente ao nó, fonte do texto pertencente ao nó, cor de fundo, estilo de borda, raio da fronteira, largura da borda, cor da borda, largura da margem e largura do preenchimento.

(GUO, J. *et al.*, 2019) apresenta uma abordagem automática e um sistema protótipo para extração de registros de dados de sites chamada *Redundancy-Driven Data Extraction* (RED). Ela é baseada em cima do padrão de interfaces de busca: o resultado de uma pesquisa é apresentada como uma lista paginada de resultados onde cada registro contém os principais atributos sobre um único objeto e links para a página dedicada aos detalhes desse objeto. O RED aproveita a redundância inerente entre os registros de resultados e as páginas de detalhes correspondentes para projetar um método eficaz de inferências de regras de extração, de forma totalmente não supervisionada e independente de domínio. Cada um é capaz de extrair os valores de um determinado atributo, por exemplo, o preço de um produto, de todos os registros de uma página de resultados. O RED gera esse conjunto de regras executando três etapas sequenciais: geração das regras de extração, busca de redundância e remoção de ruído. Na primeira, são gerados os conjuntos de regras de resultado e as regras de detalhes, selecionando as regras de um fragmento de expressões XPath (CONSORTIUM *et al.*, 1999) que inclui regras de extração corretas. Na segunda etapa, busca identificar pares de regras de resultado/detalhe que são consideradas redundantes, pois extraem os mesmos valores para todos os objetos. Para isso, foi desenvolvida uma técnica de segmentação suave que aproveita a redundância intra-site subjacente: é explorada a presença de links de navegação que apontam em cada registro de resultado para a página de detalhes correspondente. Por fim, são executados métodos para encontrar e descartar pares de regras que exibem alguma redundância limitada, mas não suficiente para provavelmente serem regras de extração corretas. Isso é feito através de um processo de limpeza de ruído que visa separar a redundância entre o

padrão de publicação de resultado/detalhe para aquela decorrente de acidente.

(VELLOSO *et al.*, 2020) propõem uma técnica automática e computacionalmente eficiente, usando processamento de sinais e aprendizado de máquina supervisionado para detectar regularidades e padrões na estrutura de páginas Web, classificando os dados extraídos como conteúdo ou ruído. Para isso, o algoritmo segmenta a página Web, detecta as regiões de dados dentro do documento, identifica os limites de início e fim dos registros, alinha os registros encontrados e os classifica como conteúdo ou ruído. Antes de extrair os próprios registros, primeiro são isoladas as regiões que contêm dados estruturados, pois regiões estruturadas têm estrutura semelhante e são contíguos, apresentando um comportamento cíclico no seu TPS. Para identificar as regiões de dados, o algoritmo calcula o contorno do TPS através da maior altura do TPS, montando um conjunto com todos os intervalos onde a diferença do contorno for maior que zero. Após isso, mescla os elementos que se intersectam entre os conjuntos gerados e utiliza uma regressão linear para cada conjunto gerado. Se a região calculada obtiver um coeficiente angular acima de um certo *threshold* ela é removida do conjunto final. Para cada região identificada, são analisados os TPS utilizando o algoritmo *Fast Fourier Transform* (FFT). A FFT fatora a sequência em seus componentes, de modo que as sequências cíclicas exibirão picos mais altos em frequências que correspondem ao período principal da sequência. Para o alinhamento dos registros é utilizado um método de aproximação chamado *Center Star*, que utiliza a *edit distance* para alinhar duas sequências, inserindo espaços nas sequências quando um desalinhamento é encontrado. Por fim, para classificar quais regiões são conteúdo ou ruído são utilizadas seis características de região extraídas da sequência do documento: tamanho da região, posicionamento central, posicionamento horizontal, posicionamento vertical, alcance de região e proporção do registro.

(XIE *et al.*, 2021) introduz um *framework* chamado WebKE, que extrai triplas de conhecimento de páginas semiestruturadas da Web. Para isso, foi utilizado um pipeline para lidar com o conteúdo da Web semiestruturada, baseada no BERT. Porém, como as principais implementações do BERT têm um limite de comprimento máximo de entrada de 512 *tokens*, o que é muito pequeno para páginas da Web, também foi implementado um codificador de conteúdo da Web chamado HTMLBERT. Ele utiliza o BERT como modelo de linguagem de marcação para extrair as informações de layout visual e a estrutura de árvore DOM. Para executar a extração das informações, ele inicia o processamento do conteúdo através de um *Machine Reading Comprehension* (MRC) em diferentes granularidades, primeiro detectando *Area of Interest* (AOI) no nível do nó da árvore DOM e, em seguida, extraindo triplas relacionais para cada área. Para definir uma AOI, um nó da árvore DOM deve estar junto de seus nós filhos, onde seu comprimento é medido pelo comprimento de todos os *tokens* da área, e também deve conter campos de informação. Para detectar uma AOI, é usado um

MRC para decidir quais nós contém campos de informação para posterior extração de conhecimento. Se houver apenas um elemento em um nível, segue-se para o próximo nível recursivamente até que o comprimento de uma área seja menor que o comprimento máximo dos modelos subsequentes ou nenhuma área no ramo atual seja de interesse. Para a extração das relações e formações das triplas é usada uma base de conhecimento, onde assumi-se que a ela seja capaz de rotular as amostras de forma assertiva.

(ZHOU *et al.*, 2022) propõem um método simples para extrair atributos de interesse dos tipos *título*, *autor*, ISBN13, *editor* ou *nenhum* das páginas de detalhes de vários sites chamado *Learning Transferable Node Representations for Attribute Extraction* (LANTERN). Ele visa construir um novo modelo transferível para reduzir esforços humanos e extrair atributos de sites não vistos de várias verticais, que são domínios específicos organizados e categorizados baseados em assunto ou setor. Para isso ele busca simplificar as árvores DOM para extrair conhecimento informativo e transferível para a tarefa de extração de atributos, construindo uma representação rica para cada nó da árvore DOM sem usar nenhum recurso visual. Primeiramente, as árvores DOM são simplificadas para extrair recursos de contexto para cada nó. Todos os recursos textuais são alimentados em um codificador de texto para gerar uma densa incorporação semântica e são adicionados também a posição relativa de cada nó. A incorporação de nós combinada é usada para prever o tipo de nó através de uma classificação binária para obter a probabilidade de correspondência para cada tipo de atributo e, ao final, é selecionado o atributo com maior probabilidade da previsão. É utilizado um codificador de texto LSTM-CNN hierárquico para codificar os recursos de nível de caractere e nível de palavra. A CNN codifica as incorporações a nível de caractere e a LSTM bidirecional obtém a representação da sequência XPath, de modo que possa utilizar todas as *tags* significativas da sequência. Também é calculada a Similaridade de Cosseno (SC) para modelar suas relações semânticas. Durante a inferência das verticais, é alterado de um modelo multi classe para um classificador binário, já que cada vertical tem uma quantidade de atributos diferentes. Por fim, é aplicado um *softmax* para obter o maior valor predito.

(REIPS *et al.*, 2023) propõem uma ferramenta de *scraping* de notícias online que realiza a filtragem das notícias relevantes para o usuário através de *strings* de interesse pré-definidas. Essas notícias são armazenadas em uma base de dados relacional, onde o sistema executa a limpeza e o pré-processamento dos dados, a fim de extrair e identificar as entidades relevantes do texto original. Após isso, é realizado a vetorização das entidades através da técnica de *Term Frequency - Inverse Document Frequency* (TF-IDF) e a distinção entre notícias relevantes e não relevantes através de uma filtragem manual ou automática. Na filtragem manual, o usuário é convidado a destacar dentre dez notícias oferecidas as que possuam relevância para ele. Caso não

existam, outras dez notícias são oferecidas e o processo é refeito. Em cada iteração, as notícias atuais são agrupadas às anteriores, não havendo perda das já identificadas como relevantes pelo usuário. Na filtragem automática, são utilizados três algoritmos para executar a tarefa de classificação: *Naive Bayes*, *Random Forest* e *Multi-Layer Perceptron*. Com as notícias filtradas, é realizado a classificação dos dados utilizando a SC aplicado sobre os vetores TF-IDF. A ferramenta também armazena o histórico das extrações e a origem dos dados.

3.2 EXTRAÇÃO COM USO DE RENDERIZAÇÃO DA PÁGINA WEB

Para a extração de dados de páginas Web com uso de renderização, os trabalhos buscam realizar a extração de informações através da percepção visual do usuário sobre a página. As abordagens necessitam, em algum momento, utilizar de um navegador para processar a página. Também podem utilizar algoritmos *Visual Blocks*, que extraem as informações posicionais e as propriedades visuais das páginas renderizadas, para realizar a tarefa de extração de informações.

(LIU, W.; MENG, X.; MENG, W., 2010b) apresenta um abordagem que se concentra na extração de registros de dados e itens de dados organizados regularmente em páginas da Web. Para isso, ela executa quatro etapas principais: construção de árvore *Visual Block*, extração de registro de dados, extração de item de dados e geração de *wrapper* visual. As informações visuais das páginas da Web usadas por ele incluem principalmente informações relacionadas ao layout da página da Web (localização e tamanho) e fonte. Uma árvore *Visual Block* é uma segmentação de uma página da Web. Para transformar uma página em uma árvore *Visual Block* e extrair a informação visual, foi usado o algoritmo *Vision-based Page Segmentation* (VIPS) (CAI *et al.*, 2003). A árvore *Visual Block* tem três propriedades interessantes: (a) o bloco X contém o bloco Y se X for um ancestral de Y; (b) X e Y não se sobrepõem se não satisfazem a propriedade (a); (c) os blocos com o mesmo pai são organizados na árvore de acordo com a ordem dos nós correspondentes que aparecem na página. O autor utiliza a proporção entre o tamanho da região de dados e o tamanho de toda a página em vez do tamanho real para localizar a região que contém todos os registros de dados. Para extrair os registros de dados são usadas algumas premissas como: os registros de dados geralmente são alinhados à esquerda em uma região de dados, todos os dados são adjacentes, dados adjacentes não se sobrepõem e o espaço entre registros adjacentes é o mesmo. Além disso, os itens que possuem a mesma semântica utilizam uma mesma fonte para elementos textuais e um mesmo tamanho para imagens, compartilhando o mesmo tipo de posição. Também se pressupõem que o primeiro dado em cada registro de dados é sempre do tipo obrigatório e sua apresentação segue uma ordem fixa. Blocos que não seguirem essas delimitações são tratados como ruído pelo autor. Para processar e extrair os itens dos dados, é proposto um algoritmo de

multi-alinhamento passo a passo de forma holística. Inicialmente, todos os itens de dados são alinhados pela ordem de seus registros correspondentes. Quando encontrado itens de dados opcionais, que não aparecem em alguns registros, essas posições vagas serão preenchidas com o item em branco predefinido. Isso garante que todos os registros de dados estejam alinhados e tenham o mesmo número de itens de dados no final. Para gerar os *wrappers* visuais existem dois *components* propostos: o *Vision-Based Data Record Wrappe* (ViDRE) e o *Vision-Based Data Item Wrapper* (ViDIE). O ViDRE localiza a região de dados na árvore *Visual Block* e, em seguida, extrai os registros de dados dos blocos filho da região de dados. Já o ViDIE agrupa itens de dados de diferentes registros de dados em colunas ou atributos, de modo que os itens de dados na mesma coluna tenham a mesma semântica.

(ANDERSON; HONG, 2013b) propõem uma abordagem que utiliza fontes comuns de evidência que humanos usam para entender os registros de dados em uma página de consultas como regularidade estrutural, semelhança visual entre registros e semelhança de conteúdo entre registros. Itens como barras de navegação e anúncios são tratados como ruídos. Para validar sua ideia é implementado um protótipo de software chamado *rExtractor*. Para realizar a extração dos dados é utilizada uma abordagem de começar a busca no centro da página, para se afastar dos blocos de ruído nas bordas e se aproximar da área de maior prioridade para registros de dados. Então é traçada uma Espiral Ulam, que organiza números inteiros positivos em forma de espiral de sentido horário, com os números primos sendo indicados de alguma forma ao longo da espiral (WEISSTEIN, 2002). A Espiral Ulam foi especialmente selecionada por cobrir a maior proporção possível da área de maior prioridade, antes de atingir as bordas da página. O crescimento exponencial da espiral combinado com sua direção de deslocamento (sentido horário) garante que ela mostre um viés para a área entre o centro e o canto superior esquerdo da página, cobrindo assim mais da área de prioridade mais alta. A espiral termina quando encontra um bloco básico pela primeira vez, onde é chamado de bloco semente. Para representar uma página da Web renderizada é utilizado o *Visual Block Model* (VBM), o que permite acessar as informações posicionais e as propriedades visuais de cada item na página. Em seguida, o autor utiliza um método próprio para medir a semelhança visual entre os VBMs, seguido algumas definições das relações espaciais entre esses *Visual Blocks* para determinar a aparência visual de cada bloco de forma flexível. Para isso, foram escolhidas propriedades que historicamente desfrutam de boa representação em vários navegadores e são mais amplamente usadas por designers de páginas Web para definir a aparência visual das páginas de consulta como, por exemplo, `fontWeight`. Ela deve decidir se dois blocos visuais têm similaridade visual, de largura ou de conteúdo baseada em alguns critérios: para terem similaridade visual todas as propriedades visuais de ambos os blocos devem iguais; para ter similaridade de largura dois blocos

visuais devem ter as propriedades de largura dentro de um limite de 5 *pixels* um do outro; para terem similaridade de conteúdo é utilizado variante do índice Jaccard (REAL; VARGAS, 1996) para medir a similaridade dos conjuntos semelhantes de blocos filhos. O bloco semente está contido dentro de vários blocos. Ao isolar apenas os blocos nos quais o bloco semente está realmente contido, é identificado o conjunto de blocos de registro candidatos. Em seguida, são filtrados os conjuntos de todos os blocos da página, descartando qualquer bloco que não tenha a mesma largura como um dos blocos de registro candidatos.

(FAYZRAKHMANOV *et al.*, 2018) propõem uma abordagem e um algoritmo para gerar automaticamente *wrappers* HTTP, visando eliminar o primeiro nível de interação, junto ao navegador e realizar a extração de dados exclusivamente usando as interações HTTP correspondente ao segundo nível, minimizando o número de interações HTTP e eliminando as que não fazem requisições aos dados desejados. O algoritmo para isso foi nomeado de *FastWrap* e recebe como entrada um *wrapper* visual juntamente com seus parâmetros de entrada e produz como saída um *wrapper* HTTP sem navegador correspondente. O *wrapper* de saída extrai os mesmos dados que o *wrapper* de entrada, interagindo diretamente com os servidores do site e sem requerer carregamento e renderização da página Web. Esta abordagem funciona executando o *wrapper* de entrada com seus parâmetros de entrada para coletar e analisar a sequência de trocas de *Request-Response* entre o *wrapper* de entrada e o servidor Web. Esses dados são usados para identificar as interações HTTP de destino com os registros de saída desejados em suas respostas. Além disso, os registros fornecidos são usados para gerar seletores de dados para extrair os dados do conteúdo e transformá-los em uma representação estruturada. Eles são usados para induzir corretamente o *wrapper* de saída, que extrai os mesmos dados que o *wrapper* de entrada. Isso torna a abordagem aplicável a uma ampla variedade de *wrappers* visuais e evita vinculá-la a qualquer sistema específico.

