UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

EDUARDO HENRIQUE RIZO

CLASSIFICAÇÃO AUTOMÁTICA DE WIDGETS E SEUS SUBCOMPONENTES POR MEIO DE UM PIPELINE DE APRENDIZADO DE MÁQUINA ATUANDO EM REGISTROS DE MUTAÇÕES DO DOM

DISSERTAÇÃO

CORNÉLIO PROCÓPIO

2019

EDUARDO HENRIQUE RIZO

CLASSIFICAÇÃO AUTOMÁTICA DE WIDGETS E SEUS SUBCOMPONENTES POR MEIO DE UM PIPELINE DE APRENDIZADO DE MÁQUINA ATUANDO EM REGISTROS DE MUTAÇÕES DO DOM

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Mestre em Informática" – Área de Concentração: Engenharia de Software.

Orientador: Prof. Dr. Willian Massami Watanabe

CORNÉLIO PROCÓPIO

2019

Dados Internacionais de Catalogação na Publicação

R627 Rizo, Eduardo Henrique

Classificação automática de widgets e seus componentes por meio de um pipeline de aprendizado de máquina atuando em registros de mutações do DOM / Eduardo Henrique Rizo. – 2019.
69 f. : il. color. ; 31 cm.

Orientador: Willian Massami Watanabe.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. Cornélio Procópio, 2019.
Bibliografia: p. 64-69.

1. Aprendizado do computador. 2. Classificação. 3. Projeto de acessibilidade. 4. Informática – Dissertações. I. Watanabe, Willian Massami, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD (22. ed.) 004

Biblioteca da UTFPR - Câmpus Cornélio Procópio
Bibliotecário/Documentalista responsável:
Romeu Righetti de Araujo – CRB-9/1676

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Programa de Pós-Graduação em Informática

Título da Dissertação Nº 63:

"CLASSIFICAÇÃO AUTOMÁTICA DE WIDGETS E SEUS SUBCOMPONENTES POR MEIO DE UM PIPELINE DE APRENDIZADO DE MÁQUINA ATUANDO EM REGISTROS DE MUTAÇÕES DO DOM".

por

Eduardo Henrique Rizo

Orientador: **Prof. Dr. Willian Massami Watanabe**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM INFORMÁTICA – Área de Concentração: Computação Aplicada, pelo Programa de Pós-Graduação em Informática – PPGI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 14h30 do dia 07 de agosto de 2019. O trabalho foi _____ pela Banca Examinadora, composta pelos professores:

_____
Prof. Dr. Willian Massami Watanabe
(Presidente – UTFPR-CP)

_____
Prof. Dr. André Pimenta Freire
(UFLA)
Participação à distância via _____

_____
Prof. Dr. Cléber Gimenez Corrêa
(UTFPR-CP)

Visto da coordenação:

_____
Danilo Sipoli Sanches
Coordenador do Programa de Pós-Graduação em Informática
UTFPR Câmpus Cornélio Procópio

A Folha de Aprovação assinada encontra-se na Coordenação do Programa.

Dedico este trabalho a minha esposa Cristiane pelo seu incentivo, companheirismo e amor, a minha filha Ana Clara pela doçura e compreensão nos momentos em que não pude estar ao seu lado e aos meus pais pelo apoio incondicional e por sempre torcerem por mim. Dedico também a Deus, que proveu o caminho e as condições necessárias para que essa longa trajetória pudesse ser cumprida.

AGRADECIMENTOS

Agradeço primeiramente ao Prof. Dr. Willian Massami Watanabe, por sua paciência, generosidade e objetividade em nossos diversos momentos de orientação. Muito obrigado!

Muito obrigado aos meus amigos e em especial ao André, Marco, Glauco, Alex, Isique e Anderson pelos diversos momentos que compartilhamos, tenham sido eles alegres, tristes ou mesmo tensos. Valeu pessoal.

Agradeço a Universidade do Oeste Paulista, por me proporcionar flexibilidade de horários durante o período de viagens a Cornélio Procópio e também por prover a infraestrutura de servidores e serviços necessários para o desenvolvimento e execução do estudo experimental realizado para esta dissertação.

Por fim, agradeço a todos os demais professores do programa de mestrado que dedicaram parte do seu tempo para compartilhar conhecimento e me ajudar a chegar até aqui.

RESUMO

RIZO, Eduardo Henrique. CLASSIFICAÇÃO AUTOMÁTICA DE WIDGETS E SEUS SUBCOMPONENTES POR MEIO DE UM PIPELINE DE APRENDIZADO DE MÁQUINA ATUANDO EM REGISTROS DE MUTAÇÕES DO DOM. 69 f. Dissertação – Programa de Pós-Graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

Com o advento da Web 2.0 e do movimento Asynchronous JavaScript and XML (AJAX), os desenvolvedores de sites web intensificaram o uso de mecanismos de interação sofisticados, denominados de *widgets*, para composição da interface de usuário das Aplicações Ricas de Internet (RIAs). Apesar desse fato melhorar a usabilidade e navegabilidade dos sites web, muitos *widgets* são atualmente implementados sem a devida preocupação com o uso das padronizações de acessibilidade especificadas pela *Accessible Rich Internet Applications* (ARIA), o que pode torna-los não acessíveis para pessoas com deficiência. Não obstante à padronização ARIA, as ferramentas atuais de desenvolvimento carecem de recursos de apoio ao desenvolvedor no que tange à disponibilização de funções automatizadas para detecção de não conformidades dos *widgets* para com as regras de acessibilidade da ARIA, o que também contribui para o seu pouco uso. Portanto, esta dissertação de mestrado apresenta uma abordagem para classificar automaticamente *widgets* do tipo *dropdown menu*, *suggestion box* e seus respectivos subcomponentes, por meio de um *pipeline* de aprendizado de máquina que analisa as alterações que ocorrem na estrutura *Document Object Model* (DOM) das páginas web. A classificação de *widgets* e dos seus respectivos subcomponentes é uma etapa essencial para a avaliação automática de conformidade dos mesmos em relação à especificação ARIA, bem como, para criação de mecanismos para adaptação automática da codificação HTML dos *widgets* para mitigar problemas de acessibilidade, assim, contribuindo com o processo de Engenharia de Software de RIAs e mantendo-as em consonância às especificações da ARIA. Para validação, foi conduzido um estudo experimental com 34 dos 50 sites web mais acessados nos EUA, para avaliar a efetividade do *pipeline* de aprendizado de máquina proposto. Os resultados fornecem evidências de que a abordagem proposta é capaz de classificar *widgets* do tipo *dropdown menu*, *suggestion box* e também os seus subcomponentes, pois obteve *F-measure* médio, que é um tipo de acurácia em relação a sensibilidade dos classificadores de aprendizado de máquina, de 0,967 e 0,894 respectivamente. Os resultados também sugerem que o desenvolvimento de artefatos de software que identifiquem automaticamente *widgets* e seus subcomponentes, podem prover subsídios para ferramentas automáticas de avaliação de acessibilidade conforme as regras da ARIA, bem como para ferramentas de adaptação automática de código HTML para acessibilidade, contribuindo para o processo de engenharia web de aplicações acessíveis.

Palavras-chave: Aprendizado de Máquina, Classificação de *Widgets*, Acessibilidade Web, ARIA, Registros de Mutações, Processo de Desenvolvimento

ABSTRACT

RIZO, Eduardo Henrique. AUTOMATIC CLASSIFICATION OF WIDGETS AND THEIR SUBCOMPONENTS BASED ON A MACHINE LEARNING PIPELINE ACTING ON DOM MUTATION RECORDS. 69 f. Dissertação – Programa de Pós-Graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

Since the emergence of Web 2.0 advent and the Asynchronous JavaScript and XML (AJAX) movement, web site developers have stepped up the use of sophisticated interaction mechanisms, called widgets, to design Rich Internet Applications (RIA) user interfaces. Despite the fact improves the usability and navigability of web sites, many widgets are currently implemented without the accessibility design solutions standardized in the Accessible Rich Internet Applications (ARIA) specification and hence they may be not accessible to disabled people. Notwithstanding the ARIA standardization, current development tools lack support in providing automated functions to detect nonconformities of widgets accessibility to the ARIA rules, which also contribute to its little use. Therefore, this master's dissertation presents an approach to automatically classifying widgets type of dropdown menu, suggestion box and their respective subcomponents, through a machine learning pipeline that analyzes the changes that occur in the Document Object Model (DOM) structure of the web pages. Classifying widgets and their subcomponents is an essential step for automatic evaluation of ARIA conformance and HTML code adaptation to mitigate accessibility issues, contributing to the Software Engineering process of RIAs and keeping them in line with ARIA specifications. The proposal validation was performed by an experimental study with 34 of the 50 most visited web sites in the USA to evaluate the effectiveness of the proposed machine learning pipeline. The results provide evidence that the proposed approach is able to classify widgets with dropdown menu, suggestion box and their subcomponents with F-measure metric, that is a kind of accuracy regarding the sensitivity of machine learning classifiers, about 0.967 and 0.894 respectively. The results also suggest that the development of software artifacts that automatically identify widgets and their subcomponents, can provide subsidies for automatic accessibility evaluation tools in conformance with ARIA rules, as well as tools for automatic HTML code adaptation for accessibility, contributing to the process of web engineering accessible applications.

Keywords: Machine Learning, Widgets Classification, Web Accessibility, ARIA, Mutation Records, Process Development

LISTA DE FIGURAS

FIGURA 1 – Exemplos de *widgets* em sites web 10
FIGURA 2 – Arquitetura das RIAs 22
FIGURA 3 – Exemplo de código HTML com recursos da ARIA 25
FIGURA 4 – Visão simplificada da abordagem 33
FIGURA 5 – Ferramenta para classificação manual dos registros de mutação do DOM . 35
FIGURA 6 – Classificação manual dos subcomponentes dos *widgets* 36
FIGURA 7 – Organização dos dados das mutações do DOM 36
FIGURA 8 – *Pipeline* de aprendizado de máquina 38
FIGURA 9 – Exemplo de característica textual para auxiliar no processo de classificação manual dos *widgets* 41
FIGURA 10 – Etapas do estudo experimental 42
FIGURA 11 – Método de geração dos conjuntos de dados 47
FIGURA 12 – *10-fold cross-validation*: *Widgets* e Subcomponentes - TRAINING-DATASET 48
FIGURA 13 – Modelo de predição: *Widgets* e Subcomponentes - TEST-DATASET 49
FIGURA 14 – Organização do código HTML de um *widget* e dos seus subcomponentes . 51

LISTA DE TABELAS

TABELA 1 – Sites selecionados para coleta de amostras 43
TABELA 2 – Lista de sites web usados para compor o grupo A – coleta de amostras para treinamento (*TRAINING-DATASET*) 45
TABELA 3 – Lista de sites web usados para compor o grupo B – coleta de amostras para teste (TEST-DATASET) 46
TABELA 4 – Resumo dos conjuntos de dados dos *widgets*: Treinamento e teste dos algoritmos do primeiro classificador do *pipeline* de aprendizado de máquina 52
TABELA 5 – Resumo dos conjuntos de dados dos subcomponentes dos *widgets*: Treinamento e teste dos algoritmos do segundo classificador do *pipeline* de aprendizado de máquina 52
TABELA 6 – Resultados do procedimento *10 fold cross-validation* para o conjunto de dados de treinamento dos *widgets* (*TRAINING-DATASET*) - Alvo: *Widgets* do tipo *DROPDOWN MENU* 54
TABELA 7 – Resultados do procedimento *10 fold cross-validation* para o conjunto de dados de treinamento dos *widgets* (*TRAINING-DATASET*) - Alvo: Widgets do tipo *SUGGESTION BOX* 55
TABELA 8 – Resultados do procedimento *10 fold cross-validation* para o conjunto de dados de treinamento dos *widgets* (*TRAINING-DATASET*) - Alvo: *Widgets* do tipo OUTROS *WIDGETS* 55
TABELA 9 – Resultados do procedimento 10 fold cross-validation para o conjunto de dados de treinamento dos subcomponentes dos *widgets* (TRAINING-DROPDOWN-SUBCOMPONENTS) - Alvo: É SUBCOMPONENTE de *widgets* do tipo *Dropdown menu* 55
TABELA 10 – Resultados do procedimento 10 fold cross-validation para o conjunto de dados de treinamento dos subcomponentes dos *widgets* (TRAINING-DROPDOWN-SUBCOMPONENTS) - Alvo: NÃO É SUBCOMPONENTE de *widgets* do tipo *Dropdown menu* 56
TABELA 11 – Resultados do procedimento 10 fold cross-validation para o conjunto de dados de treinamento dos subcomponentes dos *widgets* (TRAINING-SUGGESTION-SUBCOMPONENTS) - Alvo: É SUBCOMPONENTE de *widgets* do tipo *Suggestion box* 56
TABELA 12 – Resultados do procedimento 10 fold cross-validation para o conjunto de dados de treinamento dos subcomponentes dos *widgets* (TRAINING-SUGGESTION-SUBCOMPONENTS) - Alvo: NÃO É SUBCOMPONENTE de *widgets* do tipo *Suggestion box* 56
TABELA 13 – Predição: Matriz de confusão para o conjunto de teste dos *widgets* (TEST-DATASET) - Algoritmo: Decision Tree 57
TABELA 14 – Predição: Matriz de confusão para o conjunto de teste dos *widgets* (TEST-DATASET) - Algoritmo: Random Forest 57
TABELA 15 – Predição: Matriz de confusão para o conjunto de teste dos *widgets* (TEST-DATASET) - Algoritmo: SVM 57

TABELA 16 – Predição: Matriz de confusão para o conjunto de teste dos subcomponentes dos *widgets* (TEST-DROPDOWN-SUBCOMPONENTS) - Algoritmo: Random Forest - Alvo: SUBCOMPONENTES de *widgets* do tipo *Dropdown menu* 58
TABELA 17 – Predição: Matriz de confusão para o conjunto de teste dos subcomponentes dos *widgets* (TEST-SUGGESTION-SUBCOMPONENTS) - Algoritmo: Random Forest - Alvo: SUBCOMPONENTES de *widgets* do tipo *Suggestion box* 58

LISTA DE SIGLAS

RIAs	Rich Internet Applications
ARIA	Accessible Rich Internet Applications
DOM	Document Object Model
UI	User Interface
XML	Extensible Markup Language
AJAX	Asynchronous JavaScript and XML
CSS	Cascating Style Sheets
RIA	Rich Internet Applications
W3C	World Wide Web Consortium
HTML	Hyper Text Markup Language
WCAG	Web Content Accessibility Guidelines
WAI	Web Accessibility Initiative
ONU	Organização das Nações Unidas
ABNT	Associação Brasileira de Normas Técnicas
NBR	Norma Brasileira
AG-WG	Accessibility Guidelines Working Group
APA	Accessible Platform Architectures
EOWG	Education and Outreach Working Group
WAI-IG	Web Accessibility Initiative Interest Group

SUMÁRIO

1 INTRODUÇÃO 9
1.1 MOTIVAÇÃO 11
1.2 OBJETIVOS 12
1.3 ORGANIZAÇÃO DO TEXTO 13
2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 ACESSIBILIDADE E O CONCEITO DE ACESSIBILIDADE NA WEB 14
2.2 MODELO DA WAI 16
2.2.1 Diretrizes de Acessibilidade para Conteúdo Web (WCAG) 17
2.2.2 Acessibilidade para Aplicações Ricas de Internet (ARIA) 21
2.3 TRABALHOS RELACIONADOS 24
2.3.1 Avaliação de Conformidade das RIAs com as Regras da ARIA 26
2.3.2 Classificação Automática de *Widgets* 27
2.3.3 Adaptação Automática do código HTML para Requisitos de Acessibilidade 29
2.4 CONSIDERAÇÕES FINAIS 30
3 *PIPELINE* DE CLASSIFICAÇÃO DE *WIDGETS* E DOS SEUS SUBCOMPONENTES 32
3.1 ESTRATÉGIA DE CAPTURA DE *LOGS* 34
3.1.1 *Logger* 34
3.1.2 Módulo Receptor e Classificação das Amostras 35
3.2 PIPELINE DE CLASSIFICAÇÃO DE *WIDGETS* E SUBCOMPONENTES 36
3.3 CONSIDERAÇÕES FINAIS 41
4 AVALIAÇÃO DA ABORDAGEM PROPOSTA 42
4.1 DEFINIÇÃO E EXECUÇÃO DO EXPERIMENTO 42
4.2 SOBRE A AQUISIÇÃO DAS AMOSTRAS 49
4.3 ANÁLISE DOS RESULTADOS 53
4.4 DISCUSSÃO DOS RESULTADOS 58
4.5 CONSIDERAÇÕES FINAIS 59
5 CONCLUSÃO 61
5.1 LIMITAÇÕES E TRABALHOS FUTUROS 62
5.2 DIVULGAÇÃO DOS RESULTADOS 63
REFERÊNCIAS 64

1 INTRODUÇÃO

O ambiente web é um espaço importante para a disponibilização de conteúdos, serviços e aplicações (programas de computador) para os mais diversos propósitos, como por exemplo, para a educação, para a formação profissional, para o trabalho, para a informação, para a cultura, para as comunicações, para o comércio, para os negócios, para a saúde, para os serviços públicos e outros. Uma das premissas da Web é a possibilidade de uso amplo e acessível por indivíduos com diferentes objetivos, níveis de conhecimento e características pessoais. Assim, a acessibilidade passa a não significar acesso a uma coisa só, mas a uma infinidade de aspectos importantes da vida e do cotidiano de cada pessoa (PAGANELLI; PATERNÒ, 2002; VARGAS et al., 2010; W3C Brasil, 2013).

Com o surgimento da Web 2.0 e do "Movimento Ajax", os desenvolvedores vêm intensificando o uso de recursos de *User Interface* (UI) em aplicações web para propiciar mecanismos complexos de interação e efeitos visuais, tornando sua experiência de uso similar ao de uma aplicação *desktop* (GIBSON; SCHWERDTFEGER, 2005). Esse tipo de aplicação web normalmente é caracterizada por atualizações dinâmicas de pequenos trechos das páginas que a compõem e fazem uso de recursos de programação como JavaScript, *Asynchronous JavaScript and XML* (AJAX) e *Cascating Style Sheets* (CSS), para garantir a combinação entre interação e efeitos visuais, o que tende a prover boa usabilidade aos usuários (VELASCO et al., 2008; SCHMIDT et al., 2008; CHEN, 2010). Nesse contexto, as aplicações web que utilizam esse paradigma passaram a ser denominadas como *Rich Internet Applications* (RIA). Nas RIAs foi introduzido o conceito de separação de processamento (FRATERNALI et al., 2010), implementando funcionalidades de interações no lado do cliente da arquitetura web, por meio de componentes de interface e comunicação assíncrona com o servidor, disponibilizados pelo objeto XMLHTTPRequest do JavaScript (MUNSON; PIMENTEL, 2008). Segundo SANDHYA; DEVI (2011), as RIAs também aprimoraram a disponibilização tradicional de conteúdos e serviços apenas baseados em textos, imagens e *hyperlinks*, passando a fazer uso de componentes visuais aprimorados de interface conhecidos como *widgets*.

Widgets são componentes visuais de software que podem ser usados para compor a

interface de uma aplicação web e são baseados em padrões de desenvolvimento definidos pelo *World Wide Web Consortium* (W3C) (W3C, 2012). CHEN et al. (2013) definem o termo *widget* como um "objeto de interface discreto inserido em uma página web com o qual o usuário pode interagir". Exemplos de *widgets* incluem componentes de interface de usuário disponíveis em *toolkits* JavaScript, como por exemplo, caixas de diálogo, *tooltips*, menus suspensos, seletores de datas, caixas de sugestões, entre outros (Figura 1). Segundo GIBSON (2007), os *widgets* geram atualizações dinâmicas na estrutura DOM de uma página web e normalmente requer percepção visual dos usuários para que suas mudanças possam ser notadas. Dessa forma, apesar do modelo sofisticado de interação dos *widgets* aumentar a usabilidade dos sites web, eles podem criar barreiras para Tecnologias Assistivas ao interpretar o conteúdo das páginas e torná-las inacessíveis para usuários com deficiência. CHEN (2010) considera que a acessibilidade das RIAs depende muito de como os *widgets* foram desenvolvidos.

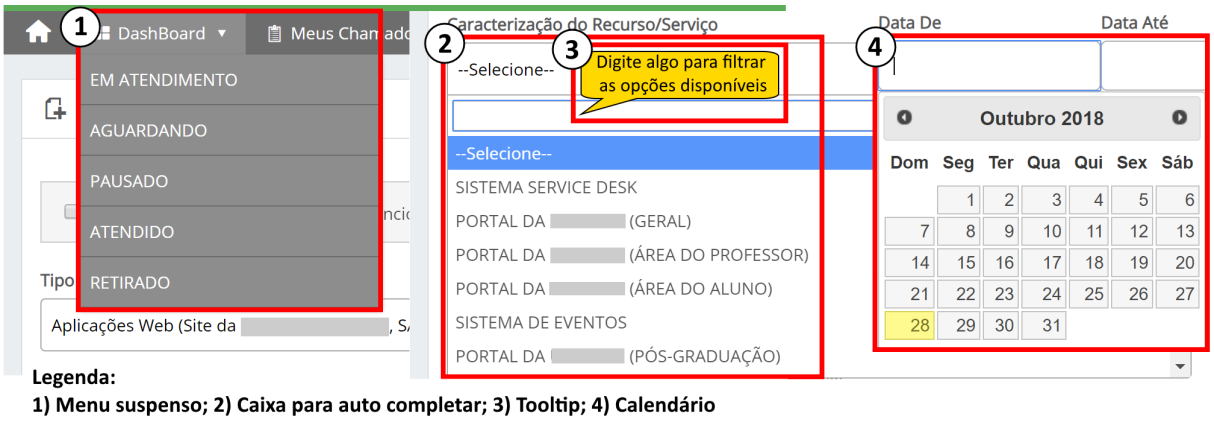

Figura 1: Exemplos de *widgets* em sites web

Além disso, os desenvolvedores web geralmente fazem uso de *widgets* gráficos sofisticados que podem produzir diversas alterações na estrutura DOM da página web. O uso desses tipos de *widgets* também requerem que os usuários tenham percepção visual para garantir que eles possam identificar os principais componentes dos *widgets* e interagir com eles (GIRAUD et al., 2011; VELASCO et al., 2008). Como resultado, as pessoas que usam Tecnologias Assistivas para interagir com aplicações web podem enfrentar várias barreiras de acessibilidade, por exemplo, usuários com deficiência visual (BROWN et al., 2012).

Pessoas com deficiência visual fazem uso de leitores de tela para ouvir e navegar sequencialmente pelo conteúdo de uma determinada página web na ordem em que seu código HTML foi especificado. No entanto, as alterações e atualizações na estrutura de uma aplicação web, geradas pelo JavaScript, nem sempre são apresentadas aos usuários interagindo por meio da adoção desse tipo de tecnologia assistiva. Infelizmente, os leitores de tela reconhecem *widgets* que aparecem dinamicamente na tela apenas se os mesmos tiverem sido desenvolvidos

seguindo as especificações contidas na ARIA (MELNYK et al., 2014). Portanto, para disponibilizar aplicações web acessíveis, os desenvolvedores devem seguir as padronizações de acessibilidade estabelecidas na documentação da ARIA (W3C, 2013).

Para lidar com o aumento da complexidade das interfaces de usuário na Web o W3C, por meio do seu grupo de trabalho WAI, elaborou a especificação ARIA. A especificação ARIA provê uma ontologia de regras/estados/propriedade HTML que podem ser aplicadas pelos desenvolvedores web enquanto implementam *widgets* acessíveis (W3C, 2013). Segundo WATANABE; FORTES (2016), a especificação ARIA disponibiliza um conjunto de atributos semânticos a serem incluídos nos elementos HTML que compõem os *widgets* com intuito de melhorar sua acessibilidade. O objetivo desses atributos semânticos é permitir que os desenvolvedores aumentem a acessibilidade das páginas web para conteúdos dinâmicos e componentes da interface do usuário (MELNYK et al., 2014). Assim, as Tecnologias Assistivas podem informar adequadamente aos usuários sobre atualizações e alterações que tenham ocorrido nas páginas web em decorrência da interação com um *widget* (GIBSON, 2007). No entanto, estudos anteriores (FREIRE et al., 2008b, 2008a; WATANABE et al., 2014) constataram que mesmo com a especificação ARIA sendo uma padronização reconhecida pela W3C e estar incluída na especificação HTML5, há pouco uso ou uso incorreto de suas regras por parte dos desenvolvedores web.

1.1 MOTIVAÇÃO

A padronização ARIA alcançou status de Recomendação W3C em 2014 (W3C, 2017a), mas estudos indicam que muitas aplicações web ainda não implementam as soluções para acessibilidade fornecidas por ela (CARVALHO et al., 2016; WATANABE et al., 2014). Como os desenvolvedores nem sempre usam ou conhecem soluções de acessibilidade, torna-se necessária a elaboração de ferramentas e metodologias de avaliação que os auxiliem no uso desse tipo de recurso (WATANABE, 2014).

Para avaliar a conformidade da implementação dos *widgets* com as regras da ARIA, é essencial conhecer o tipo de cada um deles e então verificar se suas regras e semânticas estão ajustadas aos seus respectivos tipos. Por exemplo, verificar se um elemento DIV identificado como uma “aba” é parte de um contêiner de “abas”, ou verificar se um *widget* identificado como um menu suspenso tem as regras corretas da ARIA relativas ao seu tipo. Assim sendo, a assertividade na classificação dos *widgets* é importante para verificar a conformidade desses elementos com as suas respectivas regras para acessibilidade, conforme constam elencadas na página web da W3C/ARIA *Authoring Practices* (W3C, 2017b).

Há ferramentas comerciais e gratuitas que possibilitam avaliar a conformidade de acessibilidade de sites web e dos *widgets* que os compõem de acordo a *Web Content Accessibility Guidelines* (WCAG) e regras da ARIA, como por exemplo, AChecker[1], DaSilva[2], WAVE[3], MAUVE[4] e outras (W3C, 2016). No entanto, as RIAs, diferentemente das aplicações web que não utilizam JavaScript, implementam recursos dinâmicos que alteram a estrutura hierárquica do HTML. Dessa forma, a análise da interface e conteúdo apresentado deve considerar não apenas o conteúdo HTML de uma aplicação web, mas também o seu comportamento implementado utilizando JavaScript e CSS (WATANABE, 2014).

Diante do exposto, a tarefa de classificar automaticamente *widgets* em aplicações web é um componente chave para o desenvolvimento de ferramentas de avaliação automática das regras da ARIA. Para cada tipo de *widget*, diferentes notações da ARIA devem ser empregadas em suas *tags* HTML para torná-los acessíveis. Dada a implementação de um artefato de software que identifique *widgets* e seus subcomponentes, torna-se possível o desenvolvimento de ferramentas de apoio ao desenvolvedor que permitam verificar a conformidade dos *widgets* das RIAs para com as regras da ARIA ou mesmo utilizá-las para promover a adaptação automatizada de suas respectivas codificações HTML com o intuito de torná-las acessíveis.

1.2 OBJETIVOS

O objetivo desta dissertação de mestrado foi o desenvolvimento de uma abordagem baseada em um *pipeline* de aprendizado de máquina para classificação automática de *widgets* e seus subcomponentes em RIAs. Além disso, a abordagem de classificação proposta visa apoiar o desenvolvimento ou melhoria de ferramentas de avaliação de acessibilidade, para que nelas estejam inseridas as estratégias de avaliação automatizada das regras da ARIA propostas no artigo de DOUSH et al. (2013). Em particular, pretende-se responder à seguinte questão de pesquisa (QP):

QP: *Qual é a efetividade de um pipeline de aprendizado de máquina para classificação automática de widgets e seus subcomponentes com base em conjuntos de dados extraídos das alterações que ocorrem na estrutura DOM durante as interações dos usuários em RIAs?*

Partindo do princípio que as RIAs são tipicamente compostas por diferentes tipos de *widgets*, como por exemplo, *dropdown menus*, *suggestion box sliders*, calendários, *tooltips* e

[1] http://achecker.ca/
[2] http://www.dasilva.org.br/
[3] http://wave.webaim.org/
[4] https://mauve.isti.cnr.it

outros, a *QP* objetiva investigar se as alterações, também conhecidas como mutações, ocorridas na estrutura DOM das RIAs fornecem informações suficientes para a composição de conjuntos de dados que permitam o aprendizado de máquina e por consequência a respectiva predição para classificação automática de *widgets* e dos seus subcomponentes.

O uso de aprendizado de máquina como ferramenta de apoio ao processo de classificação de *widgets* se dá pelo elevado número de mutações na estrutura DOM que potencialmente ocorrem nas páginas web que compõem as RIAs. Há também uma diversidade de implementações que regularmente são encontradas para os mesmos tipos de *widgets*, o que dificulta a criação de algoritmos especializados ou expressões regulares que possam varrer a estrutura DOM das páginas web com intuito de localizar e classificar algum tipo de *widget*.

1.3 ORGANIZAÇÃO DO TEXTO

Esta dissertação de mestrado está organizada da seguinte forma: no **Capítulo 2** são apresentados os principais conceitos sobre acessibilidade na web, as padronizações de acessibilidade propostas pela *Web Accessibility Initiative* (WAI) e também trabalhos relacionados a esse tema. No **Capítulo 3** é apresentado o desenvolvimento de um *pipeline* de aprendizado de máquina para classificação automática dos *widgets* e dos seus subcomponentes. Nele estão descritas a estratégia para captura dos registros de mutação do DOM *(logs)*, o próprio *pipeline* de classificação, a metodologia utilizada e os resultados obtidos. No **Capítulo 4** demonstra-se o procedimento adotado para avaliação da abordagem proposta, a análise e discussão dos resultados e considerações finais. Já no **Capítulo 5**, evidenciam-se as conclusões, as limitações da pesquisa e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentadas a fundamentação teórica sobre acessibilidade em seu âmbito geral, acessibilidade na web, padronizações de acessibilidade especificadas pelo W3C para sites web e, por fim, trabalhos relacionados a classificação automática de *widgets*, avaliação de conformidade das RIAs para com a ARIA e adaptação automática de código HTML para acessibilidade.

2.1 ACESSIBILIDADE E O CONCEITO DE ACESSIBILIDADE NA WEB

Exitem muitas definições para o termo acessibilidade. Elas perpassam desde a questão de mobilidade urbana, de acesso a edifícios e outras instalações até ao uso de recursos computacionais, como por exemplo, o acesso irrestrito aos diferentes sites de serviço ou portais de informação disponíveis na web. Assim, são apresentadas nesta seção, definições gerais sobre o termo para entendimento do contexto das pessoas com deficiência e o que se espera sobre o uso de recursos computacionais pelas mesmas.

O Decreto Federal nº 5.296/2004 (BRASIL, 2004), em seu artigo 8º, I, define o termo acessibilidade como uma condição para utilização, com segurança e autonomia, total ou assistida, dos espaços, mobiliários e equipamentos urbanos, das edificações, dos serviços de transporte e dos dispositivos, sistemas e meios de comunicação e informação, por pessoa com deficiência ou com mobilidade reduzida.

A Convenção Internacional Sobre os Direitos das Pessoas com Deficiência, adotada pela ONU em 30 de março de 2007, em Nova York, e ratificada pelo Decreto Federal nº 6.949 de 25 de agosto de 2009 (BRASIL, 2009), estabelece em seu artigo 9º, item 1:

> A fim de possibilitar às pessoas com deficiência viver com autonomia e participar plenamente de todos os aspectos da vida, os Estados Partes deverão tomar as medidas apropriadas para assegurar-lhes o acesso, em igualdade de oportunidades com as demais pessoas, ao meio físico, ao transporte, à informação e comunicação, inclusive aos sistemas e tecnologias da informação e comunicação, bem como a outros serviços e instalações

> abertos ou propiciados ao público, tanto na zona urbana como na rural.

A Norma Brasileira ABNT NBR 9050:2004 (ABNT, 2015), define em seu item 3.1 que acessibilidade é a possibilidade e condição de alcance, percepção e entendimento para a utilização com segurança e autonomia de edificações, espaço, mobiliário, equipamento urbano e elementos.

No contexto de aplicações web, o W3C Brasil, em sua Cartilha de Acessibilidade na Web (W3C Brasil, 2013), faz um apanhado sobre o termo "acessibilidade" e o define como:

> ... possibilidade e condição de alcance, percepção e entendimento para a utilização, em igualdade de oportunidades, com segurança e autonomia, do meio físico, do transporte, da informação e da comunicação, inclusive dos sistemas e tecnologias de informação e comunicação, bem como de outros serviços e instalações.
> Para as pessoas com deficiência e mobilidade reduzida, a acessibilidade possibilita uma vida independente e com participação plena em todos os seus aspectos; e para todas as pessoas, em diferentes contextos, pode proporcionar maior conforto, facilidade de uso, rapidez, satisfação, segurança e eficiência.

O fundador da Web Tim Berners-Lee afirma que "o poder da Web está na sua universalidade. O acesso por todas as pessoas, não obstante a sua deficiência, é um aspecto essencial" (W3C, 2005a). Dessa forma, a acessibilidade na web deve assegurar condições de igualdade no uso dos sites a quaisquer indivíduos, independentemente de suas capacidades motoras, visuais, auditivas, intelectuais, culturais ou sociais, inclusive sem levar em consideração o ambiente físico, computacional ou mesmo o dispositivo de acesso a ser utilizado (W3C Brasil, 2013).

Portanto, segundo consta nas diretrizes de acessibilidade para conteúdo web (W3C, 1999), o termo "acessibilidade na web" é uma característica de qualidade inerente ao projeto da página web de forma geral, com intuito de garantir que os usuários sejam capazes de acessá-la a partir de diferente situações. Como por exemplo, nas situações em que os usuários (a) sejam incapazes de ver, ouvir, mover ou compreender determinadas formas de disponibilização do conteúdo, (b) possuam dificuldades de leitura e compreensão do texto, (c) sejam incapazes de utilizar o teclado ou o mouse, (d) utilizem navegadores apenas em modo texto, tela com dimensões reduzidas ou conexão com a Internet lenta, (e) estejam com os olhos, ouvidos ou mãos ocupadas em outras atividades, e (f) utilizem versões antigas ou diferentes navegadores, navegador por voz ou diferentes sistemas operacionais.

Diante das situações expostas, nota-se que elas refletem tanto o âmbito dos requisitos tecnológicos necessários para a interação do usuário, quanto as características físicas e

cognitivas do mesmo, fazendo com que a preocupação com acessibilidade na web transite entre ambas. Para os requisitos tecnológicos, os desenvolvedores web devem considerar a implementação de suas aplicações observando seu funcionamento de modo independente às características do hardware e software utilizados para a interação do usuário. Já para as situações relacionadas com as características do usuário, é necessário priorizar a adaptação, redundância e substituição de conteúdo de acordo com as deficiências físicas ou cognitivas que eles possam apresentar (WATANABE, 2014).