3.3 TABELA COMPARATIVA E DISCUSSÃO

Separando os trabalhos entre os que não necessitam renderizar a página em algum momento do processo e os que necessitam, o número de trabalhos levantados que buscam resolver o problema sem utilizar de renderização é maior do que os que buscam pela alternativa da renderização. Essa tentativa de eliminar a utilização da renderização se dá pelo alto custo de processamento quando se usa uma *engine* de navegação para executar a renderização. Entretanto, essas abordagens são as mais robustas pois tendem a focar mais em como o usuário final do sistema enxerga o conteúdo da página, desconsiderando como está codificado e quais ferramentas foram empregadas pelos desenvolvedores.

Também é possível notar que em trabalhos que não utilizam de renderização,

há uma tendência de tentar utilizar dicas visuais utilizadas pelos desenvolvedores na criação dos elementos da página Web. Porém, os trabalhos tendem a limitar o uso das dicas visuais a alguns elementos específicos, baseados nas tarefas que é pretendido executar. Essa decisão se deve ou pela complexidade de adotar uma abordagem mais ampla ou pela redução de dimensionalidade na execução dos modelos propostos.

Nota-se também que de forma direta ou indireta, os trabalhos buscam fazer uma distinção do que é ou não ruído dentro de uma página Web. Isso auxilia a simplificar a extração das informações, pois elementos como menus, rodapés, propagandas, etc tendem a apresentar uma mesma forma de implementação e estarem posicionadas em partes específicas nas páginas, onde removê-las reduz o universo de busca. Também é possível notar que com o passar dos anos a abordagem do uso de árvores e heurísticas foi sendo substituída por técnicas mais modernas, utilizando aprendizagem de máquina para executar algumas tarefas.

Trabalho	**Renderização**	**Dicas Visuais**	**Remove ruídos**	**Conteúdo dinâmico**	**Abordagem**
(YANG; ZHANG, 2001)		x			*Trees* e Heurísticas
(LIU, B.; GROSSMAN; ZHAI, 2003)				x	*Trees* e Heurísticas
(ZHAI; LIU, B., 2005)				x	*Trees* e Heurísticas
(SIMON; LAUSEN, 2005)		x		x	*Trees* e Heurísticas
(LIU, W.; MENG, X.; MENG, W., 2010b)	x	x		x	VBM e Heurísticas
(ANDERSON; HONG, 2013b)	x	x		x	VBM e Heurísticas
(VELLOSO *et al.*, 2014)			x		*Trees* e Heurísticas
(PANDARGE; CHAKKARWAR, 2017)			x		*Trees*
(WAI *et al.*, 2017)		x	x		*Trees* e DL
(FAYZRAKHMANOV *et al.*, 2018)	x	x		x	*Wrappers* e Heurísticas
(GUO, J. *et al.*, 2019)			x		*Trees*
(VELLOSO *et al.*, 2020)			x		*Trees* e Heurísticas
(XIE *et al.*, 2021)			x		*Trees* e BERT
(ZHOU *et al.*, 2022)			x		LSTM-CNN e SC
(REIPS *et al.*, 2023)				x	TF-IDF, SC e ML
Este trabalho		x	x	x	*Trees* e ELMo

Tabela 2 – Tabela comparativa entre os trabalhos.

4 TRABALHO PROPOSTO

O trabalho desenvolvido, chamado de *Enhanced Data Representation for Extraction in Web* (EDREW)[1], é uma abordagem de extração de dados em páginas da Web que aproveita a estrutura do DOM HTML e sua sequência de *tags* para gerar uma representação textual dos elementos da página, visando inferir seu conteúdo relevante e reduzir ao máximo a captura de ruídos. Este capítulo é destinado a detalhar o processo de extração do conteúdo realizado pelo EDREW.

4.1 VISÃO GERAL

A arquitetura geral do EDREW é composta por cinco fases, conforme apresentado na Figura 6: (a) sanitização e montagem da árvore de elementos; (b) adição de *tokens* especiais; (c) criação do *Reverse Enhanced TPS* (reTPS); (d) classificação do reTPS; (e) inferência do modelo. Por fim, o conteúdo extraído é apresentado através de uma lista de itens ou em um formato personalizado através da utilização de uma função `builder` que permite a manipulação dos dados antes de seu retorno.

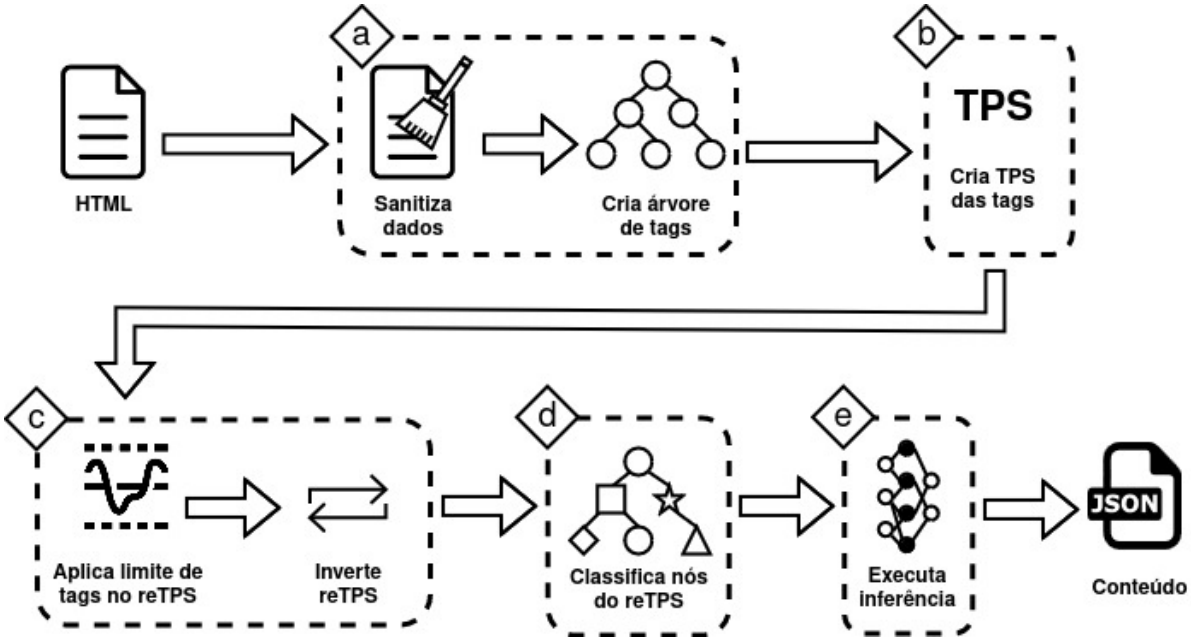

Figura 6 – Processo de execução do EDREW da aquisição de uma página HTML até a extração de seu conteúdo.

Para sua utilização, o EDREW necessita como entrada um texto HTML de uma página Web, um modelo treinado ELMo e, opcionalmente, uma função `builder` como parâmetros. O `builder` serve para personalizar a saída do modelo caso o usuário não deseje receber os dados extraídos em formato de lista, podendo formatar o resultado conforme desejar. No Algoritmo 1 é possível notar que o EDREW realiza quatro passos

[1] https://github.com/mcanzian/EDREW

para fazer a extração do dados: a montagem dos *nodes* da árvore do HTML (linha 1), a preparação dos dados dos *nodes* para serem utilizados como entrada do modelo (linha 2), carregamento e inferência do modelo (linha 3) e, por fim, realização de um pós-processamento para apresentação dos dados ao final da execução (linha 4). Ao final da execução, são retornados os dados processados e prontos para uso (linha 5).

Algorithm 1 EDREW

Require: *html*
Require: *model*
Require: *builder*
1: *nodes* ← *Extractor*(*html*)
2: *data* ← *prepare*(*nodes*)
3: *predict* ← *inference*(*data*, *model*)
4: *result* ← *postProcess*(*predict*, *builder*)
5: **return** *result*

4.2 MONTAGEM DA ÁRVORE DE ELEMENTOS

A preparação dos dados dos elementos HTML para serem utilizados pelo modelo é realizada através do Algoritmo 2. Na fase de preparação da representação textual é realizada uma requisição HTTP para uma página Web ou a leitura de um arquivo local, a fim de obter o corpo HTML da página em forma de texto. O texto obtido é utilizado como entrada do algoritmo, onde é executado a sanitização do texto (linha 1), a fim de remover excesso de caracteres de espaçamento, quebras de linha e tabulações.

Algorithm 2 Extractor

Require: *html*
1: **global** *countTags* ← 0
2: *html* ← *sanitize*(*html*)
3: *tree* ← *buildTree*(*html*)
4: *nodes* ← *classifyNodes*(*tree*)
5: **return** *nodes*

A seguir, o `Extractor` faz uma chamada ao Algoritmo 3 (linha 3), que recebe como entrada o texto HTML sanitizado e executa a construção da árvore de elementos para a criação dos *reTPS*. O `buildTree` também possui algumas variáveis de controle, para auxiliar em sua execução:

- `HTML_TAGS_REGEX`: Expressão regular usada para extrair as *tags* do texto HTML;
- `countTags`: Contador de *tags* válidas utilizadas durante a extração para referenciar o posicionamento da *tag* na página;

- `level`: Nível atual da *tag* na árvore de elementos;
- `currentNode`: *Node* atual da árvore de elementos;
- `isScript`: *Flag* para indicar se uma *tag* se localiza dentro de um `<script>`;
- `wt`: Prefixo `#WT` para ser utilizado em *tags* do tipo *Wrapper Tag*;
- `root`: *Node* raiz da árvore de elementos;
- `appendedTag`: *Flag* para indicar se a *tag* foi adicionada na árvore de elementos;
- `tagSequence`: TP da *tag* extraída do texto HTML;
- `tagSequenceH`: TPS da *tag* extraída do texto HTML;
- `classesCounter`: Contador do número de classes únicas utilizadas nas *tags* extraídas.

Após a inicialização das variáveis auxiliares, é iniciado a execução de uma expressão regular, genérica e fixa, no corpo do HTML da página (linha 10) para desmembrar os elementos da página Web. Isso permite que manipulações dos elementos da página sejam realizadas pelo EDREW sem restrições aos padrões do HTML, permitindo capturar customizações criadas nas páginas, como por exemplo *tags* customizadas.

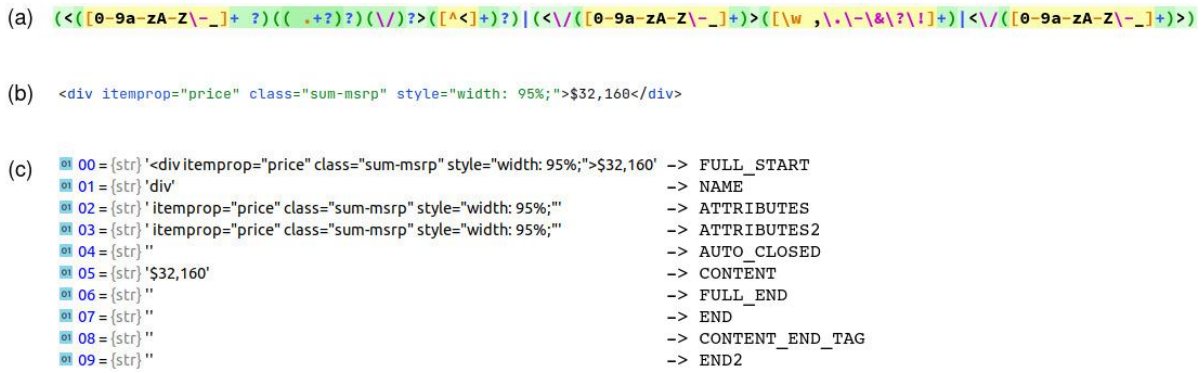

Figura 7 – (a) Expressão regular utilizada para extrair os elementos do HTML; (b) exemplo de *tag* utilizada; (c) exemplo do agrupamento dos itens extraídos pela expressão regular sobre a *tag* apresentada.

A expressão regular separa cada elemento da página entre sua *tag* e seus atributos, para serem utilizados durante a criação da árvore de elementos, conforme mostrado na Figura 7. Cada agrupamento formado pela expressão regular representa uma característica da *tag*, onde esses grupos serão utilizados pelo EDREW para executar suas tarefas. Os grupos criados são respectivamente: *tag* completa (`FULL_START`), nome da *tag* (`NAME`), primeiro grupo de atributos (`ATTRIBUTES`), segundo grupo de atributos (`ATTRIBUTES2`), auto fechamento da *tag* (`AUTO_CLOSED`), conteúdo da *tag* (`CONTENT`), fechamento completo da *tag* (`FULL_END`), fechamento da *tag* (`END`), conteúdo residual após fechamento da *tag* (`CONTENT_END_TAG`) e segundo fechamento da *tag* (`END2`).

Algorithm 3 buildTree

Require: *html*

global *HTML_TAGS_REGEX*, *countTags*
global *level* ← 0
global *currentNode* ← *null*
global *isScript* ← *false*
global *wt* ← "*#WT*"
root ← *null*
appendedTag ← *false*
tagSequence, *tagSequenceH* ← []
classesCounter ← {}
tags ← *regex*(*HTML_TAGS_REGEX*, *html*)
for each *tag* in *tags* **do**
    appendedTag ← *false*
    if *shouldSkipTag*(*tag*) **then**
        continue
    end if
    if *isWrapperTag*(*tag*, *tagSequence*, *tagSequenceH*) **then**
        continue
    end if
    if *tag*[*FULL_START*] is not empty **then**
        addNewRegularTag(*tag*, *tagSequence*, *tagSequenceH*)
    else if *tag*[*END*] ≠ "*br*" **then**
        processEndTag(*tag*, *tagSequence*, *tagSequenceH*)
    end if
    if *root* is *null* **then**
        root ← *currentNode*
    end if
end for
return *root*

A decisão de se obter as *tags* de forma manual, sem a utilização de bibliotecas propostas por trabalhos anteriores, se deu porque foi identificado que as bibliotecas utilizadas pelos trabalhos existentes se baseiam apenas nos elementos descritos no padrão da linguagem HTML, desconsiderando elementos e atributos customizados. O elemento customizado `<dhi-new-typeahead-input>` e seus atributos `_ngcontent-vmk-c53`, `formcontrolname` e `data-cy`, apresentados na Figura 8, não seriam extraídos por completo e, por consequência, haveriam perdas de informações durante a extração. A utilização da expressão regular permitiu o processamento destes dados sem comprometer a performance de extração, mas com a vantagem de total controle de todos os elementos e atributos de cada página.

```
▼<dhi-new-typeahead-input class="ng-tns-c52-0 ng-untouched ng-pristine ng-valid"
  _ngcontent-vmk-c53="" formcontrolname="keyword" data-cy="search-term-input"> event
```

Figura 8 – Exemplo de *tag* customizada utilizada pela versão atual do site Dice.

Após a identificação das *tags* pela expressão regular do Algoritmo 3, é iniciada a montagem da árvore de elementos (linha 11), baseado no relacionamento entre os elementos do DOM HTML da página. Ao criar cada elemento, são atribuídos a ele sua *tag*, seus atributos e seu conteúdo. Para isso, é feita uma iteração em cada *tag* gerada pela expressão regular, onde podem ocorrer cinco casos principais: a *tag* é do tipo `script` ou pertence a lista de *tags* excluídas (`SKIP_TAGS`) e deve ser ignorada (linhas 13 a 15), a *tag* é uma *Wrapper Tag* (WT) (linhas 16 a 18), a *tag* é uma `FULL_START` (linhas 19 e 20), a *tag* é uma `END` diferente de quebra de linha (`br`) ou a *tag* não se encaixar em nenhuma destas condições.

Algorithm 4 shouldSkipTag

Require: *tag*
1: **global** *isScript*
2: **global** *SKIP_TAGS*
3: **if** *tag*[*NAME*] = "*script*" **then**
4: *isScript* ← *true*
5: **return** *true*
6: **else if** *tag*[*END*] = "*script*" **then**
7: *isScript* ← *false*
8: **return** *true*
9: **else if** *tag*[*NAME*] in *SKIP_TAGS* or *tag*[*END*] in *SKIP_TAGS* or *isScript* **then**
10: *isScript* ← *false*
11: **return** *true*
12: **end if**
13: **return** *false*

Entre as linhas 13 e 15 do `buildTree()`, é realizado a checagem das *tags* a serem ignoradas. Caso a *tag* seja do tipo `script` ou esteja incluída na lista `SKIP_TAGS`, ela e todo o seu conteúdo devem ser ignorados através da execução do Algoritmo 4, que recebe como entrada a *tag* a ser verificada e retorna se ela é um *script* ou não. O EDREW não trata *scripts* do tipo Javascript, ECMAScripts e afins. Caso o EDREW identifique elementos HTML dentro de um *script*, estes elementos serão ignorados até que seja identificado um fechamento de *script* (`<\script>`) no corpo do HTML. Isso ocorre devido ao EDREW não saber em que local da página os elementos de dentro da *tag* `<script>` serão anexados ao corpo do HTML, em tempo de execução, o que acaba distorcendo toda a classificação dos elementos na página Web. As *tags* do conjunto `SKIP_TAGS` são referentes a estilos, metadados, *links* e *inputs*, sendo assim desconsideradas pelo EDREW. No caso de estilos, caso sejam aplicados diretamente na *tag* (*inline*) os estilos são considerados, assim como suas classes. O EDREW não considera os atributos que formam uma classe individualmente, mas a classe como um todo. Os elementos que formam o `SKIP_TAGS` são: `link`, `meta`, `noscript`, `base`, `style`, `input` e `area`.

Tags do HTML que são utilizadas envolta de outras *tags* para estilização (por exemplo `<strong>`) ou adicionar uma ação a um elemento (por exemplo `<a>`) são tratadas de forma especial pelo EDREW, sendo chamadas de WT. *Wrapper Tags* são *tags* especiais editadas pelo EDREW onde se adiciona o prefixo `'#WT'` à *tag* original do elemento. O conjunto de elementos que formam as WTs são: `span`, `a`, `b`, `strong`, `i`, `em`, `mark`, `small`, `del`, `ins`, `sub` e `sup`. Caso uma *tag* seja do tipo WT e não possua conteúdo é executado o Algoritmo 5, que recebe a *tag* a ser validada, seu TP e seu TPS como entradas e retorna se ela é ou não uma WT. Além disso, caso a *tag* seja de fato uma WT, ela é simplesmente adicionada à árvore de elementos através de um *node* e, em seguida, é indicada como o *node* atual da árvore, conforme apresentado entre as linhas 3 e 7 do algoritmo.

Algorithm 5 isWrapperTag

```
Require: tag
Require: tagSequence
Require: tagSequenceH
 1: global WRAPPER_TAGS
 2: global wt
 3: if tag[NAME] in WRAPPER_TAGS and tag[CONTENT] is empty then
 4:     node ← addNode(tag, wt + tag[NAME], tagSequence, tagSequenceH)
 5:     currentNode ← node
 6:     return true
 7: end if
 8: return false
```

De volta ao Algoritmo 3, entre as linhas 19 a 21, é verificado se a *tag* possui algum valor em seu `FULL_START`. Caso positivo, é executado o Algoritmo 6, que recebe como entradas a *tag* a ser adicionada a árvore de elementos, seu TP e seu TPS. Inicialmente, são realizadas algumas verificações na *tag* (linhas 6 a 11), a fim de remover possíveis distorções que a página possa gerar. A primeira verificação realizada é checar se o nome da *tag* é diferente de `br` (linha 6). Caso seja, será adicionado um novo `level` ao elemento (linha 7); caso contrário, o elemento será considerado do mesmo `level` que o elemento anterior. *Tags* do tipo `br` são quebras de linha e sua utilização desmedida pode gerar distorções na classificação do modelo, indicando que os elementos daquela região da página pertencem a um grupo diferente do seu grupo real. Também é verificado se a *tag* pertence ao grupo de elementos, chamado neste trabalho de `REPEATING_TAG_STRUCTURE` (linhas 9 a 11). Este grupo serve para indicar que a *tag* é utilizada para formar uma listagem ou tabulação dos elementos. Porém, da mesma forma que a *tag* `br`, a utilização excessiva destes elementos na página acaba distorcendo sua classificação de posicionamento no *layout* e, por isso, deve ser desconsiderado. O grupo `REPEATING_TAG_STRUCTURE` é constituído pelos seguintes elementos: `li`, `ol`, `ul`, `option`, `p`, `dd`, `dl`, `dt`, `br`, `th`, `tr`, `td`, `menuitem`. A repetição

destas *tags* acaba alterando o posicionamento das demais *tags* a sua volta em relação à página como um todo e, consequentemente, isso acaba alterando a sua classificação pelo modelo.

Algorithm 6 addNewRegularTag

```
Require: tag
Require: tagSequence
Require: tagSequenceH
 1: global wt
 2: global level
 3: global countTags
 4: global currentNode
 5: global REPEATING_TAG_STRUCTURE
 6: if tag[NAME] ≠ "br" then
 7:     level ← level + 1
 8: end if
 9: if tag[NAME] not in REPEATING_TAG_STRUCTURE then
10:     countTags ← countTags + 1
11: end if
12: node ← addNode(tag, wt + tag[NAME], tagSequence, tagSequenceH)
13: if tag is auto close then
14:     tagSequence.pop()
15:     tagSequenceH.pop()
16: else
17:     currentNode ← node
18: end if
```

Ao final das verificações do Algoritmo 6, a *tag* é adicionada a um *node* da árvore de elementos (linha 12). Por fim, é verificado se a *tag* é do tipo auto fechamento (linhas 13 a 18), como por exemplo `<img/>`. Se afirmativo, a *tag* é removida das listas de controle `tagSequence` e `tagSequenceH`, caso contrario ela é indicada como o *node* atual da árvore através da variável `currentNode`.

Dando sequência ao Algoritmo 3, entre as linhas 21 e 23, caso a *tag* não possua um valor em seu `FULL_START`, é verificado se seu `END_TAG` não é uma quebra de linha (`br`). Caso não seja, é executado o Algoritmo 7, que recebe como entradas a *tag* a ser processada, seu TP e seu TPS. Inicialmente, é verificado se a última *tag* da lista de controle `tagSequence` é do tipo *Wrapper Tags*, através de seu prefixo (linhas 5 a 7). Caso seja, o prefixo WT é removido da *tag*.

Após isso, é verificado se a última *tag* do `tagSequence` é do tipo `END` (linha 8). Caso positivo, ela é removida da lista de controle `tagSequence` e, caso não seja uma *Wrapper Tag*, também é removida da `tagSequenceH`. Por fim, ainda dentro da verificação do tipo `END`, caso seu valor em `END` seja de uma *tag* dentro da lista `FORMATTER_TAGS`, o conteúdo do *node* atual (`currentNode`) será concatenado com o valor da extração da última *tag*. Isto ocorre para unir conteúdos formatados dentro de uma *tag*.

Algorithm 7 processEndTag

Require: *tag*
Require: *tagSequence*
Require: *tagSequenceH*
1: **global** *wt*
2: **global** *level*
3: **global** *currentNode*
4: **global** *FORMATTER_TAGS*
5: *lastTag* ← *tagSequence*.*last*()
6: **if** *lastTag* starts with *wt* **then** *lastTag* ← *lastTag*.*substr*(3, *lastTag*.*length*)
7: **end if**
8: **if** *lastTag* is *END* **then**
9: *poppedTag* ← *tagSequence*.*pop*()
10: **if** *poppedTag* not starts with *wt* **then**
11: *tagSequenceH*.*pop*()
12: **end if**
13: **if** *tag*[*END*] IN *FORMATTER_TAGS* **then**
14: *currentNode.content* ← *currentNode.content* + *tag*[*CONTENT_END_TAG*]
15: **end if**
16: *level* ← *level* – 1
17: **end if**

`FORMATTER_TAGS` são *tags* utilizadas para dar algum tipo de formatação visual a um elemento, onde as seguintes *tags* são utilizadas pelo EDREW: `b`, `strong`, `i`, `em`, `mark`, `small`, `del`, `ins`, `sub` e `sup`. No exemplo seguinte, um conteúdo formatado da seguinte forma `texto <b>formatado<\b> aqui` é transformado em `texto formatado aqui` no valor de conteúdo do *node*. Isso evita que durante a inferência dos conteúdos realizada pelo modelo, apenas a palavra `formatado` seja indicada como conteúdo relevante, e não a frase inteira.

4.3 ADIÇÃO DE *TOKENS* ESPECIAIS

A criação e adição dos *nodes* à árvore de elementos é feita através do Algoritmo 8. Inicialmente, é gerado um TPS simples da *tag* (linha 2), onde são concatenados seus elementos antecessores até sua *tag* atual, separados pelo separador `<#SEP>`, utilizando o Algoritmo 9. Também é criada uma variável concatenando o nome da *tag* com a sua posição na página (linha 3), separadas pelo caractere *pipeline* (|), que será utilizado durante a classificação da *tag*. Após isto, é realizada a criação do *node* da árvore de elementos (linha 4), onde o *node* recebe como atributos a *tag* com seu `countTags`, seu TPS, seus atributos, seu conteúdo e seu *node* pai (que é definido através do *node* atual da árvore).

Após a criação de um *node*, são realizados alguns tratamentos nas informações do TPS da *tag* (linha 5), a fim de aprimorar a inferência do modelo na próxima fase

Algorithm 8 addNode

Require: *tag*
Require: *tagName*
Require: *tagSequence*
Require: *tagSequenceH*
1: *appendedTag* ← *true*
2: *tps* ← *toTPS*(*tagSequence*, *tagName*)
3: *tn* ← *tag*[*NAME*] + "|" + *countTags*
4: *node* ← *Node*(*tn*, *tps*, *tag*[*ATTRIBUTES*], *tag*[*CONTENT*], *currentNode*)
5: *node.tpsh* ← *toTPS*(*tagSequenceH*, *tagAttrs*(*tag*))
6: *node.tpsi* ← *toTPSi*(*node.tpsh*, *true*, 20)
7: **return** *node*

do EDREW. Para isso, são criados *tokens* especiais baseados no TPS da *tag*, que auxiliam o modelo a identificar padrões e características únicas nos elementos. Isto visa facilitar que o modelo diferencie elementos similares da página em contextos diferentes. Por exemplo, uma `<div>` posicionada no início da página pode ter uma classificação diferente de uma `<div>` posicionada mais ao centro da página, mesmo apresentando características similares.

Algorithm 9 toTPS

Require: *pathSequence*
Require: *tag*
1: *pathSequence* ← *tag*
2: *tps* ← []
3: **for** each *p* in *pathSequence* **do**
4: *tps* ← *p* + *TAG_SEPARATOR*
5: **end for**
6: **return** *tps*

O processo da criação do TPS é refeito, porém agora classificando as informações da *tag* através de suas cinco propriedades: `id`, `propriedades`, `classes`, `estilos` e `conteúdo`. Por padrão, o EDREW utiliza um conjunto de 14 valores pré-definidos para classificar *tokens* especiais às *tags*, conforme demonstrado na Tabela 3.

Token especial	Descrição
<#SEP>	Separador entre elementos de um TPS do EDEW.
<#CLSS>	*Token* indicando que os próximos elementos pertencem a uma classe da *tag*.
<#PROP>	*Token* indicando que os próximos elementos pertencem a uma propriedade da *tag*.
<#ID>	*Token* indicando que a *tag* possui um atributo do tipo *ID*.
<#VAL>	*Token* indicando que a *tag* possui um atributo com o valor *value*.
<#DESC>	*Token* indicando que a *tag* possui um atributo com o valor *description*.
<#NAME>	*Token* indicando que a *tag* possui um atributo com o valor *name*.
<#DATE>	*Token* indicando que a *tag* possui um atributo com o valor *date*.
<#CONT>	*Token* indicando que a *tag* possui um atributo com o valor *content*.
<#DET>	*Token* indicando que a *tag* possui um atributo do tipo *detail* ou *details*.
<#CNT>	*Token* indicando que a *tag* possui um valor atribuído a ela.
<#$>	*Token* indicando que a *tag* possui um valor monetário atribuído a ela.
<#HIDE>	*Token* indicando que a *tag* está oculta do usuário na renderização da página.

Tabela 3 – Apresentação do conjunto de 14 *tokens* especiais padrões do EDREW.