2.2 MODELO DA WAI

A *Web Accessibility Initiative* (WAI) é uma organização criada pelo W3C que tem como objetivo definir princípios e regras de projeto e desenvolvimento de sites que sejam acessíveis a pessoas com deficiência (LUCCA et al., 2005). A WAI desenvolve seu trabalho envolvendo diferentes interessados em acessibilidade Web, como por exemplo, a indústria, organizações para deficientes, órgãos governamentais, pesquisadores, entre outros (W3C, 2018a). Para tratar sobre os diferentes assuntos sobre acessibilidade, a WAI se subdivide em diferentes grupos de trabalho e de interesse, sendo eles:

- ***Accessibility Guidelines Working Group (AG-WG)***: A missão do Grupo de Trabalho de Diretrizes de Acessibilidade é desenvolver diretrizes para tornar o conteúdo da Web acessível a pessoas com deficiências e desenvolver e manter materiais de suporte à implementação das Diretrizes de Acessibilidade ao Conteúdo da Web - WCAG.

- ***Accessible Platform Architectures (APA) Working Group***: A missão deste grupo de trabalho é assegurar que as especificações de acessibilidade do W3C de fato forneçam suporte as pessoas com deficiências. O grupo realiza essa tarefa por meio da revisão das especificações disponibilizadas em colaboração com outros Grupos de Trabalho e coordenação de estratégias harmonizadas de acessibilidade dentro do W3C. Exemplos de trabalhos do APA: (a) força tarefa de acessibilidade para indivíduos com deficiência cognitiva ou de aprendizado; (b) força tarefa para identificar *gaps* ou barreiras de acessibilidade nas tecnologias web atuais e futuras. Também é função desta força tarefa a identificação de pesquisadores que possam contribuir para a solução dos problemas identificados; (c) grupo de trabalho sobre requisitos de acessibilidade em mecanismos de pagamento por meio da web, entre outros.

- ***Accessible Rich Internet Applications (ARIA) Working Group***: A missão do Grupo de Trabalho para Aplicações Ricas de Internet Acessíveis (ARIA WG) é desenvolver

tecnologias que melhorem a acessibilidade do conteúdo web para pessoas com deficiências. Isso inclui o desenvolvimento contínuo do conjunto de tecnologias *Accessible Rich Internet Applications* (WAI-ARIA) e outras especificações técnicas.

- ***Education and Outreach Working Group (EOWG)***: A missão do Grupo de Trabalho de Educação e Extensão (EOWG) é desenvolver estratégias e recursos para promover a conscientização, compreensão e implementação da acessibilidade na web e apoiar o trabalho de outros Grupos de Trabalho da Iniciativa de Acessibilidade à Web (WAI).
- ***WAI Interest Group (WAI-IG)***: A missão do Grupo de Interesse WAI (WAI IG) é fornecer um fórum para revisar, discutir e fornecer informações sobre os resultados que estão sendo desenvolvidos pelos Grupos de Trabalho da WAI, incluindo aspectos de acessibilidade de revisões de especificação, tópicos de pesquisa identificados durante revisões de acessibilidade e materiais educacionais.

Diante do exposto sobre a WAI e considerando os objetivos desta dissertação de mestrado, serão tratados nas subseções 2.2.1 e 2.2.2 os objetivos e especificações da WCAG e da ARIA, ambas destinadas à disponibilização de conteúdo web de forma acessível para pessoas com deficiência, portanto, de suma importância para a contextualização geral do tema ao qual este estudo está inserido.

2.2.1 DIRETRIZES DE ACESSIBILIDADE PARA CONTEÚDO WEB (WCAG)

As *Web Content Accessibility Guidelines* (WCAG) são desenvolvidas pelo W3C em cooperação com pessoas e organizações em todo o mundo. Seu objetivo é fornecer um padrão único de acessibilidade de conteúdo web que atenda às necessidades de indivíduos, organizações e governos internacionalmente. Os documentos da WCAG explicam como tornar o conteúdo web mais acessível para pessoas com deficiências e são destinados principalmente aos seguintes grupos (W3C, 2005b):

- desenvolvedores web;
- desenvolvedores de ferramentas de autoria;
- desenvolvedores de ferramentas de verificação de acessibilidade;
- outros indivíduos ou grupos que desejem ou precisem de um padrão para acessibilidade web, inclusive para acessibilidade em dispositivos móveis.

As WCAG são mantidas pela WAI, organização criada pelo W3C, e atualmente dispõem de três versões: a versão 1.0, 2.0 e 2.1 (W3C, 2018b).

A WCAG 1.0 consiste em um documento com diretrizes que discutem problemas de acessibilidade na utilização de tecnologias web e soluções de *design* para tornar o conteúdo web acessível para pessoas com deficiência. Essas diretrizes são destinadas especialmente para os desenvolvedores de conteúdo para web e para desenvolvedores de ferramentas de autoria. O primeiro objetivo dessas diretrizes é o de promover acessibilidade para pessoas com deficiência. No entanto, tomando-as por referência, é possível produzir conteúdo web mais acessível para todas as pessoas, não importando o agente de usuário[1] utilizado ou se estiverem sob alguma condição restritiva, como por exemplo, ambiente com ruído, com baixa iluminação, etc (W3C, 1999).

As diretrizes dispostas na documentação WCAG 1.0 abrangem dois temas gerais:

- **Garantir a transformação harmoniosa:** a transformação harmoniosa das páginas representa a garantia de que o conteúdo existente nas mesmas permaneça acessível, mesmo em situações em que os usuários possuam algum tipo de deficiência física, sensorial, cognitiva ou em situações relacionadas a questões tecnológicas.
- **Tornar o conteúdo compreensível e navegável:** o conteúdo deve ser disponibilizado pelos desenvolvedores de forma simples, clara e com mecanismos de navegação entre as páginas. Prover ferramentas de navegação e informações de orientação nas páginas podem maximizar sua acessibilidade e usabilidade, além disso, os usuários podem perder o contexto da informação quando eles estão acessando a página por meio de um sintetizador de voz, quando possuem visores pequenos ou quando usam resoluções de tela grandes. Sem o uso de informações de orientação, o usuário pode não ser capaz de compreender tabelas, listas ou menus quando os mesmos precisarem ter maiores dimensões para exibir toda a informação neles contida (W3C, 1999).

Para cada diretriz da WCAG 1.0, há pontos de verificação (*checkpoints*) organizados por prioridades e junto a eles técnicas que explicam como proceder para o sucesso das diretrizes. Cada ponto de verificação tem um nível de prioridade baseado em seu impacto para com a acessibilidade. As prioridades dos pontos de verificação são (W3C, 2000):

- **Prioridade 1:** especifica que o conteúdo web **tem** que satisfazer esse ponto de verificação. Caso contrário, um ou mais grupos de usuários acharão impossível acessar

[1] Exemplos: navegador web comum, por voz, dispositivos móveis, navegadores automotivos, etc

informações no documento. A satisfação deste ponto de verificação é um requisito básico para que alguns grupos possam usar o documento;

- **Prioridade 2:** estabelece que o conteúdo web **deve** satisfazer esse ponto de verificação. Caso contrário, um ou mais grupos de usuários terão dificuldade em acessar informações no documento. A satisfação deste ponto de verificação removerá barreiras significativas para o acesso ao documento;

- **Prioridade 3:** indica que o conteúdo web **pode** satisfazer esse ponto de verificação. Caso contrário, um ou mais grupos de usuários poderão experimentar dificuldades para acessar informações no documento. A satisfação deste ponto de verificação melhora o acesso ao conteúdo do documento.

A WCAG 1.0 foi publicada no ano de 1999, quando grande parte das páginas eram implementadas utilizando HTML e, portanto, ela foi desenvolvida com foco nessa tecnologia (KELLY et al., 2007; REID; SNOW-WEAVER, 2008). Sua simplicidade ajudou a aumentar a popularidade do problema de acessibilidade na web e suas diretrizes são reconhecidas como a principal abordagem na elaboração de conteúdo web acessível (KELLY et al., 2007). No entanto, os sites atuais, utilizam uma gama tecnológica não existente à época da WCAG 1.0 e, mesmo com os esforços dos desenvolvedores para com as conformidades de acessibilidade, essas novas tecnologias não estão no contexto das antigas diretrizes, que não permitem sequer o uso de soluções que promovam acessibilidade caso as mesmas não sejam da própria W3C (KELLY et al., 2005; W3C, 1999).

Outro fator limitante no uso da WCAG 1.0 é a subjetividade dos pontos de verificação, que torna difícil uma avaliação total de conformidade com suas diretrizes (REID; SNOW-WEAVER, 2008). Esse fator, inclusive, é determinante na decisão dos Estados Unidos em adotar as diretrizes da *Section 508*[2] e não as da WCAG 1.0 como padrão de acessibilidade web no país (WATANABE, 2014).

Nesse contexto, a WAI desenvolveu a WCAG 2.0 com objetivo de suceder a WCAG 1.0 publicada como Recomendação W3C em maio de 1999. Apesar dos conteúdos poderem estar em conformidade com a WCAG 1.0 ou com a WCAG 2.0 (ou ambas), há uma recomendação do W3C para que os novos conteúdos, ou os que estejam em processo de atualização, utilizem as diretrizes da WCAG 2.0. O W3C recomenda ainda que as políticas de acessibilidade na Web tenham como referência a WCAG 2.0 e não mais a 1.0 (W3C, 2008).

[2]http://www.section508.gov

As diretrizes WCAG 2.0 disponibilizam informações a respeito dos métodos conhecidos para o desenvolvimento de conteúdo em conformidade com suas diretrizes ao invés de descreverem condições tecnológicas para satisfazer os requisitos de acessibilidade. Dessa forma, assim como sua precursora, ela também define maneiras de como tornar o conteúdo web mais acessível a pessoas com deficiência (REID; SNOW-WEAVER, 2008; W3C, 2008).

Assim como ocorre na WCAG 1.0, a versão 2.0 também disciplina diretrizes. Essas diretrizes estão organizadas sob quatro princípios que constituem a base da acessibilidade web. São eles (W3C, 2008; WATANABE, 2014):

1. **Perceptível:** A informação e os componentes da interface devem ser apresentados de forma que os usuários os possam perceber/notar;
2. **Operável:** Os componentes da interface e a navegação têm de ser operáveis, ou seja, a interface não pode exigir interações com as quais o usuário não possa realizar;
3. **Compreensível:** As informações e a utilização da interface do usuário devem ser compreensíveis e estarem adequadas ao seu nível de conhecimento;
4. **Robusto:** O conteúdo deve ser suficientemente robusto para ser interpretado de forma fiel por uma ampla variedade de agentes de usuário, incluindo as Tecnologias Assistivas.

Não obstante a WCAG 2.0, o W3C publicou em junho de 2018 a WCAG 2.1. Assim como as anteriores, ela tem por objetivo fornecer diretrizes para tornar o conteúdo web mais acessível para pessoas com deficiência. Seu papel não é depreciar ou substituir sua precursora, portanto, o conteúdo que está em conformidade com a WCAG 2.1 também está em conformidade com a WCAG 2.0. A WCAG 2.1 vem a estender as diretrizes de acessibilidade da WCAG 2.0, publicada pelo W3C em dezembro de 2008 e, nesse contexto, o W3C aconselha o uso da WCAG 2.1 para maximizar a aplicabilidade futura dos esforços de acessibilidade e também incentiva o uso da versão mais atual das diretrizes para o desenvolvimento de novas políticas ou atualização de políticas de acessibilidade já existentes (W3C, 2018c).

A WCAG 2.1 provê critérios adicionais aos já ofertados pela WCAG 2.0. Os novos critérios da especificação endereçam orientações de acessibilidade para três grandes grupos (W3C, 2018d):

- pessoas com deficiências para o manuseio de dispositivos móveis para acesso a web;
- pessoas com baixa visão;

- pessoas com deficiências cognitivas ou de aprendizado.

Conforme o exposto, as diretrizes da WCAG fornecem um conjunto de boas práticas para o desenvolvimento de conteúdo web acessível não obstante às possíveis dificuldades ou deficiências inerentes aos cidadãos. Dentre os seus objetivos está o de proporcionar autonomia e acesso irrestrito aos conteúdos disponibilizados nos sites web, no entanto, esse conjunto de boas práticas não estabelece proposições técnicas a serem adicionadas às *tags* HTML e não prevê o estabelecimento de indicações semânticas aos *widgets*, em especial aos que disponibilizam conteúdos dinâmicos, sendo desta forma, insuficiente para proporcionar total acessibilidade nas RIAs.

2.2.2 ACESSIBILIDADE PARA APLICAÇÕES RICAS DE INTERNET (ARIA)

O estudo sobre acessibilidade na Web, dentre outras proposições, disponibiliza documentos e diretrizes com objetivo de tornar o conteúdo acessível para pessoas com deficiências, que por sua vez, podem fazer uso de Tecnologias Assistivas para navegar e esperam que com esses recursos elas possam "transformar" a apresentação do conteúdo web original em um formato que as permitam interagir sem barreiras de acessibilidade ao que está sendo disponibilizado. Por exemplo, o usuário pode precisar, ou optar por interagir com um *widget* de controle deslizante por meio do teclado, ao invés de arrastar e soltar com um mouse. Para conseguir isso efetivamente, o software precisa entender a semântica do conteúdo. Semântica é a ciência do significado; neste caso, usado para atribuir funções, estados e propriedades que se aplicam à interface do usuário e aos elementos de conteúdo para torná-los acessíveis, por exemplo, para ferramentas assistivas (W3C, 2017a).

Tratando especificamente sobre acessibilidade em RIAs, o W3C disponibilizou a ARIA, que assim como as WCAG, define uma maneira de tornar o conteúdo e os sites web mais acessíveis para pessoas com deficiências. No entanto, a ARIA visa acessibilidade especialmente para conteúdo dinâmico e controles avançados de interface de usuário desenvolvidos com Ajax, HTML, JavaScript e tecnologias relacionadas (W3C, 2017). Diferentemente da WCAG, a ARIA também estabelece um conjunto de regras, estados e propriedades a serem adicionados às *tags* HTML com o propósito de indicar semânticas aos *widgets* na composição da interface das RIAs.

Conforme é possível ser verificado na arquitetura das RIAs (Figura 2), linguagens como o JavaScript permitem que sejam realizadas atualizações dinâmicas nos dados e elementos HTML que compõem a interface de uma determinada página, adição de lógica e

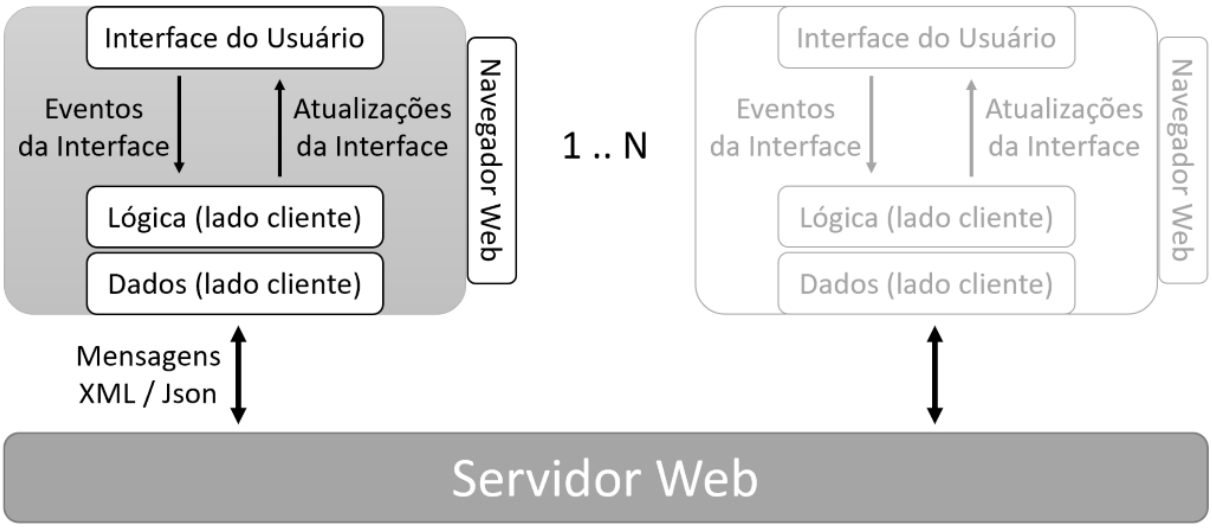

Figura adaptada pelo autor
Fonte: https://www.researchgate.net/figure/A-Rich-Internet-Application-architecture_fig5_221194734

Figura 2: Arquitetura das RIAs

validação de entrada de dados do lado cliente, chamadas assíncronas ao servidor por meio da classe XMLHTTPRequest, entre outras ações (GIBSON, 2007). Esses recursos, que compõem o Ajax e RIA, permitem que apenas partes de uma página web sejam atualizadas, melhorando a responsividade e a experiência do usuário durante sua interação, além de permitir o desenvolvimento de *widgets* semelhantes aos componentes de interface utilizados em aplicações *desktop* para criar interfaces com visual sofisticado e de alta interatividade (GIBSON; SCHWERDTFEGER, 2005; VELASCO et al., 2008).

O aumento de interatividade implementado por aplicações Ajax e RIA, apesar de propiciar benefícios em relação ao projeto de interface e usabilidade, podem apresentar barreiras de acessibilidade para usuários que interagem com o navegador por meio de Tecnologias Assistivas (MUNSON; PIMENTEL, 2008; THIESSEN; HOCKEMA, 2010). Segundo o estudo de GIBSON (2007), tais barreiras podem ser geradas devido a grande parte dessas características de interatividade exigirem percepção visual dos usuários e requererem interações do mouse para serem operadas. Além disso, a implementação dessas funcionalidades e componentes de interface para fins específicos e sem quaisquer tipos de padronização, também pode impactar negativamente na acessibilidade da aplicação (VELASCO et al., 2008).

Assim sendo, promover acessibilidade web em RIAs, fazendo uso apenas das diretrizes WCAG, pode não ser suficiente. As diretrizes WCAG focam o conteúdo da aplicação sem considerar os aspectos tecnológicos envolvidos para o seu desenvolvimento. Por outro lado, a especificação ARIA apresenta um *framework* para melhorar a acessibilidade e interoperabilidade do conteúdo e de RIAs, pois apresenta recomendações e especificações

de implementação para desenvolvedores de *widgets* customizadas e outros componentes que possam compor as aplicações web (WATANABE, 2014).

Dessa forma, por meio das especificações ARIA é possível adicionar atributos para identificar recursos para interação do usuário, como eles se relacionam entre si e seu estado atual. A ARIA descreve técnicas de navegação para marcar regiões e estruturas comuns da página como menus, conteúdo primário, conteúdo secundário, informações de banner e outros tipos de estruturas. Por exemplo, com a ARIA, os desenvolvedores podem demarcar regiões de páginas e permitir que os usuários que utilizem teclado para navegação se movam facilmente entre as regiões demarcadas, ao invés de precisar pressionar a tecla *Tab* várias vezes. Também inclui tecnologias para mapear controles, regiões ativadas e/ou modificadas por Ajax que se auto atualizam, como por exemplo áreas com dados que se modificam em períodos de tempo pré determinados, como um relógio, cotações da bolsa de valores, dentre outros, e eventos para prover acessibilidade para APIs, incluindo controles personalizados usados nas RIAs. As técnicas ARIA se aplicam a *widgets* como botões, listas suspensas, funções de calendário, controles de árvore e outros (W3C, 2017).

A ARIA disponibiliza aos desenvolvedores recursos como:

- Características (*roles*) para descrever o tipo de *widget* apresentado, como *"menu"*, *"treeitem"*, *"slider"*, *"progressmeter"* e outros;
- Características (*roles*) para descrever a estrutura da página web, como cabeçalhos, regiões e tabelas;
- Propriedades para descrever em qual estado os *widgets* estão, por exemplo, como *"checked"* para uma caixa de seleção ou *"haspopup"* para um menu;
- Propriedades para definir *live regions* de uma página com probabilidade de obter atualizações, bem como uma política de interrupção para essas atualizações. Por exemplo, atualizações críticas podem ser apresentadas em uma caixa de diálogo de alerta e atualizações corriqueiras ficarem embutidas na página;
- Propriedades para arrastar e soltar que descrevem as fontes de arrastar e os destinos de soltar;
- Uma maneira de prover navegação por teclado para objetos e eventos da página.

A Figura 3 ilustra um *widget* do tipo menu suspenso com um trecho de sua respectiva codificação HTML fazendo uso de alguns dos recursos da ARIA. Nela é possível notar o uso

de *roles* e propriedades para descrever o estado dos elementos. Neste exemplo, uma ferramenta assistiva para leitura de tela interpretaria o trecho de código identificando que há uma barra de menu para formatação de texto (linha 1 do trecho de código), que nesta barra de menu há uma opção rotulada como "*Text Align*", que essa opção de menu possui um elemento *popup* associado a ela e que no momento o elemento *popup* não está expandido (linha 5). Ainda no exemplo, as linhas de 8 a 21 identificam o *widget* do menu suspenso, sendo que as linhas entre 9 e 20 os subcomponentes do *widget*, ou seja, os itens do menu. Para os itens do menu a ferramenta assistiva para leitura de tela seria capaz de identificar as opções do menu selecionadas e as não selecionadas.

Caso os recursos da ARIA não estivessem presentes no trecho de código da Figura 3, Tecnologias Assistivas poderiam não ser capazes de identificar para os usuários quais configurações de formatação de texto estão selecionadas nos itens dos menus, quais das opções da barra de menu possuem submenus, se há alguma das opções do menu desabilitadas ou mesmo que em determinada região da tela há uma barra de menus disponível.

Outras informações técnicas sobre o uso dos recursos da ARIA e o modo como interpretá-las podem ser obtidas pela documentação WAI-ARIA *Authoring Practices* 1.1[3].

2.3 TRABALHOS RELACIONADOS

Os estudos sobre acessibilidade web perpassam sobre padrões estabelecidos por grupos de trabalho do W3C e diretrizes por eles estabelecidas. Tomando por base as diretrizes WCAG e ARIA, é possível notar que as mesmas direcionam os desenvolvedores para se preocuparem com processos de produção de conteúdo web e componentes de interface web que sejam aderentes aos seus padrões para acessibilidade. Dessa forma, recai sobre o processo de desenvolvimento e mais especificamente sobre o desenvolvedor, fazer uso das diretrizes para tornar o seu resultado final (os sites web) acessíveis (W3C, 2016, 2017). Assim, disponibilizar mecanismos que auxiliem o desenvolvedor web a identificar automaticamente pontos em sua codificação que requeiram maior atenção em relação ao uso de padrões de acessibilidade, pode contribuir com o processo de desenvolvimento desse tipo de aplicação e com a adoção das respectivas diretrizes da W3C.

Sob outra ótica, também há estudos que caminham pela linha de adaptação de código HTML pelas Ferramentas Assistivas, com a intenção de que as mesmas "transformem" o conteúdo e componentes de interface web acessíveis, mesmo que esses conteúdos ou

[3]Disponível em: https://www.w3.org/TR/wai-aria-practices/


```
<ul id="menubar1" role="menubar" aria-label="Text Formatting">
    ...
    ...
    <li role="none">
      <a role="menuitem" aria-haspopup="true" aria-expanded="false" href="#">
        Text Align
      </a>
      <ul role="menu" rel="text-align" aria-label="Text Align">
        <li role="menuitemradio" aria-checked="true">
            Left
        </li>
        <li role="menuitemradio" aria-checked="false">
            Center
        </li>
        <li role="menuitemradio" aria-checked="false">
            Right
        </li>
        <li role="menuitemradio" aria-checked="false">
            Justify
        </li>
      </ul>
    </li>
    ...
    ...
</ul>
```

Figura adaptada pelo autor

Fonte: https://www.w3.org/TR/wai-aria-practices/examples/menubar/menubar-2/menubar-2.html

Figura 3: Exemplo de código HTML com recursos da ARIA

componentes não tenham sido preparados segundo as padronizações de acessibilidade do W3C.

Tomando por base apenas as RIAs e as regras de acessibilidade previstas pela ARIA, a tarefa para indicar a conformidade ou adaptação do código HTML dos *widgets*, requer trabalho prévio de classificação dos mesmos. Uma vez classificados, é possível verificar se os atributos de regras, estados e propriedades, previstos pela ARIA, estão presentes na codificação dos *widgets* em questão e, caso não estejam, promover a adaptação dos seus respectivos códigos fonte para torná-los acessíveis.

O estado da arte sobre acessibilidade com base nas especificações da ARIA reportam estudos e ferramentas sobre: **(a)** avaliação de conformidade das RIAs com as regras da ARIA, onde são discutidos modelos, métodos e ferramentas para avaliação de acessibilidade segundo as premissas da ARIA, **(b)** classificação automática de *widgets*, onde se consideram estratégias e técnicas para automatizar o reconhecimento e classificação automática de *widgets* nas páginas web e **(c)** adaptação automática do código HTML para requisitos de acessibilidade, onde são discutidas abordagens para inclusão automatizada de *tags* ARIA nos *widgets* para torná-los

acessíveis ou mesmo a substituição dos *widgets* originais das páginas web por *widgets* genéricos que possam atender a mesma demanda dos originais, porém com os padrões de acessibilidade previamente estabelecidos.

Esta dissertação de mestrado se encaixa aos estudos indicados pelo conjunto **(b)**, ou seja, classificação automática de *widgets*. Ela também organiza os trabalhos relacionados com base nos temas identificados anteriormente.

2.3.1 AVALIAÇÃO DE CONFORMIDADE DAS RIAS COM AS REGRAS DA ARIA

DOUSH et al. (2013) introduziram um *framework* conceitual para avaliação automática de acessibilidade em RIAs baseado nas regras da ARIA. O *framework* proposto pelos autores é composto pelos seguintes componentes:

- **Controlador de Eventos RIA:** identifica os elementos presentes no DOM que contenham *listerners* de eventos atrelados a eles;
- **Robô Web:** gera entradas de eventos nativos para simular a interação do usuário com a aplicação web;
- **Especificação ARIA:** corresponde a um conjunto de regras da ARIA especificadas em um arquivo de configuração em formato XML;
- **Avaliador:** o módulo avaliador é composto pelos seguintes subcomponentes: Analisador DOM, Classificador de elementos, Testes de especificação ARIA e Ontologia ARIA;
- **Manipulador de Resultados:** tem por objetivo gerar relatórios dos resultados da avaliação.

DOUSH et al. (2013) também sugerem que uma ferramenta para avaliação automática de acessibilidade deve conter as seguintes características: **(a)** destacar partes da página web que requerem inspeção por parte do desenvolvedor; **(b)** ser capaz de fazer o *download* do código HTML hospedado para avalia-lo localmente, ou seja, de forma *offline*; **(c)** gravar os resultados da avaliação de acessibilidade em formato compreensível tanto para pessoas quanto para máquinas.

WATANABE et al. (2017) relataram o desenvolvimento de duas abordagens distintas para avaliar automaticamente os requisitos da ARIA em aplicações web, com base em testes de aceitação. Na primeira abordagem, foi implementada uma extensão da ferramenta de

teste de aceitação, denominada *Pyccuracy*, para escrever casos de teste de aceitação capazes avaliar os requisitos de acessibilidade em conteúdo HTML gerado dinamicamente. Na segunda abordagem os autores desenvolveram um *plugin* para o navegador Firefox que usa testes de aceitação para testar requisitos da ARIA em *widgets*. As duas abordagens foram propostas com o objetivo de incluir modelos de interação de usuários com deficiência em estratégias de avaliação de acessibilidade. Os autores argumentam que a inclusão do modelo de interação de usuários com deficiência nas abordagens de avaliação possibilita a análise dos requisitos ARIA, por meio de cenários de simulação desse tipo de interação. Os autores ainda conduziram dois estudos de caso para validar sua abordagem de avaliação automática de acessibilidade. No primeiro estudo de caso, a ferramenta de teste de aceitação, denominada *Pyccuracy*, foi estendida para permitir a inclusão de características de modelos de acessibilidade em seus casos de teste. No segundo estudo de caso, eles implementaram a ferramenta *aria-check*, com objetivo de mapear o modelo de interação de usuários com deficiência com tipos específicos de *widgets* em testes de aceitação.

2.3.2 CLASSIFICAÇÃO AUTOMÁTICA DE *WIDGETS*

CHEN et al. (2013) implementaram uma abordagem para classificação de *widgets* com foco aos que se encaixam nos tipos *Slideshow* e *Carousels*. A abordagem proposta por eles procura por construções de código JavaScript que possam implementar a funcionalidade dos *widgets*, como por exemplo, botão de próximo, botão de anterior, e outros componentes de tela potencialmente relacionados aos *widgets*. A técnica dos autores baseia-se na identificação dessas funcionalidades e na análise da combinação das mesmas para indicar a presença ou não de um determinado *widget* na página web que está sendo analisada. Uma ontologia foi desenvolvida para atuar como um sistema de classificação para definição de *widgets*, pois há casos onde os *widgets* compartilham componentes comuns, assim como os desenvolvedores geralmente determinam nomes para os *widgets* de forma similar. O sistema de classificação foi baseado na funcionalidade das partes dos componentes dos *widgets*, ao invés de depender de convenções de nomenclatura fornecidas pelos desenvolvedores.

MELNYK et al. (2014) descreveram uma técnica que usa aprendizado de máquina para classificar quatro tipos populares de *widgets* (*Suggestion box, Alert box, Dropdown menu* e *Datepicker*) em páginas web. Os autores propuseram um conjunto de características para compor *datasets* baseados nas mudanças que foram observadas na estrutura DOM das páginas web. Na sequência eles compararam a eficácia de classificação dos algoritmos *Decision Tree* e *Support Vector Machine*. O conjunto de características usado no estudo de MELNYK et

al. (2014) inclui: A presença da *tag* TABLE no código HTML, presença da *tag* UL no HTML, presença da *tag* INPUT no HTML, presença do nome do *widget* no atributo CLASS do elemento HTML, presença da data no atributo VALUE em qualquer *tag* do HTML, presença de uma imagem no HTML, alterações no HTML que ocorrem após o clique em um botão ou link, mudanças no HTML que ocorrem como resultado de uma entrada de dados em um campo texto, o número de nós do tipo texto, proporção de nós do tipo texto que contém apenas números, uma tabela ou lista onde seu conteúdo é composto por 80% ou mais de *tags* como links ou ancoras e o elemento ativador estar próximo ao *widget* ou não.

WATANABE; FORTES (2016) abordam o desenvolvimento de uma ferramenta que classifica automaticamente instâncias de *widgets* do tipo menu suspenso, com base em alterações ocorridas na estrutura DOM de aplicações web. Segundo os autores, os *widgets* do tipo menu suspenso podem ser implementados usando dois tipos distintos de elementos HTML: os que caracterizam o elemento ativador do menu, e os elementos de contêiner onde as opções do menu estão inseridas. O ativador do menu é o elemento exibido quando a página web é carregada no navegador. O elemento ativador normalmente aciona uma função JavaScript para mostrar um menu suspenso quando os usuários interagirem com ele. O contêiner do menu é inicialmente um elemento oculto para o usuário e somente é exibido quando o usuário interage com o elemento ativador de um menu específico. Assim, a abordagem dos autores consiste em simular o evento *mouse over* sobre o elemento ativador, registrando as mudanças na estrutura DOM para identificar o *widget* ativador e o *widget* que contém os elementos do contêiner. Os autores sugerem que esses elementos podem ser analisados para determinar se eles implementam as especificações das regras da ARIA, auxiliando os desenvolvedores a construir sites web acessíveis.

ANTONELLI et al. (2018) propõem o uso do aprendizado de máquina para classificar automaticamente as mudanças da estrutura DOM de uma página web em um determinado tipo de *widget*. A abordagem envolve a identificação dos menus suspensos que são implementados por meio da linguagem JavaScript. As alterações na estrutura do DOM foram capturadas como registros de mutação, por meio da API *MutationObserver*. O conjunto de características usado para treinar os algoritmos de aprendizado de máquina foi extraído dos atributos dos elementos HTML do DOM que sofreram alteração na aplicação web. As características foram selecionadas com base no comportamento dos *widgets* do tipo menu suspenso, bem como sua posição, exibição na aplicação web, posição do ativador, distância do *widget* em relação ao seu ativador, dimensões do *widget*, elementos HTML dentro do *widget* e número de palavras por nós de texto. Os autores também consideraram que a identificação automática de *widgets* é um componente essencial para a elaboração de avaliação automática de ARIA e estratégias de

adaptação de código HTML.

2.3.3 ADAPTAÇÃO AUTOMÁTICA DO CÓDIGO HTML PARA REQUISITOS DE ACESSIBILIDADE

CHEN (2010) propôs um projeto denominado WIMWAT. O objetivo do autor foi disponibilizar uma ferramenta para ajudar os desenvolvedores web a analisar a acessibilidade dos *widgets* enquanto desenvolviam as RIAs. Além da análise dos padrões de acessibilidade dos *widgets* a ferramenta também poderia melhorar a acessibilidade dos mesmos por meio da inclusão de *tags* ARIA em seus respectivos códigos fonte. Segundo o autor, o projeto foi dividido em três estágios: **(a)** o estágio de identificação; **(b)** o estágio de observação e **(c)** o estágio de modificação. No estágio de identificação, o código HTML de uma página web é varrido para encontrar traços de algum *widget*. Usando esse método em conjunto com regras que ajudam distinguir diferentes tipos de *widgets*, é possível descobrir os *widgets* de uma página web por meio da aplicação de expressões regulares. No estágio de observação, o código fonte dos *widgets* é analisado e sua conformidade com os padrões de acessibilidade são checados. Já no estágio de modificação, as adaptações de código HTML necessárias são feitas para deixar os *widgets* mais acessíveis.

MELNYK et al. (2015) propõem uma abordagem para tornar *widgets* acessíveis provendo uma interface genérica para *widgets* de uma determinada classe, como por exemplo, *chats*, menus suspensos, calendários e outros. Os autores sugerem que um *widget* pode se tornar acessível para um leitor de telas em três etapas principais: **(a)** detectar o *widget* na página web quando ele aparece dinamicamente, **(b)** identificar o tipo do *widget* e **(c)** prover uma interface genérica, em tempo real, para que o usuário possa interagir com um *widget* de um determinado tipo. Segundo os autores, as etapas (a) e (b) foram descritas em estudo anterior e podem ser consultadas em MELNYK et al. (2014). Para a etapa (c) eles fizeram uma prova de conceito estendendo o Capti-SR - Software para Leitura de Tela. O trabalho se baseou no mapeamento de *widgets* do tipo *chat*, em páginas web, para prover uma interface genérica acessível para esse tipo de componente. Assim, a abordagem proposta por MELNYK et al. (2015) não injeta regras da ARIA no código HTML dos *widgets* para adaptá-los, mas ela sugere que os leitores de tela ofereçam interfaces genéricas acessíveis para diferentes tipos de *widgets* quando os mesmos não estão preparados para tal.

2.4 CONSIDERAÇÕES FINAIS

Para esta dissertação de mestrado, foram observados os conhecimentos prévios identificados no estado da arte deste campo de pesquisa. Neles foram coletadas similaridades e/ou relações com base no que propõe o contexto deste trabalho, ou seja, o uso de um *pipeline* de aprendizado de máquina para classificar *widgets* do tipo *dropdown menu* e *suggestion box*, bem como classificar os seus respectivos subcomponentes, como por exemplo, os itens de menu, os submenus e os itens de sugestões de autocomplemento referentes aos *suggestion boxes*.

A classificação dos subcomponentes de um *widget* permite a correlação entre os seus respectivos elementos HTML em relação ao *widget* principal. Como essa estratégia aponta onde os subcomponentes estão inseridos na árvore de mutação DOM do *widget* principal, além de verificar o próprio *widget*, também é possível verificar a conformidade dos seus subcomponentes em relação às regras da ARIA e vislumbrar a adaptação automática dos seus respectivos códigos HTML para torná-los acessíveis quando for o caso.