Caso uma *tag* possua um atributo do tipo `id`, os *tokens* especiais `<#SEP><#ID>` são adicionados ao `tagSequenceH`. Elementos com atributo `id` tendem a ser utilizados para alguma função de interação dentro da página, mas apenas o atributo `id` isoladamente não indica, necessariamente, a relevância da *tag*. Em *tags* que possuam atributos do tipo `propriedades` e `classes` são adicionados *tokens* especiais com base na análise dos atributos e seus valores. Para isso, é realizada uma busca nos textos das propriedades baseada no conjunto de valores pré-definidos. Estes valores foram baseados em análises realizadas durante os experimentos nos elementos das páginas do *dataset* SWDE, onde frequentemente são utilizados para definir os mesmos conjuntos de dados, mesmo que em diferentes páginas. Os valores utilizados são: `price`, `value`, `description`, `name`, `date`, `content` e `details`. Também é possível estender a lista de valores através da configuração do EDREW, onde pode-se adicionar por exemplo os valores `author` e `isbn` com seus respectivos *tokens* especiais.

Para marcação de `estilos`, caso a *tag* esteja oculta na página, através da utilização de estilos como `display:none` ou `visibility:hidden` é adicionado um *token* especial do tipo `<#HIDE>` ao `tagSequenceH`, sinalizando ao modelo que o elemento está oculto na página. Por fim, a marcação de conteúdo através do *token* especial `<#CNT>` é realizada caso a *tag* esteja localizada em uma folha da árvore de elementos e tenha algum conteúdo associado a ela. Caso seja identificado que o conteúdo inicie com um cifrão ($), um *token* especial do tipo `<#$>` é adicionado junto ao *token* de conteúdo, indicando um possível valor monetário.

A ordem de inserção dos *tokens* especiais no `tagSequenceH` segue a seguinte ordem, permitindo que o modelo possa prever o próximo *token* baseado em sua sequencia pré-definida: `<tag><id><classes><propriedades><conteúdo>`. O processo de adição de um *node* a árvore de elementos e da montagem do TPS com *tag* especiais pode ser vista na Figura 9. Primeiro é feito a iteração pelo corpo do HTML (a), é realizado a extração da *tag* (b), é gerada a lista com todos os elementos da hierarquia da *tag* (c), seu TPS simples com separador (d) e por fim seu TPS com a classificação dos

elementos (e).

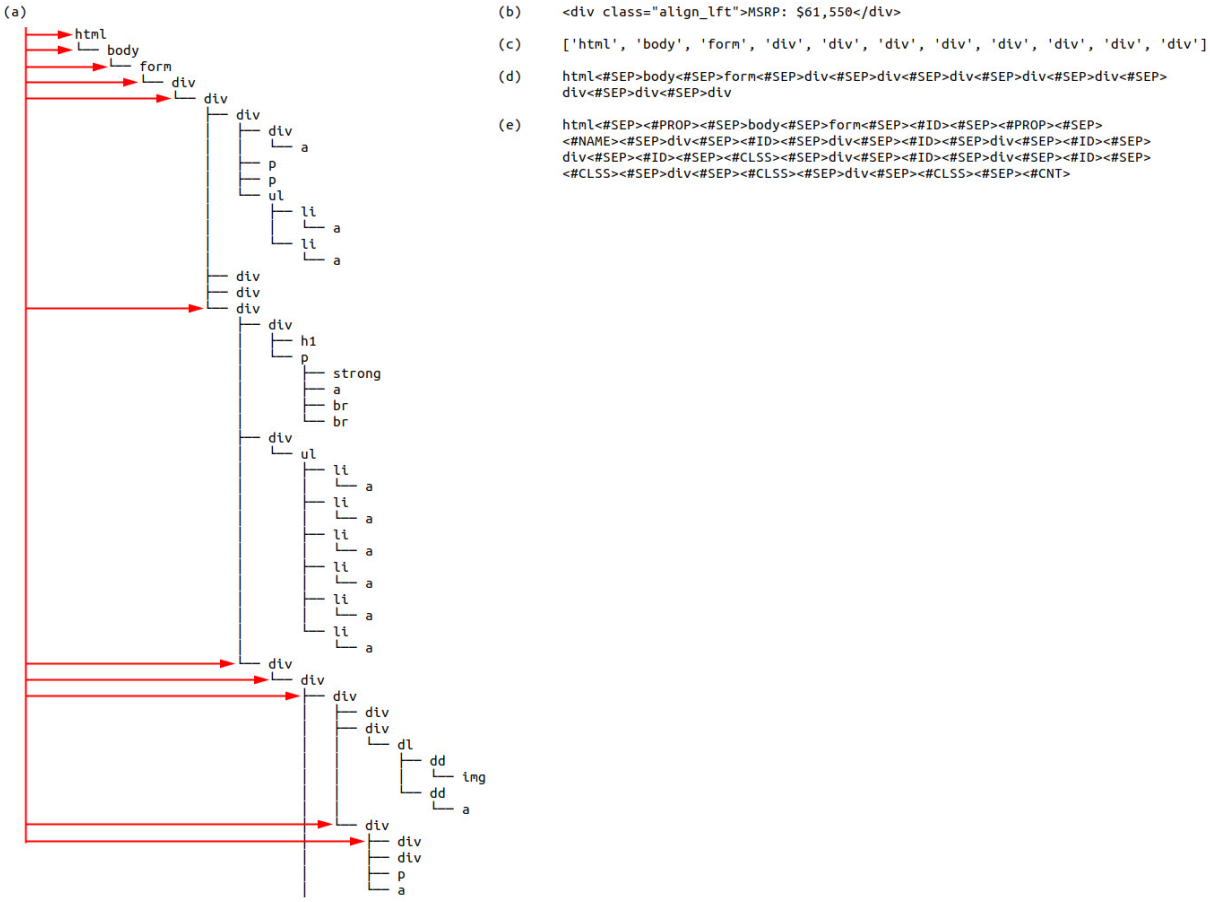

Figura 9 – Exemplo de montagem de um TPS pelo EDREW iniciando pela árvore de *tags*.

4.4 *REVERSE ENHANCED TPS* (RETPS)

Ao final do Algoritmo 8, é feita a criação do reTPS (linha 6) através da utilização do Algoritmo 10. Ele recebe como entrada o TPS e o *threshold* do total de *tags*. O valor de *threshold* utilizado para o corte de elementos é de 20, baseado em testes realizados com o *dataset* SWDE através da vertical *auto* e com a métrica *F1-score*, conforme apresentado na Figura 10.

O Algoritmo 10 inicia com a criação de uma lista dos elementos do TPS, separando os valores pelo separador `<#SEP>` (linha 3), e iterando seus elementos (linhas 4 a 12) a fim de atualizar o valor da variável global `maxLevel`. Isso é feito através da separação do valor do elemento pelo caractere *pipeline* (linha 8), onde o primeiro valor representa a *tag* e o segundo valor a sua posição na página. Caso o valor atual seja maior que o valor em `maxLevel` seu valor é atualizado (linhas 9 a 12). Caso a *tag* seja uma *tag* especial ou um `br` o valor é ignorado.

Algorithm 10 toTPSi

Require: *pathSequence*
Require: *maxTags*
global *maxLevel*
global *TAG_SEPARATOR*
tpsi ← *pathSequence.split*(*TAG_SEPARATOR*)
for each *tagWithLevel* in *tpsi* **do**
    if *tagWithLevel* starts with "<#" OR *tagWithLevel* = "*br*" **then**
        continue
    end if
    t ← *tagWithLevel.split*("|")
    level ← *t*[1]
    if *level* > *maxLevel* **then**
        maxLevel ← *level*
    end if
end for
tpsi2 ← []
for each *tagWithLevel* in *tpsi* **do**
    tpsi2 ← *tagWithLevel*
end for
tpsi2.reverse()
tpsi ← []
c ← 0
for each *tag* in *tpsi2* **do**
    tpsi ← *tag* + *TAG_SEPARATOR*
    if *c* > *maxTags* **then**
        break
    end if
    c ← *c* + 1
end for
return *tpsi*

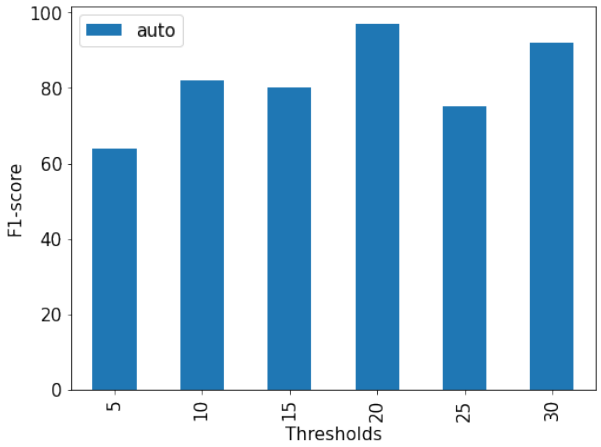

Figura 10 – Testes de diferentes *thresholds* utilizando a vertical *auto* e a métrica *F1-score*.

Após isso, é gerada uma cópia da lista de elementos do TPS (linhas 14 a 17) e é aplicada a reversão destes elementos (linha 18). Esta inversão busca trazer os elementos localizados nas folhas da árvore para o início da lista e levar a sua raiz para o final, removendo do reTPS os elementos que supostamente mais se repetem entre os elementos da página, já que os elementos mais ao final do reTPS estão mais próximos da raiz da árvore. Com isso, é realizada a iteração dos elementos, transformando-os em uma *string* novamente através de sua concatenação com o separador `<#SEP>` (linhas 19 a 27).

Durante a iteração dos elementos invertidos, também é executado um corte no total de elementos utilizados pelo reTPS, onde o número de elementos permitido por um reTPS é feito através de um *threshold* do seu total de *tags* (linhas 23 a 26). A limitação dos elementos no reTPS serve para que o modelo foque nos elementos mais próximos da folha, onde normalmente estão localizados os conteúdos das páginas. Por fim, é retornado o reTPS da *tag*. A Figura 11 apresenta um exemplo de reTPS gerado pelo EDREW.

```
<#CNT><#SEP>div|34<#SEP>div|21<#SEP><#ID><#SEP>div|21<#SEP><#ID><#SEP>div|21<#SEP><#ID><#SEP>div|21<#SEP><#ID><#SEP>
div|13<#SEP><#ID><#SEP>div|5<#SEP><#ID><#SEP>div|3<#SEP><#ID><#SEP>form|3<#SEP>body|2<#SEP>html|1
```

Figura 11 – Exemplo de reTPS que será utilizado pelo modelo.

4.5 CLASSIFICAÇÃO DE POSICIONAMENTO

Como último passo do Algoritmo 2, são realizadas as classificações das *tags* baseadas em seu posicionamento na página, conforme apresentado pelo Algoritmo 11. A classificação de sua posição relativa à página é concatenada à *tag* no reTPS, separado por um *pipeline* (|), conforme é demonstrado na Figura 11.

O Algoritmo 11 recebe como entrada a árvore de elementos gerada pelo HTML da página, através de sua raiz. A partir disso, é realizada a iteração por todos os elementos da árvore de forma recursiva (linhas 20 a 22) a fim de classificar todos os seus *nodes*, gerando assim o reTPS final. Para cada *node* da árvore, é realizada a separação de suas *tags* através do separador `<#SEP>` (linhas 3 e 4) para então fazer a iteração em todas as *tags* do *node* da árvore (linha 6). Caso a *tag* seja uma *tag* especial ou um `br`, a *tag* é apenas adicionada à lista de *tags* e é avançado para a próxima *tag* da iteração. Caso contrário, a *tags* é separada entre nome da *tag* e sua posição relativa à página através do separador *pipeline* (|). Então é executado o Algoritmo 12, que recebe como entradas a posição da *tag* e o contador de *tags* válidas.

A classificação do reTPS através de seu posicionamento é calculado baseado na Série de Fibonacci e serve para criar uma diferenciação entre mesmas *tags* em diferentes posições da página. Seu posicionamento é calculado baseado na altura do elemento relativo ao tamanho do documento HTML. A Série de Fibonacci foi escolhida

Algorithm 11 classifyNodes

```
Require: root
global countTags
global TAG_SEPARATOR
SEP ← "<#SEP>"
nodeTags ← root.tpsi.split(SEP)
tags ← []
for each tag in nodeTags do
    if tag starts with "<#" OR tag = "br" then
        tags ← tag
        continue
    end if
    tagName ← tag.split("|")[0]
    tagLevel ← tag.split("|")[1]
    levelClassification = getFibonacci(tagLevel, countTags)
    tags ← tagName + "|" + levelClassification
end for
root.reTPS = " "
for each t in tags do
    root.reTPS = root.reTPS + t + TAG_SEPARATOR
end for
for each child in root.children do
    classifyNodes(tree, countTags)
end for
return root
```

Algorithm 12 getFibonacci

```
Require: number
Require: maxNumber
fibonacci ← [1,2,3,5,8,13,21,34,55,89,100]
nearestFibonacciNumber ← 1
if maxNumberisnotnull then
    number ← (number/maxNumber) ∗ 100
end if
for each f in fibonacci do
    if number >= f then
        nearestFibonacciNumber ← f
    else
        break
    end if
end for
return nearestFibonacciNumber
```

pelo modo que ela é calculada, na qual cada número inteiro subsequente corresponde à soma dos dois números inteiros anteriores. Essa sequência gera uma distribuição de regiões na página, que fará com que elementos posicionados abaixo do número 21 e acima do número 55 de Fibonacci tendam a ter menos relevância comparados aos elementos mais centrais da página, de forma isolada, conforme apresentado nas Figuras 12 e 13. Para elementos dentro dos intervalos dos números da sequência, foi escolhida a sequência mais próxima da altura atual do elemento. A Figura 14 apresenta as etapas de transformação de um texto HTML para o formato de reTPS, onde é possível notar a geração de novas informações extraídas de uma *tag* HTML.