Os trabalhos relacionados foram organizados em três temas. Os estudos referentes a avaliação de conformidade das RIAs com as regras da ARIA apresentaram os trabalhos de DOUSH et al. (2013) e de WATANABE et al. (2017). O primeiro apresentou um framework conceitual para criação de ferramentas para avaliação automática de acessibilidade em RIAs e, desse framework, em especial o "módulo avaliador", devido a sua proximidade com o tema deste trabalho. O segundo trabalho apresenta uma ferramenta para testes de acessibilidade em *widgets* e métodos distintos dos desta dissertação para validação de conformidade com base nos requisitos ARIA.

Já os trabalhos dos autores CHEN et al. (2013), MELNYK et al. (2014), WATANABE; FORTES (2016) e ANTONELLI et al. (2018) foram relacionados ao tema de classificação automática de *widgets*. Por meio deles foi possível identificar diferentes estratégias para captura e organização dos dados e para a classificação dos *widgets*. Para o processo de classificação, observou-se o uso de ontologias criadas pelos próprios autores ou o uso de algoritmos de aprendizado de máquina. Para a captura dos dados foram usados *crawlers*, coleta manual e por meio de Javascript com o uso da biblioteca *MutationObserver API* relacionada às mutações do DOM. Além dos pontos já indicados, esses trabalhos também propiciaram a seleção de algumas das características que compõem o conjunto de dados deste estudo.

Nos estudos de CHEN (2010), MELNYK et al. (2015) e MELNYK et al. (2014), identificados pelo tema adaptação automática do código HTML para requisitos de acessibilidade, constatou-se o desenvolvimento de ferramentas e técnicas para avaliação de

conformidade das RIAs com os padrões de acessibilidade da ARIA e detalhes sobre os métodos utilizados em cada abordagem. Os métodos para adaptação de código HTML citados nos trabalhos dos autores vão desde a injeção de *tags* ARIA no código HTML das RIAs até a substituição por completo de um *widget* não acessível por outro, compatível em termos de funcionalidades, previamente preparado para garantia de acessibilidade por ferramentas assistivas, como por exemplo, leitores de tela.

No próximo capítulo é apresentada uma proposta para classificação automática de *widgets* do tipo *dropdown menu*, *suggestion box* e dos seus subcomponentes. Nele é possível observar correlações entre os trabalhos relacionados e a pesquisa desenvolvida nesta dissertação, que dentre os temas indicados acima, está mais aderente aos estudos dos autores que exploraram estratégias para classificação de *widgets*.

3 *PIPELINE* DE CLASSIFICAÇÃO DE *WIDGETS* E DOS SEUS SUBCOMPONENTES

Esta dissertação de mestrado propõe o uso de um *pipeline* de aprendizado de máquina para classificação automática de *widgets* e dos seus subcomponentes em páginas web. O reconhecimento dos *widgets* e dos seus subcomponentes foi modelado como um problema de classificação para determinar se uma mutação ocorrida na estrutura DOM de uma página web é um *widget* ou não e onde seus subcomponentes diretos estão localizados. Uma mutação na estrutura DOM pode ser explicada como quaisquer mudanças na estrutura HTML de uma página web que ocorreram por meio de uma ação do usuário ou por efeitos visuais gerados por classes CSS ou códigos em JavaScript (MDN MOZILLA WEB DOCS, 2012).

O uso de aprendizagem de máquina para classificação dos *widgets* e dos seus subcomponentes foi adotado devido a variedade de combinações de código HTML capazes de compor um mesmo tipo de *widget*. Dessa forma, a especificação de ontologias de classificação com base em expressões regulares ou por varredura de código e posterior identificação de padrões não seria flexível o suficiente para ampla cobertura das diferentes estruturas de código possíveis para implementação desses elementos.

Páginas web ou aplicações web normalmente são compostas por diferentes tipos de *widgets*, como por exemplo, *sliders*, *dropdown menus*, calendários, *tooltips*, *suggetion boxes* e outros. Nesta dissertação, como prova de conceito da abordagem, o *pipeline* de classificação foi instrumentalizado para o reconhecimento de *widgets* do tipo *dropdown menu* e *suggestion box*, bem como dos seus respectivos subcomponentes. Em estudo anterior de RIZO et al. (2019), os autores fizeram uso da mesma abordagem para classificar apenas *widgets* do tipo *dropdown menu*, o que sugere que o *pipeline* de aprendizado de máquina pode ser utilizado para classificar outros tipos de *widgets*. A Figura 4 apresenta uma visão simplificada da abordagem desta dissertação e dos módulos que a compõem:

A. *Logger:* Extensão de navegador web projetada para aquisição automatizada dos dados (mutações);

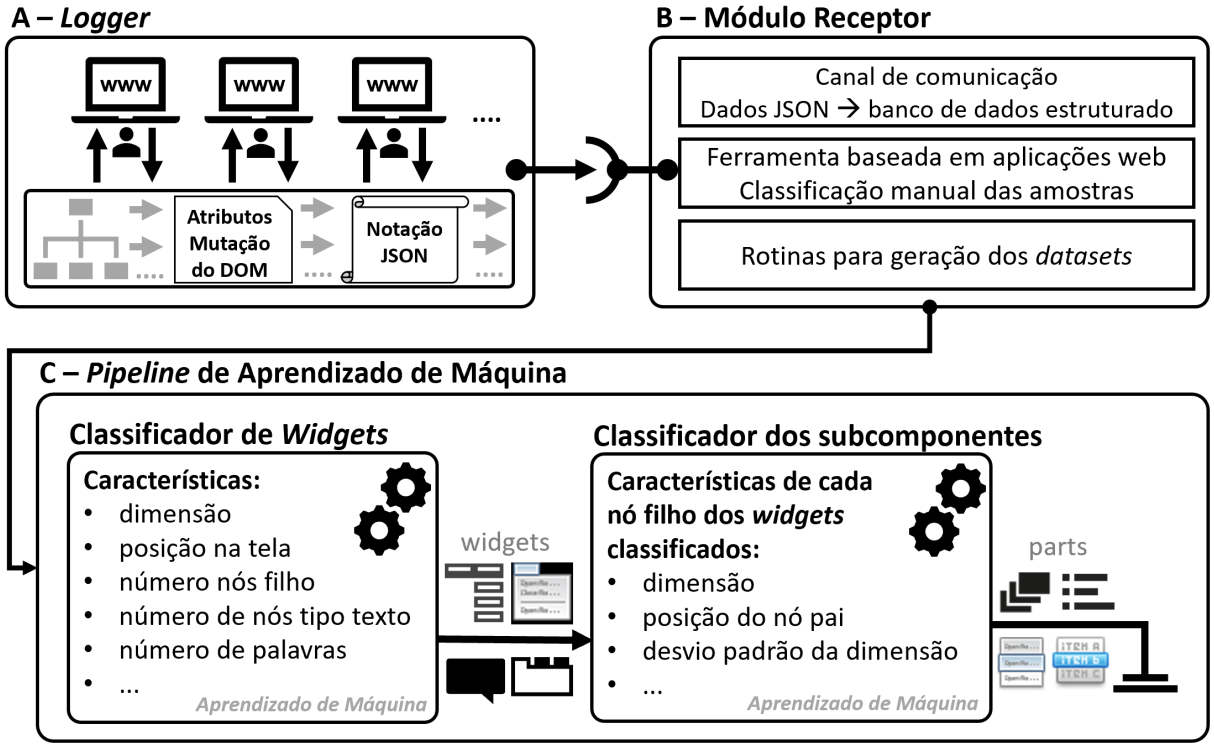

Figura 4: Visão simplificada da abordagem

B. Módulo Receptor: Projetado para persistência dos dados (mutações) e para disponibilizar ferramenta de auxílio na classificação manual dos *widgets* e dos subcomponentes presentes nas mutações do DOM;

C. Pipeline de Aprendizado de Máquina: Planejado para classificação automática dos *widgets* presentes nas mutações do DOM e dos seus respectivos subcomponentes.

A ideia central do modelo e o relacionamento entre os módulos pertencentes a ele podem ser resumidos da seguinte forma: as mudanças na estrutura HTML das páginas web são coletadas pelo ***Logger*** e enviadas para o **Módulo Receptor** para persistência em um banco de dados estruturado para esse fim. Na sequência, são gerados os conjuntos de dados para treinar o **Pipeline de Aprendizado de Máquina** afim de que o mesmo possibilite predições. Uma vez que o *pipeline* esteja ajustado, ele poderá ser incorporado em ferramentas automatizadas para validação de conformidade das RIAs para com as regras da ARIA ou mesmo em ferramentas para adaptação automática do código HTML dos *widgets* para torná-los acessíveis. Esses tipos de ferramenta pressupõem a classificação prévia dos *widgets* para que elas possam ser implementadas, sendo este o ponto em que esta dissertação contribui.

3.1 ESTRATÉGIA DE CAPTURA DE *LOGS*

Tomando por base os trabalhos relacionados na seção 2.3 desta dissertação, a captura ou interceptação dos *logs* das mutações que ocorrem na estrutura DOM das páginas web é uma das estratégias utilizadas pelos autores para propiciar técnicas para classificação de *widgets* que compõem as interfaces de usuário das RIAs. Não obstante aos estudos anteriores, este trabalho propõe uma estratégia similar de captura de *logs*, porém de forma não intrusiva à navegação dos usuários nas páginas web, automatizada e com coleta de dados de sites reais da Internet.

3.1.1 *LOGGER*

O *Logger*, nomeado como ARIA Observer, é um *plugin* implementado para o navegador Google Chrome, com objetivo específico de oferecer suporte ao processo de aquisição de dados. O *Logger* foi implementado como uma extensão do navegador web (*plugin*) para possibilitar a coleta de registros de mutações ocorridas no DOM de quaisquer sites da Web. Adicionalmente, como uma extensão do navegador web, ele pode ser distribuído e instalado em outros computadores, o que provê a escalabilidade do processo de coleta dos dados.

A base da implementação do *Logger* se deu por meio da biblioteca JavaScript *MutationObserver API*[1], pois a mesma provê funções que permitem interceptar e colher informações a respeito das mudanças ocorridas na estrutura DOM de uma página web (MDN MOZILLA WEB DOCS, 2012). Assim sendo, o objetivo do *Logger* é coletar automaticamente registros de mutações da estrutura DOM de sites web específicos.

Como qualquer outra extensão de navegador web (GOOGLE, 2016), o *Logger* precisa ser instalado nos navegadores web para que possa ser executado. Como parte de suas premissas, ele precisa ser instrumentalizado com uma lista de sites web a serem observados. A configuração da lista dos sites web a serem observados ocorre por meio de um arquivo de manifesto atrelado ao *plugin* do *Logger*. Portanto, quando o usuário navega por uma página web que pertença a lista de sites web apresentadas ao *Logger*, ele passa a observar todas as mudanças ocorridas na estrutura DOM da página em questão. É importante notar que todos os dados coletados se baseiam na experiência de navegação natural do usuário e de forma não intrusiva. Consequentemente, o usuário pode navegar normalmente pelas páginas web enquanto o *Logger* coleta todas as mudanças em suas estruturas HTML por meio da biblioteca *MutationObserver API*.

Para cada mutação observada, o *Logger* organiza os dados coletados em uma *string*

[1] https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver

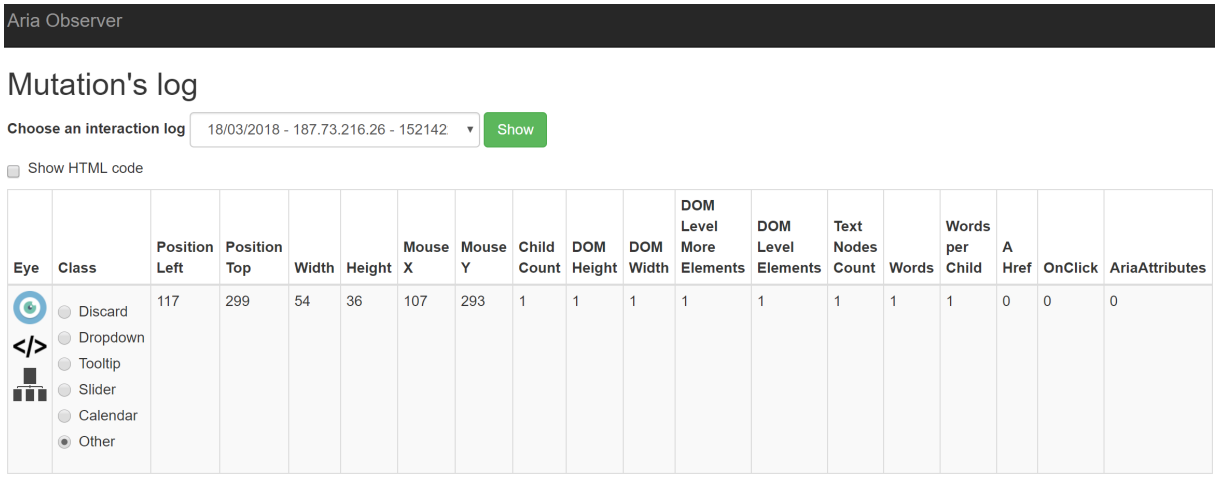

Figura 5: Ferramenta para classificação manual dos registros de mutação do DOM

JavaScript Object Notation (JSON) e a envia para o Módulo Receptor para persistência em um banco de dados estruturado. A comunicação entre o *Logger* e o Módulo Receptor ocorre por meio de um canal de comunicação web estabelecido via *websockets*[2]. Finalmente, as características enviadas para o banco de dados serão usadas para posterior classificação manual dos *widgets* e dos seus subcomponentes.

3.1.2 MÓDULO RECEPTOR E CLASSIFICAÇÃO DAS AMOSTRAS

O Módulo Receptor é uma aplicação web[3] que implementa três funções necessárias para a estratégia de captura automatizada dos *logs*, gerados pelas mutações ocorridas no DOM das páginas que compõem as RIAs. A **primeira função** estabelece um canal de comunicação para obter os registros de mutação (dados em formato JSON) que são enviados pelo *Logger*. Os dados recebidos pelo Módulo Receptor passam por um processo de conversão para possibilitar sua persistência em um banco de dados relacional. A persistência dos dados em um banco de dados relacional facilita a condução do processo de classificação manual. A **segunda função** provê uma ferramenta web para facilitar a tarefa de classificação manual dos dados capturados pelo *Logger* (Figura 5). Os dados coletados são organizados por data de captura e pelo nome do domínio onde o site web está hospedado. Neste módulo também está disponível uma ferramenta para auxiliar a classificação manual dos registros de mutação e identificar o tipo dos *widgets* e os seus subcomponentes (Figura 6). Finalmente, após classificar manualmente todos os registros de mutação, a **terceira função** do módulo receptor é usada para gerar os arquivos dos conjuntos de dados a serem analisados pelo *pipeline* de aprendizado de máquina desta abordagem.

[2]https://developer.mozilla.org/en-US/docs/Web/API/WebSockets-API

[3]Recursos utilizados para o desenvolvimento: Visual Studio 2017, ASP.NET MVC, C#, Javascript e Banco de Dados SQL Server

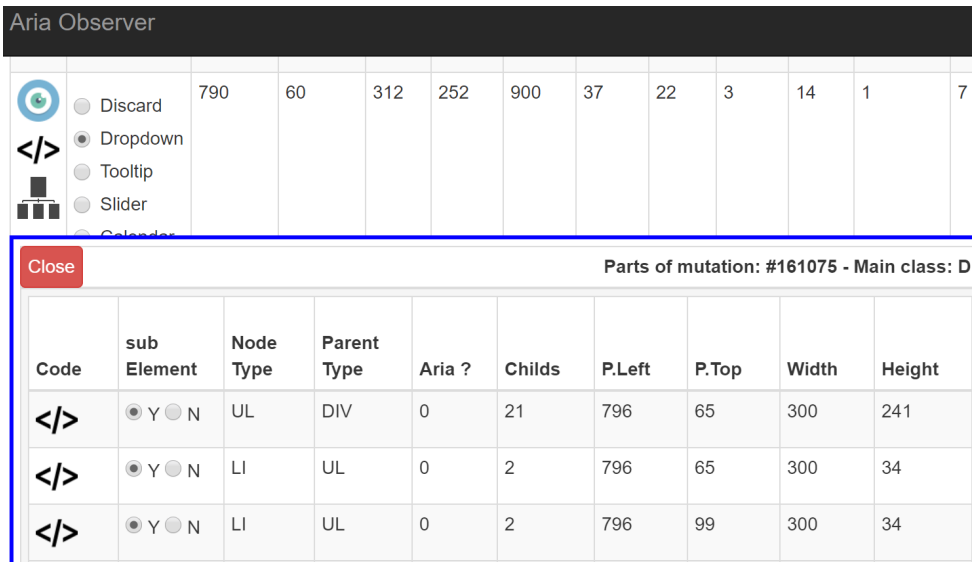

Figura 6: Classificação manual dos subcomponentes dos *widgets*

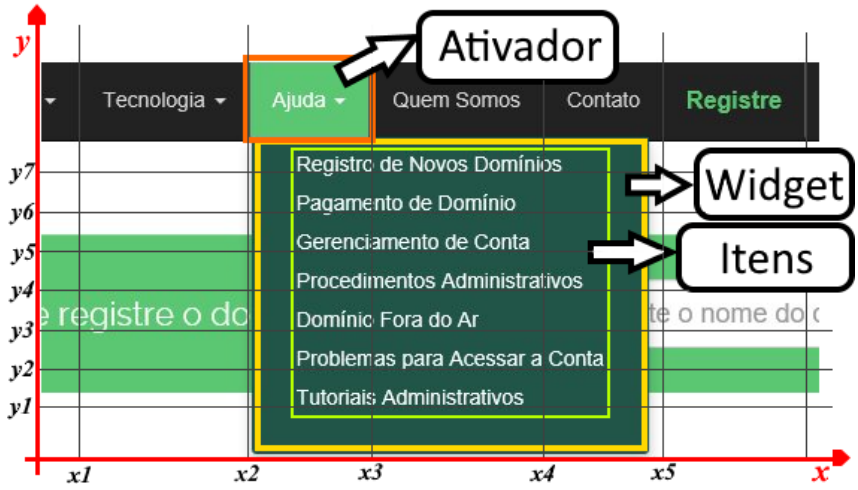

Figura 7: Organização dos dados das mutações do DOM

3.2 PIPELINE DE CLASSIFICAÇÃO DE *WIDGETS* E SUBCOMPONENTES

O *pipeline* de aprendizado de máquina desta dissertação de mestrado tem por objetivo classificar *widgets* do tipo *dropdown menu*, *suggestion box* e os seus respectivos subcomponentes. Os registros de mutações do DOM coletados pelo *Logger* de páginas web de RIAs e enviados para o Módulo Receptor, são usados para gerar conjuntos de dados que servem como entrada de dados para treinamento de dois classificadores supervisionados (compondo o *pipeline*), para que posteriormente eles sejam capazes de oferecer predições. A Figura 7 apresenta como as características que compõem os conjuntos de dados estão organizadas.

O **primeiro** algoritmo de aprendizado de máquina do *pipeline* tem por objetivo classificar *widgets* do tipo *dropdown menu* e *suggestion box* (Figura 8). O conjunto de dados

apresentado a ele inclui as seguintes características:

- **Tamanho do *widget*:** representa a dimensão do *widget* em relação a sua largura e altura. *Widgets* do tipo *dropdown menu* e *suggestion box* podem ser projetados com diferentes tamanhos, no entanto, é esperado que suas dimensões sejam maiores que um botão, um ícone ou *widgets* do tipo *tooltip*. O *suggestion box* tende a ter a mesma largura do seu elemento ativador, como por exemplo, uma caixa de texto;

- **Posição do *widget*:** informações sobre o posicionamento do *widget* em relação a margem superior e esquerda da página web. É esperado que um *widget* do tipo *dropdown menu* esteja localizado próximo a borda superior ou a borda esquerda da página web. Já os *widgets* do tipo *suggestion box* normalmente tendem a ter seu posicionamento mais ao centro e na parte superior da página web;

- **Posição do ativador:** identifica a posição do elemento que ativou uma determinada mutação na estrutura DOM de uma página web. Os *widgets* do tipo *dropdown menu* usualmente são ativados quando o usuário clica ou passa o ponteiro do mouse sobre algum elemento que compõe a interface, como por exemplo, um botão, um link, entre outros. Os *widgets* do tipo *dropdown menu* normalmente estão posicionados próximos ao seu ativador, portanto, a distância entre a posição do *widget* e do seu atividador tende a ser pequena. Os *widgets* do tipo *suggestion box* usualmente são ativados por um clique ou pela ação de digitação do usuário em um elemento HTML do tipo *text*. A posição do seu ativador tende a ser a mesma onde o *suggestion box* está posicionado;

- **Número de filhos:** representa o número de elementos HTML que compõem os *widgets*. *Widgets* do tipo *dropdown menu* e *suggestion box* tendem a ter mais que um elemento HTML em suas estruturas internas, sendo assim, *widgets* compostos por menos que dois elementos provavelmente não são do tipo *dropdown menu* ou *suggestion box*;

- **Dimensão da árvore DOM do *widget*:** é a dimensão relativa à quantidade de elementos e ao nível de profundidade da árvore DOM do *widget* que sofreu a mutação, onde a quantidade indica o número de elementos HTML em cada nó da árvore e a profundidade indica os seus respectivos subelementos. Assim, em relação ao *widget* do tipo *dropdown menu*, esta característica pode sugerir a relação dos seus itens ou mesmo a presença de submenus dentro dos itens existentes. Já para o *widget* do tipo *suggestion box* esta característica pode indicar a relação dos itens sugeridos para a digitação realizada, no entanto, o quesito profundidade do *suggestion box* tende a ser menor que o do *dropdown menu* visto que o mesmo normalmente não apresenta subníveis de sugestões;

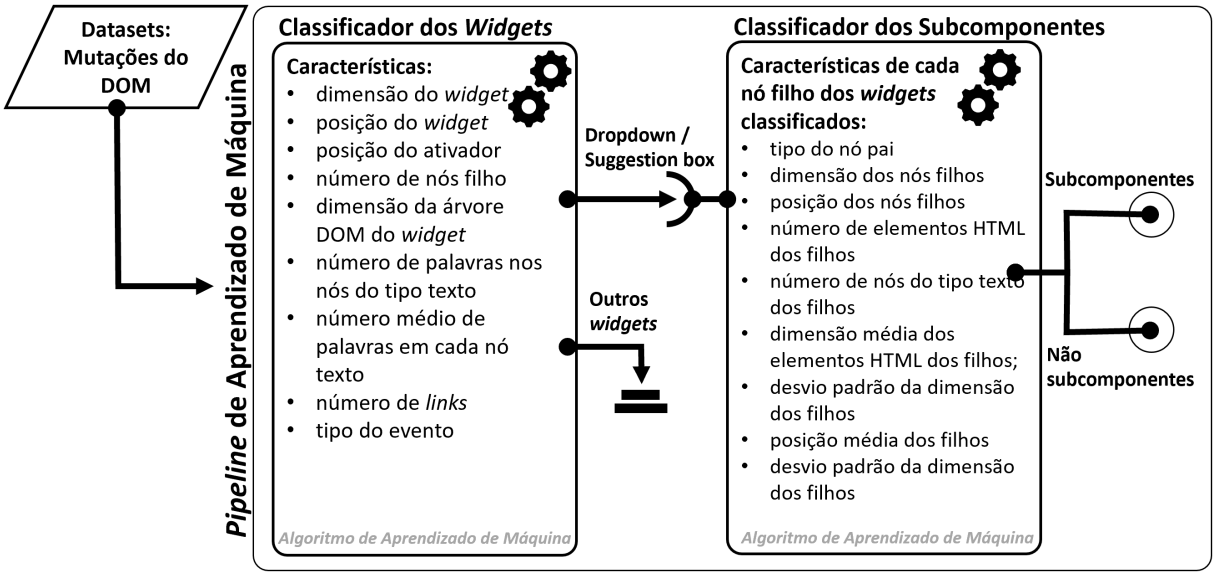

Figura 8: *Pipeline* de aprendizado de máquina

- **Número de palavras em nós do tipo texto:** quantifica o número total de palavras em todos os nós do tipo texto presentes nos *widgets*. É esperado que um *widget* do tipo *dropdown menu* tenha poucas palavras em cada um dos seus nós do tipo texto, visto que cada nó tende a representar um item de menu do *widget*. Entretanto, o número de palavras em nós do tipo texto de *widgets* do tipo *suggestion box* tende a ser maior que o do *dropdown menu* devido a sua característica de indicar textos sugestivos referentes à digitação inicial feita pelo usuário na página web;

- **Número médio de palavras por nó do tipo texto:** representa o número médio de palavras em cada nó do tipo texto do *widget*. Observações empíricas dos pesquisadores demonstraram que os itens de um *dropdown menu* tendem a ter entre uma e três palavras em cada nó do tipo texto que o compõe. Já nos *widgets* do tipo *suggestion box* esse número é mais esparso, mas normalmente acima de 3 palavras em média;

- **Número de links:** é o número total de links encontrados dentro do *widget*. É esperado que *widgets* do tipo *dropdown menu* e *suggestion box* tenham um número de links igual ou menor que o número de elementos filho que os compõem;

- **Tipo do Evento:** representa o tipo do evento que provocou a mutação na estrutura DOM. Essa característica é importante pois diferentes tipos *widgets* podem ser compostos por estruturas de código HTML similares, como é o caso entre *widgets* do tipo *dropdown menu* e *suggestion box*.

O **segundo** classificador do *pipeline* foca na classificação dos subcomponentes

(elementos filho) dos *widgets* do tipo *dropdown menu* e *suggestion box*, que foram classificados pelo primeiro algoritmo de aprendizado de máquina do mesmo *pipeline* (Figura 8). Consequentemente, o classificador dos subcomponentes é executado somente quando o primeiro classificador identifica um dos *widgets*, dos tipos mencionados acima, nos registros de mutação do DOM. O segundo classificador foi modelado a partir das seguintes características pertencentes aos subcomponentes dos *widgets* dos tipos *dropdown menu* e *suggestion box*:

- **Tipo do nó pai do elemento filho:** representa o tipo do nó pai no qual o elemento filho está inserido. Os subcomponentes de um *dropdown menu* e *suggestion box* normalmente são dispostos em *tags* contêineres do HTML, como por exemplo, DIV, UL ou outras;
- **Tamanho dos nós filhos do *widget*:** informação sobre a largura e a altura (em pixel) de cada nó filho que compõe o *widget* identificado pelo primeiro classificador do *pipeline*. É esperado que as dimensões dos itens que compõem um *widget* do tipo *dropdown menu* e *suggestion box* sejam similares;
- **Posição dos nós filhos do *widget*:** representa a posição de cada nó filho do *widget* em relação à borda esquerda e superior da página web. Um *widget* do tipo *dropdown menu* tende a ter seus itens na mesma posição horizontal ou vertical dependendo de sua orientação. *Dropdown menus* com orientação horizontal tendem a ter seus itens na mesma posição do eixo y e variam seus posicionamentos no eixo x. Por outro lado, quando a orientação do *dropdown menu* é vertical, os itens variam suas posições no eixo y e mantêm a mesma posição no eixo x (Figura 7). Os subcomponentes dos *widgets* do tipo *suggestion box* possuem a mesma característica dos *dropdown menus* quando em orientação vertical, portanto, seus itens variam o posicionamento em relação ao eixo y e tendem a manter a mesma posição no eixo x;
- **Número de elementos HTML contidos nos nós filhos:** quantifica o número total de elementos HTML inseridos em cada nó filho que compõe o *widget* identificado pelo primeiro classificador do *pipeline*. É esperado que os subcomponentes de um *dropdown menu* ou de um *suggestion box* sejam compostos por outros elementos HTML, como por exemplo, links, ancoras, chamadas a funções JavaScript e outros. A existência de um item de *dropdown menu* com número elevado de outros elementos HTML inseridos dentro dele, pode sugerir a representação de um submenu do *dropdown menu* identificado pelo primeiro classificador do *pipeline*;
- **Número de nós do tipo texto contidos nos nós filhos:** quantifica o número total de nós do tipo texto presentes em cada nó filho do *widget* identificado pelo primeiro classificador

do *pipeline*. No caso de itens de *widgets* do tipo *dropdown menu* e *suggestion box*, é esperado que eles tenham pelo menos um nó do tipo texto a ser apresentado como opção de menu aos usuários;

- **Dimensão média dos elementos HTML dos nós filhos:** representa a dimensão média (largura e altura separadamente) de todos elementos HTML inseridos em cada descendente direto de cada item que pertence ao *widget* que foi identificado pelo primeiro classificador do *pipeline*. Em *widgets* do tipo *dropdown menu* e *suggestion box*, seus subcomponentes tendem a ter dimensões similares, consequentemente, esta característica pode ser usada para encontrar elementos HTML que tenham os tamanhos de sua largura e a altura próximos do valor médio;

- **Desvio padrão das dimensões dos elementos HTML dos nós filhos:** representa o desvio padrão das dimensões (largura e altura separadamente) de todos elementos HTML inseridos em cada descendente direto de cada item que pertence ao *widget* que foi identificado pelo primeiro classificador do *pipeline*. Como os itens pertencentes a um *dropdown menu* ou a um *suggestion box* geralmente possuem dimensões iguais ou similares, o desvio padrão dos seus tamanhos tendem a estar próximos do valor zero;

- **Posição média dos elementos HTML dos nós filhos:** é a posição média de todos elementos HTML inseridos em cada descendente direto de cada item que pertence ao *widget* que foi identificado pelo primeiro classificador do *pipeline*. Esses valores são calculados separadamente em relação a borda esquerda e superior da página web. Dependendo da orientação vertical ou horizontal de um *widget* do tipo *dropdown menu*, os seus subcomponentes tendem a ter os mesmos valores para o eixo x ou eixo y. Para o caso do *suggestion box*, seus itens tendem a ter o mesmo valor para o eixo x, visto que normalmente esse componente apresenta seus textos sugestivos um abaixo do outro;

- **Desvio padrão das posições dos elementos HTML dos nós filhos:** é o desvio padrão das posições de todos elementos HTML inseridos em cada descendente direto de cada item que pertence ao *widget* que foi identificado pelo primeiro classificador do *pipeline*. Esses valores são calculados separadamente em relação a borda esquerda e superior da página web. Em um *widget* do tipo *dropdown menu* com orientação horizontal, o desvio padrão dos elementos em relação ao eixo y tende a zero porque todos os itens estão na mesma posição em relação a borda superior da página web. Por outro lado, na orientação vertical todos os itens ficam posicionados sob a mesma distância em relação a borda esquerda da página web, logo o desvio padrão das posições em relação ao eixo x tendem ao valor zero. Em um *widget* do tipo *suggestion box* o desvio padrão dos seus itens em relação ao

eixo *x* tendem a zero. Esse fato ocorre devido a sua característica de organização visual ser similar a de um *dropdown menu* com orientação vertical.

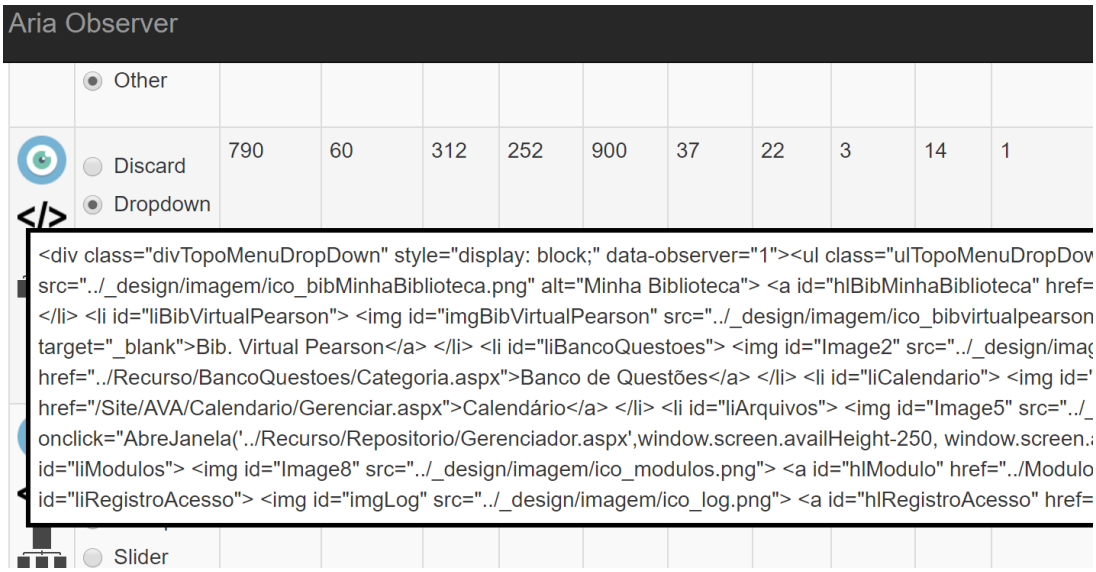

Figura 9: Exemplo de característica textual para auxiliar no processo de classificação manual dos *widgets*

É importante lembrar que o Logger (Seção 3.1.1) também coleta outras informações textuais das mutações ocorridas na estrutura DOM, no entanto, elas não são usadas como características para compor o conjunto de dados para treinamento do *pipeline* de classificação apresentado nesta dissertação. Essas características textuais são usadas para auxiliar a fase de classificação manual, que faz uso da ferramenta que foi implementada no Módulo Receptor (Seção 3.1.2) deste trabalho. (Figura 9).

3.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o *pipeline* de aprendizado de máquina e o conjunto de ferramentas criadas para materializar a abordagem de classificação automática de *widgets* e dos seus subcomponentes descrita nesta dissertação. A abordagem foca *widgets* do tipo *dropdown menu* e *suggestion box* e é composta pela estratégia de captura de *logs* das mutações ocorridas no DOM das páginas web (*Logger* e Módulo Receptor) e pelo *pipeline* de aprendizado de máquina.

O próximo capítulo apresenta uma avaliação dessa abordagem com objetivo de investigar a efetividade do *pipeline* de aprendizado de máquina para classificação automática de *widgets* e dos seus respectivos subcomponentes.