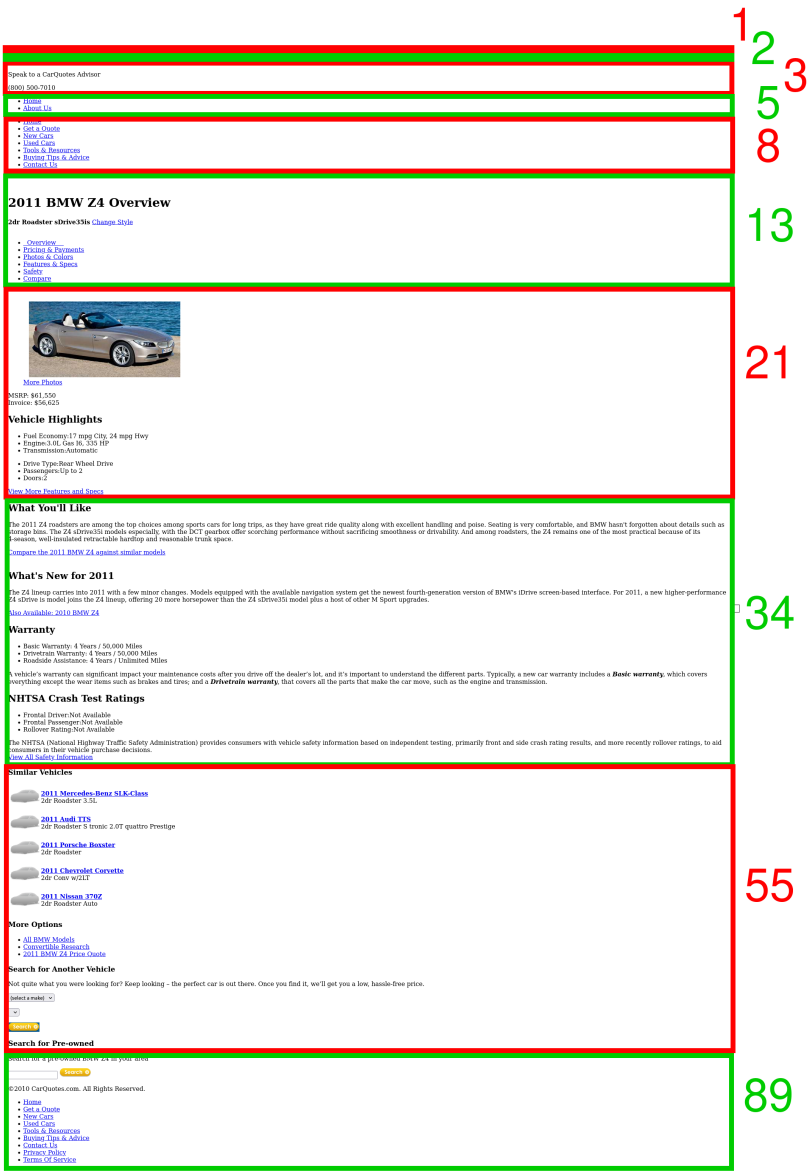

Figura 12 – Exemplo visual da distribuição da classificação de Fibonacci das *tags* pelo site.

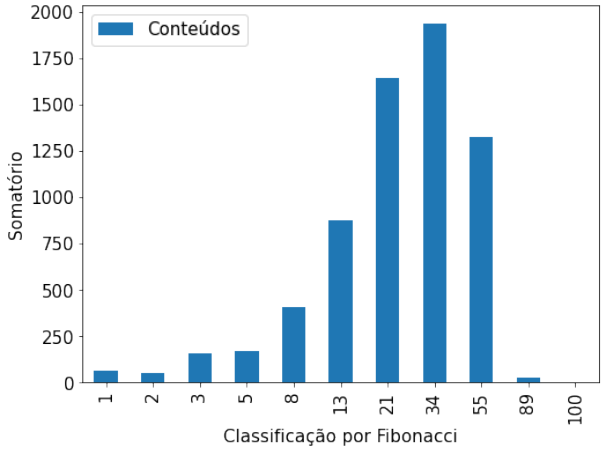

Figura 13 – Distribuição dos conteúdos do *dataset* SWDE pela classificação por sequência de Fibonacci, baseada na altura do elemento relativo ao tamanho do documento HTML.

(a) MSRP: $61,550

(b) ['html', 'body', 'form', 'div', 'div', 'div', 'div', 'div', 'div', 'div', 'div']

(c) html<#SEP>body<#SEP>form<#SEP>div<#SEP>div<#SEP>div<#SEP>div<#SEP>div
<#SEP>div<#SEP>div<#SEP>div

(d) html|1<#SEP><#PROP><#SEP>body|4<#SEP>form|5<#SEP><#ID><#SEP><#PROP><#SEP>
<#NAME><#SEP>div|6<#SEP><#ID><#SEP>div|7<#SEP><#ID><#SEP>div|39<#SEP>
<#ID><#SEP>div|61<#SEP><#ID><#SEP><#CLSS><#SEP>div|62<#SEP><#ID><#SEP>
div|63<#SEP><#ID><#SEP><#CLSS><#SEP>div|71<#SEP><#CLSS><#SEP>div|72<#SEP>
<#CLSS><#SEP><#CNT>

(e) <#CLSS><#SEP>div|71<#SEP><#CLSS><#SEP><#ID><#SEP>div|63<#SEP><#ID><#SEP>
div|62<#SEP><#CLSS><#SEP><#ID><#SEP>div|61<#SEP><#ID><#SEP>div|39<#SEP>
<#ID><#SEP>div|7<#SEP><#ID><#SEP>div|6<#SEP><#NAME><#SEP><#PROP><#SEP>
<#ID><#SEP>form|5'

(f) <#CNT><#SEP><#CLSS><#SEP>div|72<#SEP><#CLSS><#SEP>div|71<#SEP><#CLSS><#SEP>
<#ID><#SEP>div|63<#SEP><#ID><#SEP>div|62<#SEP><#CLSS><#SEP><#ID><#SEP>
div|61<#SEP><#ID><#SEP>div|39<#SEP><#ID><#SEP>div|7<#SEP><#ID><#SEP>div|6<#SEP>
<#NAME>

(g) <#CNT><#SEP><#CLSS><#SEP>div|21<#SEP><#CLSS><#SEP>div|21<#SEP><#CLSS><#SEP>
<#ID><#SEP>div|21<#SEP><#ID><#SEP>div|13<#SEP><#CLSS><#SEP><#ID><#SEP>
div|13<#SEP><#ID><#SEP>div|13<#SEP><#ID><#SEP>div|2<#SEP><#ID><#SEP>div|2<#SEP>
<#NAME>

Figura 14 – Processo de transformação de um texto HTML para reTPS utilizando uma *tag* que contém a informação de um preço.

4.6 INFERÊNCIA DO MODELO

Ao finalizar a execução do `Extractor()` pelo EDREW é retornada a árvore de elementos do HTML, com seu reTPS finalizado. Após isso, é iniciada a fase de inferência dos elementos pelo modelo no Algoritmo 1 (linhas 2 a 4). Para isso, são realizadas três fases de processamento dos dados:

1. Preparação dos reTPS para um formato de lista de tuplas;

2. Inferência dos reTPS gerados através de um modelo treinado;
3. Pós processamento dos resultados do modelo, para uma melhor apresentação.

Na fase de preparação dos dados, o Algoritmo 13 realiza a iteração dos *nodes* da árvore gerada pelo Algoritmo 2, criando uma lista de tuplas, onde a tupla é formada pelo reTPS *tag* e seu conteúdo. Caso o elemento não tenha um conteúdo associado a *tag*, o elemento é descartado, já que não há conteúdo a ser apresentado ao final do EDREW. Até este momento, o EDREW não sabe quais são os conteúdos relevantes dentro da lista de reTPS gerada.

Algorithm 13 prepare

Require: *nodes*
1: *data* ← []
2: **for** each *node* in *nodes* **do**
3: **if** *node.content* is not empty **then**
4: *data* ← [*node.reTPS*, *node.content*]
5: **end if**
6: **end for**
7: **return** *data*

Após a preparação dos dados, a inferência é realizada pelo Algoritmo 14, que recebe como dados de entrada a lista de tuplas e o modelo que se deseja executar. Após isso, o modelo é carregado para a memória RAM (linha 1) e é realizada a inferência na lista passada (linha 2), adicionando um novo elemento na tupla chamado de `result`. A saída do modelo (linha 3) retorna a classificação binária de um elemento entre **conteúdo relevante** ou **ruído da página**. Em caso do modelo inferir que o conteúdo apresentado é um ruído, o campo `result` será preenchido com o número `0`, caso contrário, o modelo decidirá que o conteúdo é relevante e definirá o número `1` no campo `result`. Ao final do Algoritmo 14 será retornada uma lista de predições, contendo os mesmos campos da lista de entrada mais o campo `result`, que são as predições do modelo.

Algorithm 14 inference

Require: *data*
Require: *model*
1: *m* ← *load*(*model*)
2: *predict* ← *model*.*transform*()
3: **return** *predict*

Por fim, o Algoritmo 15 recebe como entrada a nova lista de tuplas e, opcionalmente, a função de `builder`. O algoritmo irá realizar a iteração dos elementos preditos realizando um filtro pelos elementos com valor `1` no campo `result` e selecionando apenas o conteúdo da lista de predições. Após isto, caso um `builder` tenha sido informado

na chamada do algoritmo, ele é executado passando como parâmetro de entrada a lista de predições filtradas e sua saída é retornada pelo algoritmo. Caso contrário, ao fim do algoritmo é retornada a lista de predições com os conteúdos identificados.

Algorithm 15 postProcess

Require: *predict*
Require: *builder*
data ← []
for each *item* in *predict* **do**
    if *item.result* = 1 **then**
        data ← [*item.content*]
    end if
end for
if *builder* ≠ *null* **then**
    result ← *builder*(*data*)
    return *result*
else
    return *data*
end if

A Figura 15 ilustra como a informação é retornada ao final do algoritmo, caso não seja utilizado nenhum `builder` para realizar a formatação da lista de resultados. Pode-se notar que além do conteúdo extraído, em verde, também foi retornado pelo EDREW um pouco de ruído página, em vermelho.

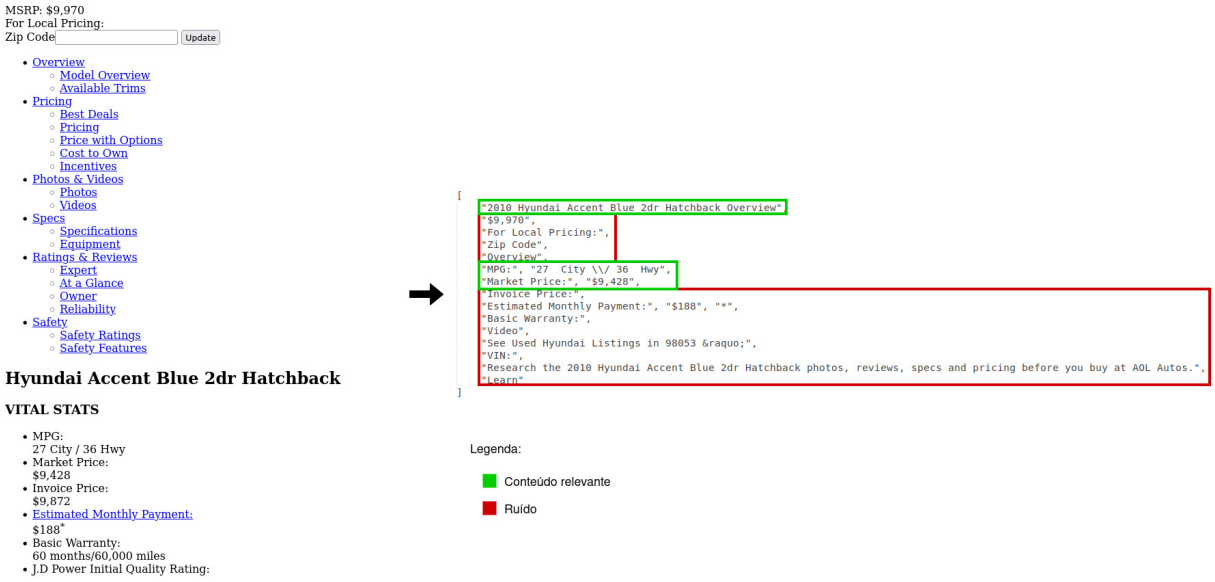

Figura 15 – Exemplo de extração de dados com poucos ruídos após o conteúdo.

Por outro lado, há casos em que o modelo também pode extrair muitos ruídos junto ao conteúdo da página, como é apresentado na Figura 16. Nela, é possível notar que houve um excesso de informações extraídas abaixo dos conteúdos, sinalizado

em vermelho. Entretanto, neste caso em específico, boa parte destas informações são referentes ao conteúdo da página, mas estão ocultadas por se tratarem de informações adicionais sobre o produto. Dependendo do caso de uso, essas informações podem ser tratadas como informações relevantes pelo usuário.

Figura 16 – Exemplo de extração de dados com excesso de ruídos após o conteúdo.

Outro ponto a se destacar é que, mesmo nos casos onde há excesso de ruído na extração do conteúdo da página, os dados extraídos tendem a ser organizados na lista de resultados nas mesmas posições para as mesmas páginas, facilitando assim a implementação de um `builder` para cada *site* em específico. Isso ocorre porque as páginas são construídas através da implementação de *templates*, onde o que são alterados entre uma página e outra é apenas o seu conteúdo. O modelo tende a ignorar o *template*, retornando apenas o que ele identifica como conteúdo, mesmo que parte desta inferência traga algum ruído associado. Ainda no exemplo da Figura 16, mesmo neste caso onde há excesso de ruído retornado, vale destacar que a página utilizada tem um total de 6329 *nodes*, dos quais 212 ruídos foram retornados junto ao resultado final. Este número representa apenas 3,35% dos *nodes* analisados da página, reduzindo o universo amostral.

5 RESULTADOS DOS EXPERIMENTOS

Este capítulo descreve os *datasets*, as configurações computacionais, os hiper parâmetros utilizados pelo modelo e como foi dividido o *dataset* no treinamento, validação e testes dos experimentos. O trabalho foi avaliado utilizando dois conjuntos de dados como base. Foi realizada a comparação dos conteúdos relevantes extraídos pela nossa abordagem contra os trabalhos LANTERN (ZHOU *et al.*, 2022) e WebKE (XIE *et al.*, 2021).

5.1 DATASETS

Para os experimentos foram utilizados dois conjuntos de dados:

- **SWDE** (*Structured Web Data Extraction*) (HAO *et al.*, 2011): conjunto de dados públicos criado em 2011 composto por 124.291 páginas da Web, provenientes de 80 sites distintos, que abrange 8 verticais de assuntos distintos.
- **N-SWDE** (*New Structured Web Data Extraction*): conjunto de dados públicos baseado nas versões atuais das páginas do SWDE, que possui 10 páginas de cada site para cada vertical, coletadas para este trabalho.

Verticais	Sites	Páginas	Páginas Únicas	Classes
auto	10	17.923	192	model, price, engine, fuel
book	10	20.000	707	title, author, isbn13, publisher, publication date
camera	10	5.258	474	model, price, manufacturer
job	10	20.000	253	title, company, location, date posted
movie	10	20.000	94	title, director, genre, mpaa rating
nbaplayer	10	4.405	27	name, team, height, weight
restaurant	10	20.000	219	name, address, phone, cuisine
university	10	16.705	146	name, phone, website, type

Tabela 4 – Apresentação dos elementos do conjunto de dados público SWDE.

A Tabela 4 apresenta algumas das informações sobre o SWDE, como suas verticais, o total de *sites* utilizados por vertical, o total de páginas por *site*, o total de páginas únicas por *site* e as classes de cada vertical. Para gerar a coluna de total de páginas únicas utilizou-se um *hash* MD5 em cada elemento das páginas do *dataset*, realizando um agrupamento pelos *hashs* dos elementos, permitindo selecionar apenas as páginas distintas de cada vertical. Observando a diferença entre as colunas Páginas e Páginas Únicas, é possível notar que apesar do elevado número de páginas do *dataset* há uma alta repetibilidade de elementos entre as páginas.

O N-SWDE foi criado para este trabalho a fim de validar o trabalho EDREW contra seus concorrentes através da extração de conteúdo de páginas Web que utilizam novas técnicas e tecnologias para além de 2011, época em que foi criado o SWDE. A Tabela 5 apresenta algumas informações sobre o novo *dataset*, apresentando as 8 verticais utilizadas, o número de *sites* que ainda seguem ativos dos *sites* do SWDE, o

Vertical	A	D	O	P	*Frameworks*
auto	8	2	0	0	AMP, Emotion, Handlebars, Mustache, Phoenix Framework, Preact, ReactJS
book	7	3	0	0	CivicTheme, Microsoft ASP.NET, Mustache, ReactJS
camera	5	5	0	0	CivicTheme, ZURB Foundation, Microsoft ASP.NET, ReactJS
job	5	4	1	0	AngularJS, Emotion, jQuery, Vue.js
movie	7	3	0	0	jQuery, Preact, ReactJS
nbaplayer	9	1	0	0	Express/JSS, Handlebars, jQuery, Microsoft ASP.NET, ReactJS, Vue.js
restaurant	5	4	0	1	AngularJS, jQuery, Preact, Vue.js
university	9	0	1	0	AngularJS, Bootstrap, Emotion, jQuery, ReactJS, Vue.js

Tabela 5 – Demonstração dos *status* das páginas atuais do novo SWDE e os *frameworks* utilizados (A: Ativo, D: Desativado, O: *Offline*, P: Privado).

número de *sites* que foram desativados, o número de *sites* que se apresentaram indisponíveis para acesso e o número de sites que se tornaram privados, impossibilitando o acesso a eles. Também na Tabela 5 é apresentado as tecnologias utilizadas pelas páginas para renderizar os conteúdos, onde sua distribuição de utilização pode ser vista na Figura 17.

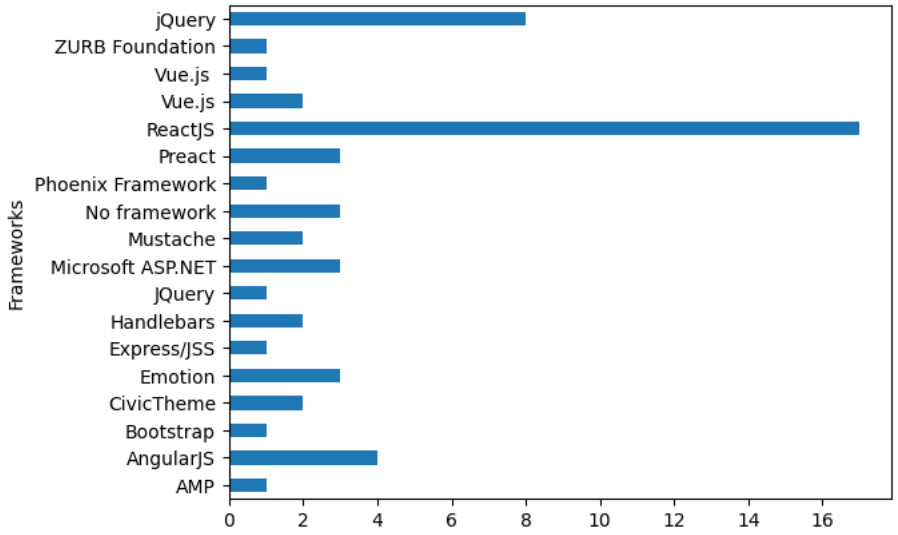

Figura 17 – Apresentação dos total de uso dos *frameworks* pelas páginas.

Pode-se notar que grande parte das páginas utilizam o ReactJS como *framework* de desenvolvimento, seguidos por jQuery e Angular, onde os três representam 53,5% do total de páginas. Para coletar as informações de quais tecnologias cada página utiliza para a apresentação visual dos elementos foi utilizada a plataforma *open source* Wappalyzer (HE; CHEN; GUO, W., 2017; RAKHMAWATI *et al.*, 2018) que identifica tecnologias em sites, como *frameworks* Web, bibliotecas de JavaScript, etc. Algumas páginas não puderam ser acessadas por estarem *offline* durante os testes, desativadas ou por terem se tornado privadas após à aquisição por outra empresa. Há também páginas que mesmo podendo ser acessadas não foi possível localizar seu conteúdo baseado no SWDE de forma manual durante a criação do N-SWDE.

5.2 CONFIGURAÇÕES

Os experimentos sobre os trabalhos EDREW e WebKE são conduzidos em um computador com sistema operacional Ubuntu 22.04.2 LTS de 64 bits, com processador Intel i7-9750H, com 16 Gb de memória RAM e com uma GPU NVIDIA GeForce GTX 1660 com 6 Gb de memória. Para o trabalho LANTERN é utilizado um computador com sistema operacional Ubuntu 20.04.1 LTS de 64 bits, com processador AMD EPYC 7742, com 1 Tb de memória RAM e com uma GPU NVIDIA NVIDIA A100 com 40 Gb de memória. Esta segunda configuração ocorreu devido ao primeiro computador não conseguir carregar o modelo do LANTERN e nem executar seus testes com apenas 16 Gb de memória RAM. Todos os trabalhos utilizaram a versão 3.8 do Python.

5.3 TREINAMENTO DO MODELO

Para o uso do ELMo, é utilizada a biblioteca *open source* de processamento de texto SparkNLP[1]. Para o *pipeline* do modelo, é utilizado como entrada o pré-processador `DocumentAssembler` para preparar os dados em um formato processável pelo SparkNLP. Para isso, o corpo do documento de entrada é transformado em um *dataframe* que extrai metadados referentes aos textos do documento, como por exemplo o tamanho de cada texto, seu início e fim, etc. Em seguida, é utilizado o anotador `Tokenizer` para separar as palavras do texto bruto em *tokens*, onde seu separador utiliza o *token* especial `<#SEP>`. Por fim, é utilizado o modelo ELMo para fazer a classificação do texto, através da utilização do `ElmoEmbeddings` por meio de um modelo pré treinado de 1 bilhão de palavras. O `ElmoEmbeddings` recebe como entrada as saídas do `DocumentAssembler` do `Tokenizer`, formando ao final um `SentenceEmbeddings`. Para a classificação do modelo é utilizado o classificador `ClassifierDLApproach` utilizando os seguintes hiper parâmetros: uma taxa de aprendizado de 0.001, um *batch size* de 1, um *dropout* de 0.5 e 50 épocas. A Figura 18 ilustra o *pipeline* de processamento do texto pelo SparkNLP até a saída com sua classificação.

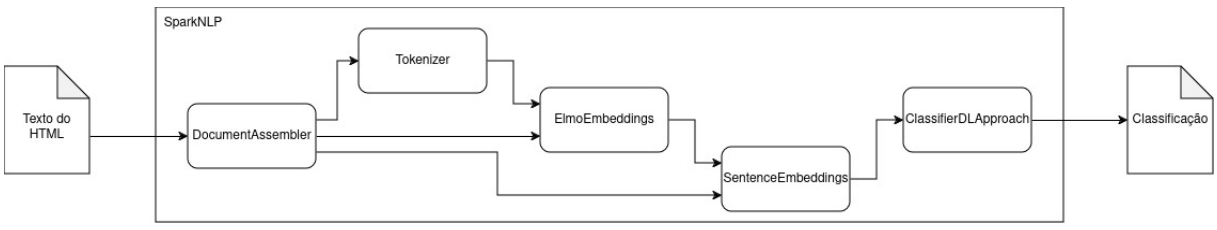

Figura 18 – *Pipeline* de execução do modelo pelo SparkNLP.

Conforme apresentado nas Figuras 19, 20 e 21, testes foram realizados para a escolha dos hiper parâmetros, onde não se notou muita evolução no modelo após

[1] https://nlp.johnsnowlabs.com

50 épocas e, entre os valores testados, as melhores configurações para a taxa de aprendizado e *batch size* foram utilizadas, mesmo que com pouco diferença entre as demais. Também foram realizados testes durante o treinamento do modelo para a escolha do classificador do texto HTML. Conforme a Figura 22, são utilizados os modelos GloVe, BERT e ELMo para realizar a tarefa de classificação, utilizando como métrica o *F1-score*, em que o modelo ELMo se saiu levemente melhor que o modelo BERT e muito superior ao modelo GloVe.

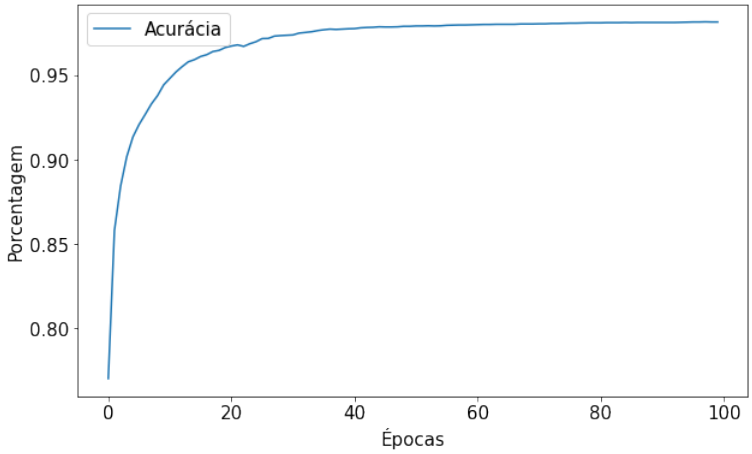

Figura 19 – Curva de evolução do treinamento do modelo baseado nas épocas.

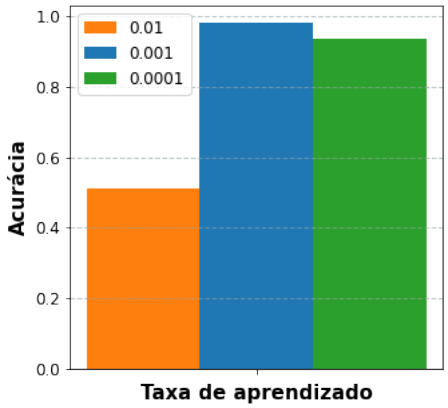

Figura 20 – Comparação entre diferentes taxas de aprendizado utilizadas pelo treinamento do modelo.

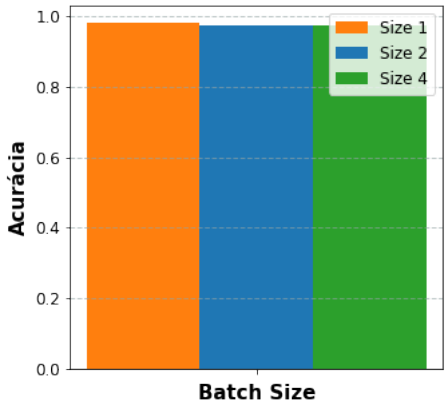

Figura 21 – Comparação entre diferentes *Batch Sizes* utilizados pelo treinamento do modelo.

Para realizar o treinamento do modelo, os dados foram divididos em dados de treino, validação e teste, utilizando uma proporção de 70/20/10 respectivamente. O treinamento, validação e teste do modelo foi realizado utilizando apenas as páginas únicas do SWDE e não o *dataset* inteiro. O *dataset* N-SWDE foi utilizado apenas na fase de teste do modelo, não participando das fases de treinamento e validação. Os dados de teste não foram utilizados durante o treinamento do modelo, sendo assim completamente desconhecidos pelo modelo.

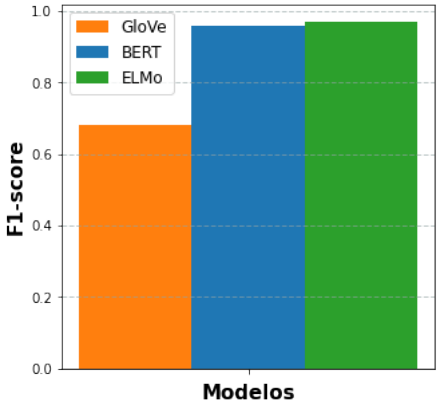

Figura 22 – Comparação entre diferentes modelos testados para a classificação do texto.

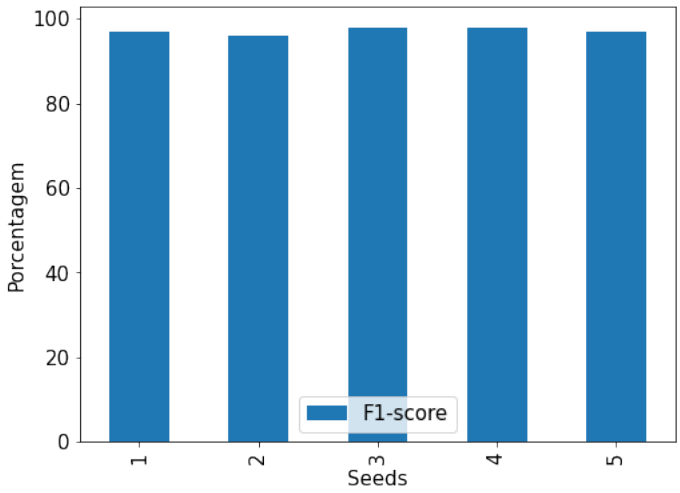

Figura 23 – Testes com diferentes valores de *seeds* durante a criação do *dataset* de treinamento, utilizando como métrica de comparação a métrica *F1-score*.

Para a criação dos *datasets* de treinamento do modelo foi utilizado um *seed* de valor 1, que serve para inicializar a distribuição dos dados de forma randômica, porém permitindo a reprodução da mesma distribuição através do valor do *seed*. Foram realizados testes com outros valores de *seeds* para validar a falta de influência do valor utilizado na criação do *dataset* sobre o modelo, conforme a Figura 23.

5.4 RESULTADOS

Para avaliar o trabalho EDREW foram realizadas comparações contra os trabalhos WebKE e LANTERN utilizando as métricas *precision*, *recall* e *F1-score*. O objetivo dos testes é verificar como cada trabalho se comporta na tarefa de identificação e extração dos conteúdos das páginas do SWDE utilizando todas as verticais, conforme apresentado na Tabela 6. Também foram realizados comparativos entre os trabalhos nas verticais *auto*, *movie*, *nbaplayer* e *university*. Os resultados apresentados foram executados localmente, utilizando os códigos disponibilizados pelos trabalhos em seus

repositórios públicos. De modo geral, o EDREW conseguiu apresentar melhores resultados na métrica *F1-score* nas verticais *auto*, *movie* e *university*. Porém, o EDREW se mostra inferior na vertical *nbaplayer* em comparação ao trabalho WebKE.