4 AVALIAÇÃO DA ABORDAGEM PROPOSTA

4.1 DEFINIÇÃO E EXECUÇÃO DO EXPERIMENTO

Para avaliação da abordagem proposta nesta dissertação de mestrado foi conduzido um estudo experimental para análise da sua efetividade e para investigar a questão de pesquisa lançada na subseção 1.2 deste trabalho. **Questão de Pesquisa:** *"Qual é a efetividade de um pipeline de aprendizado de máquina para classificação automática de widgets e seus subcomponentes com base em conjuntos de dados extraídos das alterações que ocorrem na estrutura DOM durante as interações dos usuários em RIAs?"*

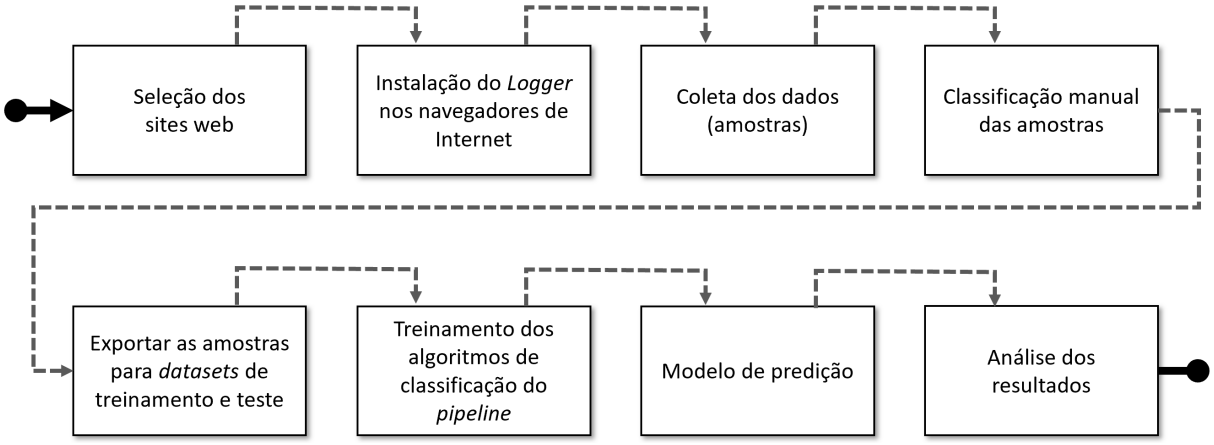

Figura 10: Etapas do estudo experimental

A condução do estudo experimental seguiu as etapas indicadas na Figura 10. A primeira etapa, caracterizada pela seleção dos sites web que serviram como base para a coleta das amostras, ocorreu por meio da observação dos 50 sites mais acessados nos Estados Unidos, segundo o portal Alexa[1]. Em cada um dos 50 sites contidos no relatório, avaliou-se o teor do seu conteúdo e também a presença ou não de *widgets* do tipo *dropdown menu* e/ou *suggestion box* em suas respectivas páginas iniciais. Os sites que apresentaram conteúdo destinado a adultos e aqueles que não possuíam os tipos de *widgets* que são objetos do estudo desta dissertação foram excluídos da lista de sites selecionados. Os sites de conteúdo adulto foram evitados pois

[1]https://www.alexa.com/topsites/countries/US - Data relatório: 14/03/2019 às 21h00

o estudo foi conduzido em ambiente acadêmico e o seu uso poderia causar inconvenientes. Já os sites que não possuíam os tipos de *widgets* alvo desta pesquisa foram excluídos por não disponibilizarem dados aderentes ao estudo. A Tabela 1 apresenta o resultado da seleção.

Tabela 1: Sites selecionados para coleta de amostras

Posição	**Site**	**Selecionado**	**Motivo**
01	Google.com	Sim	-
02	Youtube.com	Sim	-
03	Facebook.com	Sim	-
04	Amazon.com	Sim	-
05	Wikipedia.org	Sim	-
06	Reddit.com	Sim	-
07	Twitter.com	Sim	-
08	Yahoo.com	Sim	-
09	Linkedin.com	Sim	-
10	Ebay.com	Sim	-
11	Instagram.com	Sim	-
12	Netflix.com	Sim	-
13	Twitch.tv	Sim	-
14	Microsoftonline.com	Não	Problemas de conexão com o site.
15	Instructure.com	Não	Não há *dropdown menu* ou *suggestion box*
16	Pornhub.com	Não	Site com conteúdo para adultos
17	Live.com	Não	Não há *dropdown menu* ou *suggestion box*
18	Craigslist.org	Não	Não há *dropdown menu* ou *suggestion box*
19	Imgur.com	Sim	-
20	Chase.com	Sim	-
21	Paypal.com	Sim	-
22	Cnn.com	Sim	-
23	T.co	Não	Não há *dropdown menu* ou *suggestion box*
24	Espn.com	Sim	-
25	Bing.com	Sim	-
26	Imdb.com	Sim	-
27	Pinterest.com	Sim	-
28	Fandom.com	Não	Não há *dropdown menu* ou *suggestion box*
29	Office.com	Sim	-

Tabela 1 continuação da página anterior

Posição	Site	Selecionado	Motivo
30	Nytimes.com	Sim	-
31	Github.com	Sim	-
32	Microsoft.com	Sim	-
33	Livejasmin.com	Não	Site com conteúdo para adultos
34	Stackoverflow.com	Não	Não há *dropdown menu* ou *suggestion box*
35	Salesforce.com	Sim	-
36	Zillow.com	Sim	-
37	Force.com	Não	Redirecionado para a URL salesforce.com
38	Hulu.com	Não	Não há *dropdown menu* ou *suggestion box*
39	Weather.com	Sim	-
40	Intuit.com	Sim	-
41	Yelp.com	Sim	-
42	Apple.com	Sim	-
43	Bankofamerica.com	Não	Não há *dropdown menu* ou *suggestion box*
44	Quizlet.com	Não	Não há *dropdown menu* ou *suggestion box*
45	Walmart.com	Sim	-
46	Tumblr.com	Não	Não há *dropdown menu* ou *suggestion box*
47	Quora.com	Não	Não há *dropdown menu* ou *suggestion box*
48	Dropbox.com	Não	Não há *dropdown menu* ou *suggestion box*
49	Wellsfargo.com	Sim	-
50	Amazonaws.com	Sim	-
Sites selecionados: 34			
Sites não selecionados: 16			

Após a etapa de seleção dos sites web, o *Logger*, descrito na subseção 3.1.1, foi instrumentalizado com a lista dos 34 sites web selecionados para obtenção das amostras e instalado em um computador equipado com o sistema operacional Windows 10 e com o navegador de Internet Google Chrome (Versão 68.0.3440.106). Na sequência, o pesquisador acessou 5 (cinco) vezes a página inicial de cada site selecionado e interagiu com todos os *widgets* presentes em cada uma delas. As mudanças na estrutura HTML (mutações no DOM), ocorridas durante a interação do pesquisador nas páginas iniciais dos sites web selecionados, foram coletadas pelo *Logger* que, por sua vez, as enviou para o Módulo Receptor (Subseção

3.1.2) para organização e persistência dos dados.

A lista dos sites web selecionados para coleta das amostras foi dividida em 2 (dois) grupos, nomeados como **grupo A** e **grupo B**. Cada grupo foi composto por uma relação de sites web diferentes. Essa estratégia teve por objetivo construir conjuntos de dados distintos para serem usados no *pipeline* de aprendizado de máquina (Subseção 3.2), sendo o primeiro para treinamento (Tabela 2) e o segundo para teste (Tabela 3). Portanto, prevenindo vieses de similaridade dos dados para a abordagem, visto que foi garantido que os registros do conjunto de dados para treinamento não estão contidos nos registros do conjunto de dados para teste.

Tabela 2: Lista de sites web usados para compor o grupo A – coleta de amostras para treinamento (*TRAINING-DATASET*)

Posição	**Site**	**Mutações**	***Dropdown menu***	***Suggestion box***	**Outros**
03	Facebook.com	2752	0	256	2496
04	Amazon.com	2624	480	128	2016
05	Wikipedia.org	2816	1536	0	1280
06	Reddit.com	3776	1024	64	2688
07	Twitter.com	1684	100	144	1440
08	Yahoo.com	1760	352	128	1280
09	Linkedin.com	1488	256	160	1072
10	Ebay.com	2296	784	64	1448
11	Instagram.com	6	0	1	5
12	Netflix.com	1384	272	0	1112
26	Imdb.com	19	8	2	9
27	Pinterest.com	544	16	32	496
29	Office.com	108	1	0	107
30	Nytimes.com	864	136	0	728
31	Github.com	576	192	256	128
32	Microsoft.com	456	6	0	450
35	Salesforce.com	352	96	0	256
36	Zillow.com	288	0	128	160
39	Weather.com	1696	224	32	1440
40	Intuit.com	736	384	0	352
41	Yelp.com	296	48	80	168
42	Apple.com	424	32	0	392
45	Walmart.com	1304	224	48	1032

Tabela 2 continuação da página anterior

Posição	Site	Mutações	Dropdown menu	Suggestion box	Outros
49	Wellsfargo.com	608	320	0	288
50	Amazonaws.com	2540	11	0	2529

Tabela 3: Lista de sites web usados para compor o grupo B – coleta de amostras para teste (TEST-DATASET)

Posição	**Site**	**Mutações**	**Dropdown menu**	**Suggestion box**	**Outros**
01	Google.com	7168	896	128	6144
02	Youtube.com	175	0	0	175
13	Twitch.tv	688	64	0	624
19	Imgur.com	512	0	192	320
20	Chase.com	480	28	0	452
21	Paypal.com	544	224	0	320
22	Cnn.com	764	16	0	748
24	Espn.com	51	0	0	51
25	Bing.com	1568	272	32	1264

Após a coleta dos dados, foi dado início à etapa de classificação manual das amostras. Em primeiro lugar, os registros de mutação do DOM foram classificados como *widgets* do tipo *dropdown menu*, *suggestion box* ou outros. Na sequência, os registros relativos às mutações dos *widgets* do tipo *dropdown menu* e *suggestion box*, os subcomponentes, foram classificados como itens diretamente relacionados aos mesmos ou não. Para a classificação manual dos conjuntos de dados, foi utilizada a ferramenta especialmente desenvolvida para auxiliar o pesquisador nesse processo (Figura 5 e Figura 6) e que faz parte do Módulo Receptor descrito na subseção 3.1.2.

É importante ressaltar que, durante a etapa de classificação manual, foram identificados diferentes tipos de *widgets* (Exemplo: *sliders*, *tooltips*, calendários, e outros elementos HTML relativos a efeitos visuais nas páginas web) entre os registros de mutações do DOM que não foram classificados como *dropdown menu* ou *suggestion box*. Para os subcomponentes que não foram classificados como "item do *dropdown menu*" ou como "item do *suggestion box*", foram identificados elementos HTML como contêineres, recursos visuais e definições para responsividade inerentes ao uso em dispositivos móveis.

Com base nas amostras coletadas em cada grupo, foram gerados os conjuntos de dados para treinamento (*TRAINING-DATASET*) e para teste (*TEST-DATASET*). Cada conjunto de dados possui estrutura e registros diferentes para cada classificador do *pipeline*, assim, há uma estrutura e conjunto de dados para o classificador de *widgets* e outra estrutura e conjunto de dados para o classificador dos subcomponentes dos *widgets*. Como resultado da estratégia de organização dos dados, é possível avaliar a efetividade dos classificadores do *pipeline* separadamente (Figura 11).

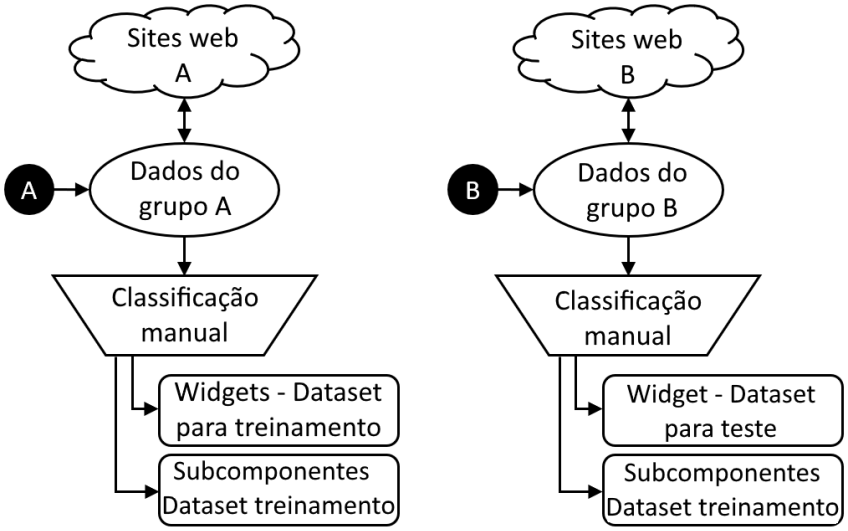

Figura 11: Método de geração dos conjuntos de dados

O conjunto de dados *TRAINING-DATASET* dos *widgets* e dos subcomponentes dos *widgets* foram submetidos separadamente para a etapa de treinamento do *pipeline* de classificação, fazendo uso do procedimento *10-fold cross-validation* (Figura 12) disponível na ferramenta Orange Tool (DEMSAR et al., 2013). Cada conjunto de dados foi processado por 5 (cinco) algoritmos diferentes de aprendizado de máquina para possibilitar a análise e avaliação de suas respectivas taxas de assertividade, com base nos resultados obtidos pelas métricas *F-measure*, *Precision* e *Recall* de cada um (POWERS, 2011).

Os algoritmos de aprendizado de máquina utilizados para processar os conjuntos de dados e uma breve descrição de suas características estão relacionados abaixo:

- *k-Nearest Neighbors* (kNN): O algoritmo kNN ou k-vizinhos mais próximos faz uso de medidas de distância, também conhecidas como similaridade, para identificar a qual classe uma determinada amostra pertence. O treinamento do classificador ocorre por meio de um conjunto de amostras de treinamento que é formado por vetores n-dimensionais e cada amostra desse conjunto representa um ponto no espaço n-dimensional. Para determinar a classe de uma amostra que não pertença ao conjunto de treinamento, o

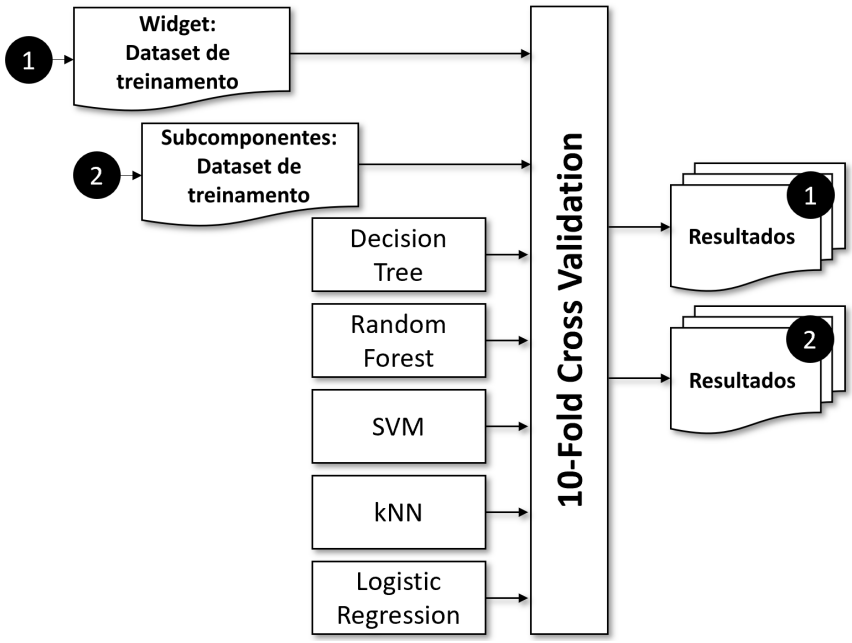

Figura 12: *10-fold cross-validation*: *Widgets* e Subcomponentes - TRAINING-DATASET

classificador kNN procura k amostras do conjunto de treinamento que estejam mais próximas (k-vizinhos mais próximos) da amostra desconhecida, ou seja, que tenham a menor distância. As métricas comumente encontradas na literatura para o cálculo de distância entre dois pontos são a distância Euclidiana, a distância Manhattan e a distância Minkowski (ZHANG et al., 2017);

- *Logistic Regression*: Baseia-se em uma técnica estatística que tem como objetivo produzir, a partir de um conjunto de dados, um modelo que permita a predição de valores tomados por uma variável categórica, frequentemente binária, a partir de uma série de variáveis explicativas contínuas e/ou binárias (HOSMER; LEMESHOW, 2000);

- *Support Vector Machine* (SVM): Caracteriza-se por métodos de aprendizado supervisionado que analisam conjuntos de dados e reconhecem padrões, com intuito de ser utilizado para classificação de amostras e análise de regressão. O SVM padrão infere sobre um conjunto de dados e prediz, para cada amostra, qual de duas possíveis classes ela pertence, o que caracteriza esse algoritmo como um classificador linear binário não probabilístico. Um modelo SVM é uma representação de amostras em um espaço divido por uma linha divisória também conhecida por hiperplano. As amostras são então mapeadas no mesmo espaço e preditas como pertencentes a uma categoria ou outra dependendo do lado do hiperplano em que elas são colocadas (HEARST, 1998);

- *Decision Tree*: Consiste em um mapa dos possíveis resultados de uma série de escolhas relacionadas. Por meio desse algoritmo comparam-se possíveis ações com base em seus

custos, probabilidades e benefícios, ou seja, é um algoritmo que matematicamente prevê a melhor escolha dentre as possibilidades contidas no conjunto de dados fornecido a ele. Uma árvore de decisão geralmente começa com um único nó, que se divide em possíveis resultados. Cada um desses resultados leva a nós adicionais, que se ramificam em outras possibilidades (QUINLAN, 1986);

- *Random Forest*: Trata-se de um algoritmo de classificação que faz uso do método de árvores de decisão. Porém, enquanto uma árvore de decisão possui o objetivo de construção total de uma estrutura a partir de uma base de dados, o Random Forest efetua a criação de múltiplas árvores de decisão usando um subconjunto de atributos selecionados aleatoriamente a partir do conjunto original. Essa estratégia pode aumentar sua capacidade de predição pois permite realizar votações ou mesmo médias de predições sobre as múltiplas árvores criadas (BREIMAN, 2001).

Após a etapa de treinamento, a acurácia do *pipeline* de aprendizado de máquina foi avaliada em um modelo de predição usando o conjunto de dados *TEST-DATASET* dos *widgets* e dos subcomponentes dos *widgets* (Figura 13). Finalmente, com os dados resultantes do modelo de predição foi possível compor matrizes de confusão que serviram como mecanismo de avaliação do comportamento dos classificadores e para observação do grau de assertividade entre as classes reais versus as classes preditas pelo modelo.