Trabalho	Precision	Recall	F1-score
EDREW (Todas)	**97.0**	**97.0**	**97.0**
EDREW (auto)	**94.0**	**98.0**	**96.0**
LANTERN (auto) (ZHOU *et al.*, 2022)	82.1	67.8	72.3
EDREW (movie)	**97.0**	**95.0**	**96.0**
LANTERN (movie) (ZHOU *et al.*, 2022)	44.9	14.3	20.5
WebKE (movie) (XIE *et al.*, 2021)	81.7	60.8	66.5
EDREW (nbaplayer)	67.0	62.0	64.0
LANTERN (nbaplayer) (ZHOU *et al.*, 2022)	42.7	27.6	33.0
WebKE (nbaplayer) (XIE *et al.*, 2021)	**82.7**	**69.3**	**71.8**
EDREW (university)	93.0	**93.0**	**93.0**
WebKE (university) (XIE *et al.*, 2021)	**96.3**	58.9	73.0

Tabela 6 – Comparativo dos trabalhos EDREW, LANTERN e WEBKE utilizando as métricas *precision*, *recall* e F1-score utilizando o *dataset* SWDE original.

Também foram realizados testes do EDREW utilizando as verticais de forma individual, conforme apresentado na Figura 24. As verticais *book*, *camera* e *restaurant* são as que possuem mais variações de *layouts* de páginas e de estruturas no HTML dentro do SWDE, conforme é demonstrado nas colunas **Páginas** e **Únicos** na Tabela 4, onde isso pode explicar a maior facilidade do modelo em identificar os conteúdos destas páginas.

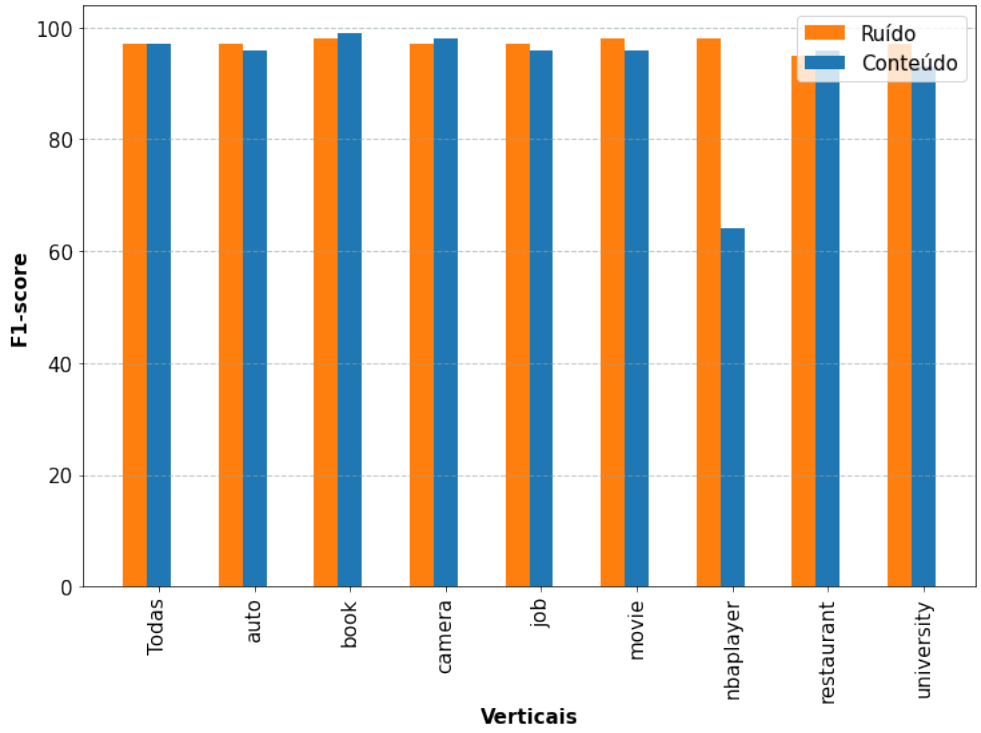

Figura 24 – Resultados do EDREW para cada vertical SWDE utilizando a métrica *F1-score*.

Por outro lado, a vertical *nbaplayer* é a que menos possui variações tanto de páginas quanto de estruturas no HTML em comparação com as demais verticais, o que pode explicar a dificuldade do EDREW em identificar os conteúdos das páginas e corrobora com o resultado apresentado contra o WebKE. Ainda na vertical *nbaplayer*, o site que mais apresenta dificuldade para extração dos conteúdos é a Wikipédia, devido ao seu *card* de informações ficar posicionado na parte superior da página em relação ao tamanho do documento e apresentar uma estrutura tabular, o que pode ser confundido com um menu de opções pelo modelo do EDREW, devido ao reTPS gerado levando em conta seus elementos e sua posição na página. Ao remover o *site* Wikipédia do conjunto de dados, seu *F1-score* sobre a extração de conteúdos sobe de 40% para 71%, o que representa um aumento de 77,5% de acerto sobre o conjunto utilizando o site Wikipédia. Analisando o EDREW contra os outros trabalhos utilizando as verticais *auto*, *movie*, *nbaplayer* e *university* do SWDE pode-se notar que o EDREW se comporta em média 23,2% acima do outros dois trabalhos na extração dos conteúdos das páginas, utilizando a métrica *F1-score*, conforme apresentado na Figura 25. Entretando, o EDREW acaba ficando levemente baixo apenas na vertical de *nbaplayer* contra o trabalho WebKE.

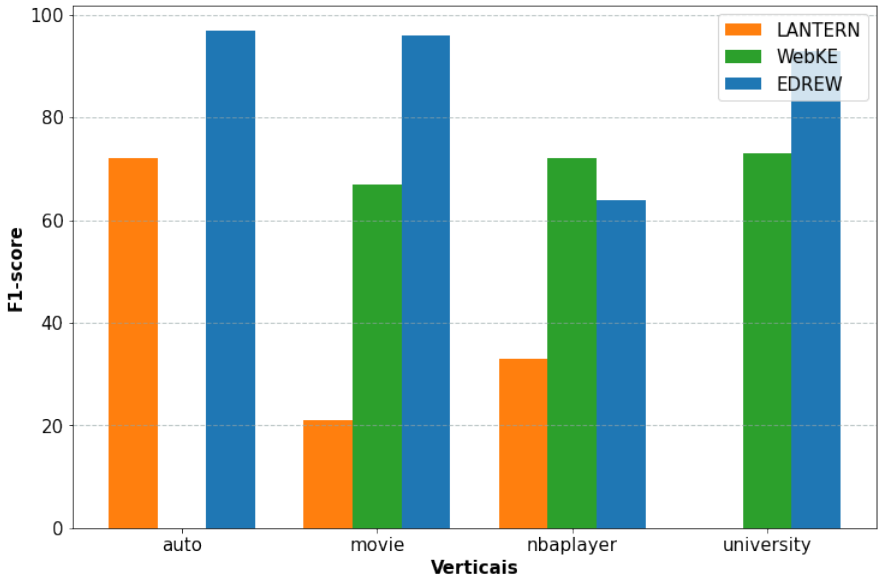

Figura 25 – Comparação entre os trabalhos LANTERN, WebKE e EDREW nas verticais de *auto*, *movie*, *nbaplayer* e *university*, utilizando a métrica *F1-score*.

Quando utilizado o N-SWDE, que contém as páginas atuais dos mesmos sites do SWDE, o EDREW destaca-se com 76% de acertos contra 42% do LANTERN, utilizando a métrica *F1-score* na verical *auto*. Isso representa 80,9% a mais de acertos do EDREW na extração de conteúdos nas páginas desta vertical, conforme apresentado na Figura 26. Durante os testes realizados, não é possível verificar como o trabalho WebKE se comporta utilizando o N-SWDE, pois o código disponibilizado pelo autor não possui um meio de utilizar novos *datasets* na inferência do modelo. O autor dispo-

nibilizou em seu repositório público apenas um modelo pré-configurado com todos os dados das verticais *movie*, *nbaplayer* e *university* já preparados para a execução do modelo, impossibilitando de utilizar outras verticais ou novos *datasets*.

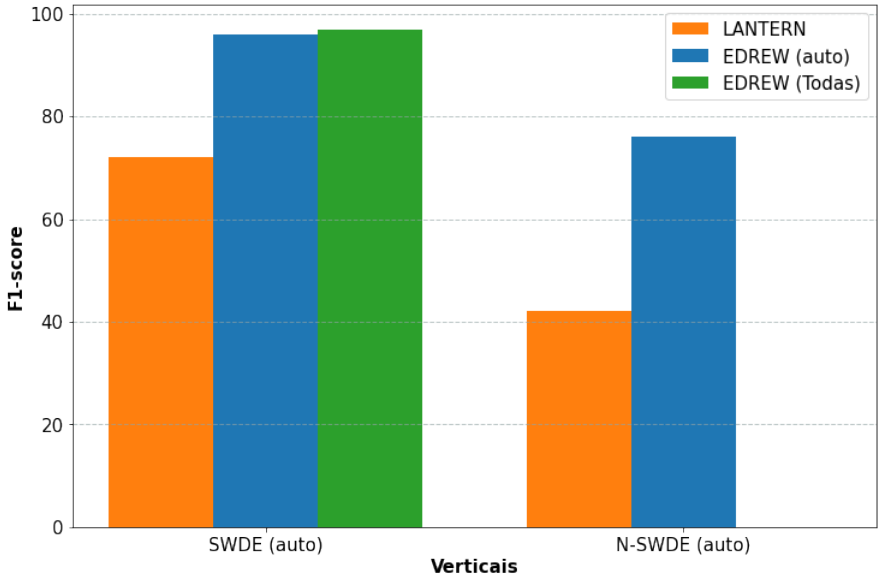

Figura 26 – Comparação entre o trabalho LANTERN e o EDREW utilizando todas as verticais (Todas) e a vertical *auto*, utilizando a métrica *F1-score*, nos *datasets* SWDE e N-SWDE.

É importante ressaltar que o modelo não foi retreinado com as páginas do *dataset* N-SWDE para os novos testes realizados. Isso significa que o modelo conseguiu abstrair as informações relevantes das páginas do SWDE e identificar as mesmas no N-SWDE, sendo indiferente quanto à tecnologia utilizada pelas páginas.

6 CONCLUSÃO

Neste trabalho, o objetivo foi verificar a viabilidade de extração de conteúdos de páginas Web que utilizam de novas abordagens de apresentação dinâmica de conteúdo, como por exemplo SPA. Para isso, foi desenvolvida uma nova abordagem de extração de dados da Web, chamado de EDREW, que tem como objetivo criar uma representação textual dos elementos da página para simbolizar o contexto do elemento dentro da página Web, sem a necessidade de utilizar de renderização. Foram utilizados algoritmos de extração e montagem das *tags* de uma página Web em representações textuais das mesmas e empregado um modelo bidirecional de processamento de linguagem natural para a tarefa de extração dos conteúdos relevantes das páginas Web.

Para avaliar a proposta, foi utilizado o *dataset* público SWDE, com 80 sites distintos, contendo diversas páginas já rotuladas de 8 assuntos diversos. Também foi criado um novo *dataset* chamado de *New Structured Web Data Extraction* (N-SWDE), com as versões atualizadas das páginas do SWDE e com a mesma estrutura de utilização, facilitando que trabalhos possam utilizar ambos *datasets* sem realizar alterações em suas implementações. Para validar o trabalho, foram realizadas comparações com os trabalhos WebKE (XIE *et al.*, 2021) e LANTERN (ZHOU *et al.*, 2022), através da métrica *F-1 score*, utilizando os *datasets* citados acima. Com a nova abordagem, foi possível superar os trabalhos citados em três das quarto verticais testadas utilizando o *dataset* SWDE original e extrair 80,9% a mais de conteúdos das páginas da vertical *auto* em comparação ao LANTERN, utilizando o novo *dataset* N-SWDE.

Com os resultados apresentados, conclui-se que é possível realizar a tarefa de extração dos conteúdos relevantes das páginas Web que utilizam de abordagens tradicionais e SPAs, sem a utilização de renderização das páginas Web, através da utilização de um modelo adaptável a diversos domínios e tecnologias para realizar a tarefa.

A presente dissertação foi publicada em formato de artigo no XXXVII Simpósio Brasileiro de Bancos de Dados na sessão de *Workshop* de Teses e Dissertações em Bancos de Dados (NUNES, Marcelo C; DORNELES, Carina F, 2022) e no *29th Brazilian Symposium on Multimedia and the Web* (NUNES, Marcelo C.; DORNELES, Carina F., 2023).

6.1 TRABALHOS FUTUROS

Mesmo com a conclusão de que é possível realizar a tarefa de extração de dados através da abordagem apresentada, ainda há espaço para melhorias no processo de extração dos dados. Como apresentado na Seção 4.6, há casos em que mesmo que o modelo retorne todos os conteúdos da página Web, também é retornado uma pequena

quantidade de ruído junto. Para melhorar isto, alguns possíveis trabalhos futuros podem ser realizados:

- Criar novas verticais para o N-SWDE a fim de aumentar a diversidade de *layouts* de páginas Web, principalmente em páginas com *layout* similar a vertical *nbaplayers*.
- Adicionar análise de arquivos Javascript e CSS da página, a fim de inferir os posicionamentos dos elementos na página e seus estilos para além do que é apresentado apenas no HTML, incrementando a representação textual do EDREW.
- Investigar o uso de *Large Language Model* (LLM) para realizar a interpretação do reTPS do EDREW, buscando melhorar a inferência dos conteúdos.

REFERÊNCIAS

ANDERSON, Neil; HONG, Jun. Visually Extracting Data Records from the Deep Web. *In*: PROCEEDINGS of the 22nd International Conference on World Wide Web. Rio de Janeiro, Brazil: Association for Computing Machinery, 2013. (WWW '13 Companion), p. 1233–1238.

ANDERSON, Neil; HONG, Jun. Visually Extracting Data Records from the Deep Web. *In*: (WWW '13 Companion), p. 1233–1238.

BEAIRD, Jason; WALKER, Alex; GEORGE, James. **The principles of beautiful web design**. [*S.l.*]: SitePoint Pty Ltd, 2020.

CAI, Deng; YU, Shipeng; WEN, Ji-Rong; MA, Wei-Ying. Extracting Content Structure for Web Pages Based on Visual Representation. *In*: PROCEEDINGS of the 5th Asia-Pacific Web Conference on Web Technologies and Applications. Xian, China: Springer-Verlag, 2003. (APWeb'03), p. 406–417.

CONSORTIUM, WWW *et al.* Xml path language (xpath) version 1.0. **http://www. w3. org/TR/1999/REC-xpath-19991116/**, 1999.

CRESTAN, Eric; PANTEL, Patrick. Web-Scale Table Census and Classification. *In*: PROCEEDINGS of the Fourth ACM International Conference on Web Search and Data Mining. Hong Kong, China: Association for Computing Machinery, 2011. (WSDM '11), p. 545–554.

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

DOWNEY, Doug; ETZIONI, Oren; SODERLAND, Stephen; WELD, Daniel S. Learning text patterns for web information extraction and assessment. *In*: AAAI-04 workshop on adaptive text extraction and mining. [*S.l.*: *s.n.*], 2004. p. 50–55.

ESTER, Martin; KRIEGEL, Hans-Peter; SANDER, Jörg; XU, Xiaowei. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *In*: PROCEEDINGS of the Second International Conference on Knowledge Discovery and Data Mining. Portland, Oregon: AAAI Press, 1996. (KDD'96), p. 226–231.

FACEBOOK. **React**. 2024. Disponível em: `https://react.dev/`. (accessed: 25.04.2024).

FANG, Yixiang; XIE, Xiaoqin; ZHANG, Xiaofeng; CHENG, Reynold; ZHANG, Zhiqiang. STEM: a suffix tree-based method for web data records extraction. **Knowledge and Information Systems**, Springer, v. 55, p. 305–331, 2018.

FAYZRAKHMANOV, Ruslan R.; SALLINGER, Emanuel; SPENCER, Ben; FURCHE, Tim; GOTTLOB, Georg. Browserless Web Data Extraction: Challenges and Opportunities. *In*: PROCEEDINGS of the 2018 World Wide Web Conference. Lyon, France: International World Wide Web Conferences Steering Committee, 2018. (WWW '18), p. 1095–1104.

GOOGLE. **Angular Core**. 2024. Disponível em: `https://angular.io/api/core/`. (accessed: 25.04.2024).

GOOGLE. **AngularJS**. 2021. Disponível em: `https://angularjs.org/`. (accessed: 25.04.2024).

GOOGLE. **What are Scopes?** 2020. Disponível em: `https://docs.angularjs.org/guide/scope/`. (accessed: 25.04.2024).

GUO, Jinsong; CRESCENZI, Valter; FURCHE, Tim; GRASSO, Giovanni; GOTTLOB, Georg. RED: Redundancy-Driven Data Extraction from Result Pages? *In*: THE World Wide Web Conference. San Francisco, CA, USA: Association for Computing Machinery, 2019. (WWW '19), p. 605–615.

HAO, Qiang; CAI, Rui; PANG, Yanwei; ZHANG, Lei. From one tree to a forest: a unified solution for structured web data extraction. *In*: PROCEEDINGS of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. [*S.l.*: *s.n.*], 2011. p. 775–784.

HE, Hao; CHEN, Lulu; GUO, Wenpu. Research on web application vulnerability scanning system based on fingerprint feature. *In*: ATLANTIS PRESS. 2017 International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2017). [*S.l.*: *s.n.*], 2017. p. 150–155.

HOPPE, Pedro Henrique Brunoro; SOUZA, Vítor Estêvão Silva. Support for Single Page Application Frameworks on FrameWeb. *In*: PROCEEDINGS of the 29th Brazilian Symposium on Multimedia and the Web. , Ribeirão Preto, Brazil, Association for Computing Machinery, 2023. (WebMedia '23), p. 260–268.

ILIĆ, Ivana; STEFANOVIĆ, Milena; SADIKOVIĆ, Dušan. Mathematical determination in nature: The golden ratio. **Acta Medica Medianae**, v. 57, n. 3, p. 124–129, 2018.

KANG, Yue; CAI, Zhao; TAN, Chee-Wee; HUANG, Qian; LIU, Hefu. Natural language processing (NLP) in management research: A literature review. **Journal of Management Analytics**, Taylor & Francis, v. 7, n. 2, p. 139–172, 2020.

LIU, Bing; GROSSMAN, Robert; ZHAI, Yanhong. Mining Data Records in Web Pages. *In*: PROCEEDINGS of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, D.C.: Association for Computing Machinery, 2003. (KDD '03), p. 601–606.

LIU, Wei; MENG, Xiaofeng; MENG, Weiyi. ViDE: A Vision-Based Approach for Deep Web Data Extraction. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 3, p. 447–460, 2010.

LIU, Wei; MENG, Xiaofeng; MENG, Weiyi. ViDE: A Vision-Based Approach for Deep Web Data Extraction. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 3, p. 447–460, 2010.

MARPLES, Callum Robert; WILLIAMS, Philip Michael. The Golden Ratio in Nature: A Tour Across Length Scales. **Symmetry**, MDPI, v. 14, n. 10, p. 2059, 2022.

MEHTA, Bhavdeep; NARVEKAR, Meera. DOM tree based approach for Web content extraction. *In*: 2015 International Conference on Communication, Information & Computing Technology (ICCICT). [*S.l.*: *s.n.*], 2015. p. 1–6.

MIAO, Gengxin; TATEMURA, Junichi; HSIUNG, Wang-Pin; SAWIRES, Arsany; MOSER, Louise E. Extracting Data Records from the Web Using Tag Path Clustering. *In*: PROCEEDINGS of the 18th International Conference on World Wide Web. Madrid, Spain: ACM, 2009. (WWW '09), p. 981–990.

MIKOLOV, Tomas; LE, Quoc V; SUTSKEVER, Ilya. Exploiting similarities among languages for machine translation. **arXiv preprint arXiv:1309.4168**, 2013.

MOHAMED, Abdulrehman A; CHERUIYOT, WK; RIMIRU, Richard; ONDAGO, C. Responsive web design inFluid grid concept literature survey. **The International Journal Of Engineering And Science (IJES)**, v. 3, n. 7, 2014.

MOLIN, Eric. **Comparison of Single-Page Application Frameworks: A method of how to compare Single-Page Application frameworks written in JavaScript**. [*S.l.*: *s.n.*], 2016.

NAMOUN, Abdallah. Three column website layout vs. grid website layout: An eye tracking study. *In*: SPRINGER. DESIGN, User Experience, and Usability: Designing Interactions: 7th International Conference, DUXU 2018, Held as Part of HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings, Part II 7. [*S.l.*: *s.n.*], 2018. p. 271–284.

NARWAL, Neetu. Improving web data extraction by noise removal. IET, 2013.

NUNES, Marcelo C; DORNELES, Carina F. NAEWI-Non-rendering Approach to Extract Web Information. *In*: SBC. ANAIS Estendidos do XXXVII Simpósio Brasileiro de Bancos de Dados. [*S.l.*: *s.n.*], 2022. p. 161–167.

NUNES, Marcelo C.; DORNELES, Carina F. EDREW - Enhanced Data Representation for Extraction in Web. *In*: PROCEEDINGS of the 29th Brazilian Symposium on

Multimedia and the Web. , Ribeirão Preto, Brazil, Association for Computing Machinery, 2023. (WebMedia '23), p. 230–237.

PANDARGE, Sangmesh S.; CHAKKARWAR, V. A. Automatic web information extraction and alignment using CTVS technique. *In*: 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA). [*S.l.*: *s.n.*], 2017. v. 2, p. 94–99.

PARK, Kyounghyun; NGUYEN, Minh Chau; WON, Heesun. Web-based collaborative big data analytics on big data as a service platform. *In*: 2015 17th International Conference on Advanced Communication Technology (ICACT). [*S.l.*: *s.n.*], 2015. p. 564–567.

PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher D. Glove: Global vectors for word representation. *In*: PROCEEDINGS of the 2014 conference on empirical methods in natural language processing (EMNLP). [*S.l.*: *s.n.*], 2014. p. 1532–1543.

PETERS, Matthew E.; NEUMANN, Mark; IYYER, Mohit; GARDNER, Matt; CLARK, Christopher; LEE, Kenton; ZETTLEMOYER, Luke. Deep Contextualized Word Representations. *In*: PROCEEDINGS of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). New Orleans, Louisiana: Association for Computational Linguistics, jun. 2018. p. 2227–2237.

PROKOPAKIS, Emmanuel P; VLASTOS, Ioannis M; PICAVET, VA; NOLST TRENITE, G; THOMAS, Regan; CINGI, Cemal; HELLINGS, Peter W. The golden ratio in facial symmetry. **Rhinology**, International Rhinologic Society, v. 51, n. 1, p. 18–21, 2013.

RAKHMAWATI, Nur A; HARITS, Sayekti; HERMANSYAH, Deny; FURQON, Muhammad Ariful. A survey of web technologies used in Indonesia local governments. **Sisfo**, v. 7, n. 3, p. 213–222, 2018.

REAL, Raimundo; VARGAS, Juan M. The Probabilistic Basis of Jaccard's Index of Similarity. **Systematic Biology**, v. 45, n. 3, p. 380–385, set. 1996. ISSN 1063-5157. eprint: `https://academic.oup.com/sysbio/article-pdf/45/3/380/19501760/45-3-380.pdf`.

REIMERS, Nils; GUREVYCH, Iryna. Alternative weighting schemes for elmo embeddings. **arXiv preprint arXiv:1904.02954**, 2019.

REIPS, Lisiane; MUSICANTE, Martin; VARGAS-SOLAR, Genoveva; POZO, Aurora TR; HARA, Carmem S. ENoW-Extrator de Dados de Notícias da Web. *In*: SBC. ANAIS Estendidos do XXXVIII Simpósio Brasileiro de Bancos de Dados. [*S.l.*: *s.n.*], 2023. p. 78–83.

SALAZAR, Jose J; GARLAND, Lean; OCHOA, Jesus; PYRCZ, Michael J. Fair train-test split in machine learning: Mitigating spatial autocorrelation for improved prediction accuracy. **Journal of Petroleum Science and Engineering**, Elsevier, v. 209, p. 109885, 2022.

SCOTT JR, Emmit A. **SPA Design and Architecture: Understanding single-page web applications**. [*S.l.*]: Simon e Schuster, 2015.

SIMON, Kai; LAUSEN, Georg. ViPER: Augmenting Automatic Information Extraction with Visual Perceptions. *In*: PROCEEDINGS of the 14th ACM International Conference on Information and Knowledge Management. Bremen, Germany: Association for Computing Machinery, 2005. (CIKM '05), p. 381–388.

SIU, Christina; CHAPARRO, Barbara S. First look: Examining the horizontal grid layout using eye-tracking. *In*: SAGE PUBLICATIONS SAGE CA: LOS ANGELES, CA, 1. PROCEEDINGS of the Human Factors and Ergonomics Society Annual Meeting. [*S.l.*: *s.n.*], 2014. v. 58, p. 1119–1123.

SU, Weifeng; WANG, Jiying; LOCHOVSKY, Frederick H.; LIU, Yi. Combining Tag and Value Similarity for Data Extraction and Alignment. **IEEE Transactions on Knowledge and Data Engineering**, v. 24, n. 7, p. 1186–1200, 2012.

TANI, Fauzia Yasmeen; FARID, Dewan Md; RAHMAN, Mohammad Zahidur. Ensemble of decision tree classifiers for mining web data streams. **International Journal of Applied Information Systems**, Citeseer, v. 1, n. 2, p. 30–36, 2012.

TSANG, Sik-Ho. **Review — ELMo: Deep Contextualized Word Representations**. [*S.l.*]: Medium, 2022. Disponível em: `https://sh-tsang.medium.com/review-elmo-deep-contextualized-word-representations-8eb1e58cd25c`.

TSENG, Chun-Hsiung. Crowd aided Web search. *In*: 2014 6th International Conference on Knowledge and Smart Technology (KST). [*S.l.*: *s.n.*], 2014. p. 1–6.

ULČAR, Matej; ROBNIK-ŠIKONJA, Marko. High quality ELMo embeddings for seven less-resourced languages. **arXiv preprint arXiv:1911.10049**, 2019.

VELLOSO, Roberto Panerai *et al.* Algoritmo não supervisionado para segmentação e remoção de ruído de páginas web utilizando tag paths, 2014.

VELLOSO, Roberto Panerai *et al.* Optimized record extraction from web pages using signal processing and machine learning, 2020.

VELLOSO, Roberto Panerai; DORNELES, Carina F. Automatic web page segmentation and noise removal for structured extraction using tag path sequences. **Journal of Information and Data Management**, v. 4, n. 3, p. 173–173, 2013.

VELLOSO, Roberto Panerai; DORNELES, Carina F. Optimized Extraction of Records from the Web Using Signal Processing and Machine Learning. *In*: SBBD. [*S.l.*: *s.n.*], 2020. p. 109–120.

W3C. **Tableless layout HOWTO**. 2002. Disponível em: `https://www.w3.org/2002/03/csslayout-howto`. (accessed: 25.04.2024).

W3C. **Web Content Accessibility Guidelines 1.0**. 2021. Disponível em: `https://www.w3.org/TR/WAI-WEBCONTENT/`. (accessed: 25.04.2024).

WAI, Fok Kar; YONG, Lim Wee; THING, Vrizlynn L. L.; POMPONIU, Victor. CMDR: Classifying nodes for mining data records with different HTML structures. *In*: TENCON 2017 - 2017 IEEE Region 10 Conference. [*S.l.*: *s.n.*], 2017. p. 1862–1862.

WANG, Jiying; LOCHOVSKY, Fred H. Data extraction and label assignment for web databases. *In*: PROCEEDINGS of the 12th international conference on World Wide Web. [*S.l.*: *s.n.*], 2003. p. 187–196.

WEISSTEIN, Eric W. Prime spiral. **https://mathworld. wolfram. com**/, Wolfram Research, Inc., 2002.

WENINGER, Tim; HSU, William H. Text Extraction from the Web via Text-to-Tag Ratio. *In*: 2008 19th International Workshop on Database and Expert Systems Applications. [*S.l.*: *s.n.*], 2008. p. 23–28.

XIE, Chenhao; HUANG, Wenhao; LIANG, Jiaqing; HUANG, Chengsong; XIAO, Yanghua. WebKE: Knowledge Extraction from Semi-Structured Web with Pre-Trained Markup Language Model. *In*: PROCEEDINGS of the 30th ACM International Conference on Information & Knowledge Management. New York, NY, USA: Association for Computing Machinery, 2021. p. 2211–2220. ISBN 9781450384469.

YANG, Yudong; ZHANG, HongJiang. HTML page analysis based on visual cues. *In*: PROCEEDINGS of Sixth International Conference on Document Analysis and Recognition. [*S.l.*: *s.n.*], 2001. p. 859–864.

ZHAI, Yanhong; LIU, Bing. Web Data Extraction Based on Partial Tree Alignment. *In*: PROCEEDINGS of the 14th International Conference on World Wide Web. Chiba, Japan: Association for Computing Machinery, 2005. (WWW '05), p. 76–85.

ZHOU, Yichao; SHENG, Ying; VO, Nguyen; EDMONDS, Nick; TATA, Sandeep. Learning Transferable Node Representations for Attribute Extraction from Web Documents. *In*: PROCEEDINGS of the Fifteenth ACM International Conference on Web Search and Data Mining. [*S.l.*: *s.n.*], 2022. p. 1479–1487.