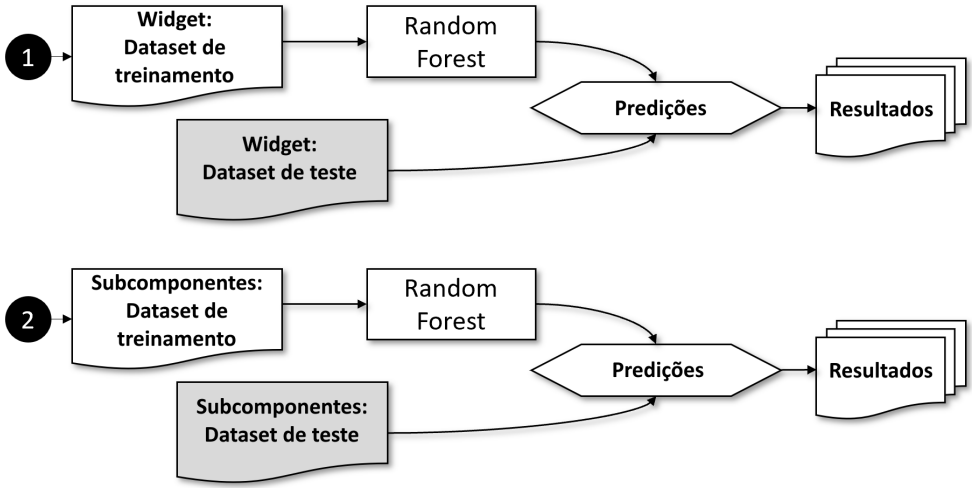

Figura 13: Modelo de predição: *Widgets* e Subcomponentes - TEST-DATASET

4.2 SOBRE A AQUISIÇÃO DAS AMOSTRAS

Para o procedimento de aquisição das amostras, o *Logger* foi instrumentalizado para coletar registros de mutações do DOM de 34 sites web presentes no relatório dos 50 sites mais

acessados nos Estados Unidos, segundo o Portal Alexa[2] (Tabela 1).

Do total dos sites analisados, 70% deles foram utilizados para composição do conjunto de dados *TRAINING-DATASET* dos *widgets* e 30% para composição do conjunto de dados *TEST-DATASET* dos *widgets*. O resultado da separação dos dois conjuntos pode ser observado nas Tabelas 2 e 3 respectivamente.

A separação do total das amostras coletadas das páginas HTML analisadas, proveu para o conjunto de dados *TRAINIG-DATASET*, 31397 registros de alterações em suas respectivas estruturas DOM (mutações) e 167461 registros de subcomponentes (elementos HTML) pertencentes nas mesmas. Esses registros de mutações foram classificados manualmente como *widgets* do tipo *dropdown menu* e seus respectivos subcomponentes, *widgets* do tipo *suggestion box* e seus respectivos subcomponentes, ou classificados como outros tipos de *widgets*. Ao final do processo de classificação foram obtidos os seguintes dados para o *TRAINING-DATASET* dos *widgets*:

- 6495 *widgets* do tipo *dropdown menu*;
- 1522 *widgets* do tipo *suggestion box*;
- 23380 *widgets* classificados como "outros". As amostras classificadas como "outros" são aquelas que não foram selecionadas como *widgets* do tipo *dropdown menu* ou do tipo *suggestion box*. Dentre elas, há *widgets* do tipo *slider*, calendário, *tooltip*, *label*, *banner* e outros.

A Figura 14 apresenta um trecho de código HTML que irá renderizar um *widget* do tipo *dropdown menu* no navegador web. Por ela, é possível observar o critério utilizado neste estudo para classificar uma mutação como um determinado tipo de *widget*, e dele, os seus respectivos subcomponentes.

As amostras direcionadas para o conjunto de dados *TEST-DATASET*, proveram 11950 registros de alterações em suas respectivas estruturas DOM (mutações) e 30224 registros de subcomponentes (elementos HTML) pertencentes às mesmas. Assim como feito para o *TRAINING-DATASET*, esses registros de mutações foram classificados manualmente como *widgets* do tipo *dropdown menu* e seus respectivos subcomponentes, *widgets* do tipo *suggestion box* e seus respectivos subcomponentes, ou classificados como outros tipo de *widgets*. Ao final do processo de classificação foram obtidos os seguintes dados para o *TEST-DATASET* dos *widgets*:

[2]https://www.alexa.com/topsites/countries/US - Data relatório: 14/03/2019 às 21h00

Figura 14: Organização do código HTML de um *widget* e dos seus subcomponentes

- 1500 *widgets* do tipo *dropdown menu*;
- 352 *widgets* do tipo *suggestion box*;
- 10098 *widgets* classificados como "outros". As amostras classificadas como "outros" são aquelas que não foram selecionadas como *widgets* do tipo *dropdown menu* ou do tipo *suggestion box*. Dentre elas, há *widgets* do tipo *slider*, calendário, *tooltip*, *label*, *banner* e outros.

Cabe ressaltar que as amostras relativas à classificação dos subcomponentes dos *widgets* foram organizadas em conjuntos de dados distintos. Tal procedimento foi realizado pois o segundo classificador do *pipeline* de aprendizado de máquina, proposto nesta dissertação, entra em operação apenas quando o primeiro classificador já identificou um determinado tipo de *widget*. Assim, caso o primeiro classificador tenha identificado um *widget* do tipo *dropdown menu*, então o segundo classificador do mesmo *pipeline* deve analisar apenas as amostras relativas aos subcomponentes referentes ao tipo do *widget* em questão.

A seguir, é apresentada a relação dos conjuntos de dados dos subcomponentes dos *widgets* e a quantidade de amostras classificadas como pertencentes ao tipo do *widget* identificado pelo primeiro classificador:

- *TRAINING-DROPDOWN-SUBCOMPONENTS*: representa as amostras para treinamento dos classificadores de aprendizado de máquina para os subcomponentes dos *widgets* do

tipo *dropdown menu*. Das 113367 amostras, 30245 são subcomponentes diretos;

- *TRAINING-SUGGESTION-SUBCOMPONENTS*: assim como o conjunto de dados do item anterior, esse conjunto de dados representa as amostras para treinamento dos classificadores de aprendizado de máquina para os subcomponentes dos *widgets* do tipo *suggestion box*. Das 54094 amostras, 24719 são subcomponentes diretos;
- *TEST-DROPDOWN-SUBCOMPONENTS*: amostras de subcomponentes de *widgets* do tipo *dropdown menu* a serem utilizadas no modelo de predição dos subcomponentes. Das 17520 amostras, 7715 são subcomponentes diretos;
- *TEST-SUGGESTION-SUBCOMPONENTS*: amostras de subcomponentes de *widgets* do tipo *suggestion box* a serem utilizadas no modelo de predição dos subcomponentes. Das 12704 amostras, 4803 são subcomponentes diretos.

As Tabelas 4 e 5 apresentam um resumo de todos os conjuntos de dados gerados pelo Módulo Receptor (Seção 3.1.2), dos tipos das amostras contidas em cada um deles e das suas respectivas quantidades:

Tabela 4: Resumo dos conjuntos de dados dos *widgets*: Treinamento e teste dos algoritmos do primeiro classificador do *pipeline* de aprendizado de máquina

Conjunto de Dados	**Total de amostras**	***Widgets* do tipo *Dropdown menu***	***Widgets* do tipo *Suggestion box***	**Outros *widgets***
TRAINING-DATASET	31397	6495	1522	23380
TEST-DATASET	11950	1500	352	10098

Tabela 5: Resumo dos conjuntos de dados dos subcomponentes dos *widgets*: Treinamento e teste dos algoritmos do segundo classificador do *pipeline* de aprendizado de máquina

Conjunto de Dados	**Total de amostras**	**É subcomponente do *widget***	**Não é subcomponente do *widget***
TRAINING DROPDOWN-SUBCOMP.	113367	30245	83122
TEST DROPDOWN-SUBCOMP.	17520	7715	9805
TRAINING SUGGESTION-SUBCOMP.	54094	24719	29375

TEST SUGGESTION-SUBCOMP.	12704	4803	7901

4.3 ANÁLISE DOS RESULTADOS

A efetividade da abordagem desta dissertação foi avaliada em fases por meio dos resultados obtidos pela ferramenta *Orange Tool* (ORANGE, 2013), sobre os conjuntos de dados já descritos na seção 4.2. As análises foram realizadas com intuito de responder a questão de pesquisa proposta, usando como critério as métricas *Precision*, *Recall* e *F-measure*, definidas no trabalho de POWERS (2011) e elencadas como seguem:

- ***Precision***: a métrica *Precision* representa a proporção dos casos previstos como verdadeiros positivos (*True Positives*) pelo classificador, e que de fato o são, em relação a soma dos casos verdadeiros positivos com os falsos positivos (*False Positives*) também identificados pelo classificador. O cálculo desta métrica é definido como:

$$Precision = \frac{TP}{TP + FP}$$

  onde *TP* é o número de casos previstos como verdadeiros positivos e que realmente são positivos e *FP* é o número de casos previstos como positivos, mas que de fato não são.

- ***Recall***: esta métrica representa a proporção dos casos verdadeiros positivos e que de fato o são, em relação a soma dos casos verdadeiros positivos com os falsos negativos. O calculo desta métrica é definido como:

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

  onde *TP* é o número de casos previstos como verdadeiros positivos e que realmente são positivos e *FN* é o número de casos previstos como falsos negativos (*False Negatives*), ou seja, os casos que deveriam ter sido classificados como positivos, mas foram classificados como negativos.

- ***F-measure***: é o meio harmônico (*Harmonic Mean*) entre a métrica *Precision* e a métrica *Recall*. Esse valor pode ser usado como uma comparação entre a classificação manual feita pelo humano em relação a classificação automática feita pelo algoritmo de aprendizado de máquina. A fórmula desta métrica é definida como:

$$Fmeasure = \frac{2 * Precision * Recall}{Precision + Recall}$$

A **primeira fase** da avaliação dos resultados foi mensurada pelas métricas obtidas dos valores referentes ao procedimento *10 fold cross-validation*. Ela foi realizada com base no conjunto de dados de treinamento (*TRAINING-DATASET*) dos *widgets*, submetendo-o aos algoritmos de aprendizado de máquina *Decision Tree*, *Support Vector Machine (SVM)*, *k-Nearest Neighbors (kNN)*, *Random Forest* e *Logistic Regression*. Nesta fase também avaliou-se a qualidade de classificação de cada algoritmo em relação aos alvos desejados, ou seja, a identificação de *widgets* do tipo *dropdown menu*, *suggestion box* e outros tipos de *widgets* contidos entre as amostras do conjunto de dados em questão.

A Tabela 6 apresenta os valores das métricas referentes a análise dos registros do conjunto de dados de treinamento dos *widgets*, tendo como alvo as amostras do tipo *dropdown menu*. Nela, os algoritmos *Decision Tree* e *Random Forest* apresentaram-se como os mais eficientes para esse tipo de classificação. Na Tabela 7 verifica-se o mesmo padrão de métricas e análises, no entanto, o alvo para classificação foram as amostras dos *widgets* do tipo *suggestion box*, sendo que para esse caso os algoritmos com melhores posicionamentos foram o *Decision Tree* e o *SVM*. Já a Tabela 8 demonstra os resultados das métricas quando os classificadores tiveram como alvo os demais tipos de *widgets* existentes entre as amostras do conjunto de dados de treinamento, ou seja, os *widgets* que não foram classificados como *dropdown menu* ou *suggestion box*. Para este caso, os algoritmos que apresentaram melhores resultados de classificação foram novamente o *Decision Tree* e o *Random Forest*.

Tabela 6: Resultados do procedimento *10 fold cross-validation* para o conjunto de dados de treinamento dos *widgets* (*TRAINING-DATASET*) - Alvo: *Widgets* do tipo *DROPDOWN MENU*

Algoritmo	***F-measure***	***Precision***	***Recall***
kNN	0.339	0.693	0.224
Decision Tree	**0.973**	**0.948**	**1.000**
SVM	0.817	0.862	0.776
Random Forest	**0.968**	**0.939**	**1.000**
Logistic Regression	0.839	0.873	0.808

O mesmo procedimento de análise da classificação dos algoritmos *Decision Tree*, *Support Vector Machine (SVM)*, *k-Nearest Neighbors (kNN)*, *Random Forest* e *Logistic Regression* foi realizado para o conjunto de dados de treinamento dos subcomponentes dos *widgets* do tipo *dropdown menu* e *suggestion box* (*TRAINIG-DROPDOWN-SUBCOMPONENTS* e *TRAINING-SUGGESTION-SUBCOMPONENTS*). O

Tabela 7: Resultados do procedimento *10 fold cross-validation* para o conjunto de dados de treinamento dos *widgets* (*TRAINING-DATASET*) - Alvo: Widgets do tipo *SUGGESTION BOX*

Algoritmo	*F-measure*	*Precision*	*Recall*
kNN	0.918	0.924	0.917
Decision Tree	**0.936**	**0.964**	**0.909**
SVM	**0.986**	**0.972**	**1.000**
Random Forest	0.920	0.930	0.909
Logistic Regression	0.752	0.918	0.638

Tabela 8: Resultados do procedimento *10 fold cross-validation* para o conjunto de dados de treinamento dos *widgets* (*TRAINING-DATASET*) - Alvo: *Widgets* do tipo OUTROS *WIDGETS*

Algoritmo	*F-measure*	*Precision*	*Recall*
kNN	0.925	0.875	0.982
Decision Tree	**0.994**	**0.997**	**0.991**
SVM	0.974	0.967	0.981
Random Forest	**0.988**	**0.997**	**0.980**
Logistic Regression	0.983	0.972	0.993

objetivo deste procedimento foi o de identificar quais amostras eram subcomponentes diretos dos *widgets* identificados pelo primeiro classificador do *pipeline* e quais não eram. Os resultados dessa nova análise, que corresponde a **segunda fase** da avaliação, podem ser observados nas Tabelas 9, 10, 11 e 12, onde é possível verificar que o algoritmo *Random Forest* apresentou melhores resultados para todas as combinações utilizadas, ou seja, para quando o alvo de classificação eram subcomponentes de *widgets* do tipo *dropdown menu* e subcomponentes de *widgets* do tipo *suggestion box*.

Tabela 9: Resultados do procedimento 10 fold cross-validation para o conjunto de dados de treinamento dos subcomponentes dos *widgets* (TRAINING-DROPDOWN-SUBCOMPONENTS) - Alvo: É SUBCOMPONENTE de *widgets* do tipo *Dropdown menu*

Algoritmo	*F-measure*	*Precision*	*Recall*
kNN	0.432	0.782	0.299
Decision Tree	0.340	0.836	0.214
SVM	0.071	0.667	0.037
Random Forest	**0.875**	**0.947**	**0.813**
Logistic Regression	0.319	0.874	0.195

A soma dos resultados apresentados pelas métricas dos classificadores, que compuseram o *pipeline* de aprendizado de máquina proposto nesta dissertação, forneceu evidências para sustentar a **Questão de Pesquisa** levantada. As evidências obtidas sugerem a efetividade do *pipeline* para classificação automática de *widgets* do tipo *dropdown menu* e *suggestion box*, bem como seus respectivos subcomponentes. Além disso, elas também

Tabela 10: Resultados do procedimento 10 fold cross-validation para o conjunto de dados de treinamento dos subcomponentes dos *widgets* (TRAINING-DROPDOWN-SUBCOMPONENTS) - Alvo: NÃO É SUBCOMPONENTE de *widgets* do tipo *Dropdown menu*

Algoritmo	***F-measure***	***Precision***	***Recall***
kNN	0.752	0.629	0.934
Decision Tree	0.748	0.610	0.967
SVM	0.719	0.566	0.985
Random Forest	**0.913**	**0.868**	**0.964**
Logistic Regression	0.749	0.607	0.978

Tabela 11: Resultados do procedimento 10 fold cross-validation para o conjunto de dados de treinamento dos subcomponentes dos *widgets* (TRAINING-SUGGESTION-SUBCOMPONENTS) - Alvo: É SUBCOMPONENTE de *widgets* do tipo *Suggestion box*

Algoritmo	***F-measure***	***Precision***	***Recall***
kNN	0.897	0.842	0.960
Decision Tree	0.856	0.904	0.813
SVM	0.301	0.246	0.387
Random Forest	**0.914**	**0.908**	**0.920**
Logistic Regression	0.436	0.420	0.453

Tabela 12: Resultados do procedimento 10 fold cross-validation para o conjunto de dados de treinamento dos subcomponentes dos *widgets* (TRAINING-SUGGESTION-SUBCOMPONENTS) - Alvo: NÃO É SUBCOMPONENTE de *widgets* do tipo *Suggestion box*

Algoritmo	***F-measure***	***Precision***	***Recall***
kNN	0.930	0.973	0.891
Decision Tree	0.919	0.893	0.947
SVM	0.338	0.429	0.279
Random Forest	**0.947**	**0.951**	**0.943**
Logistic Regression	0.635	0.651	0.619

evidenciam que as características selecionadas para compor os conjuntos de dados de treinamento foram suficientes para que os algoritmos utilizados pudessem identificar as amostras obtidas na fase de aquisição dos dados (Seção 4.2).

A **terceira fase** da avaliação dos resultados levou em consideração os algoritmos que obtiveram as melhores métricas nos procedimentos *10 fold cross-validation* já demonstrados nas fases anteriores. Isso posto, os conjuntos de dados para teste, já identificados nas Tabelas 4 e 5, foram submetidos ao procedimento de predição (Figura 13) com objetivo de obter as matrizes de confusão resultantes da classificação dos *widgets* do tipo *dropdown menu* e *suggestion box*, bem como dos seus respectivos subcomponentes.

As matrizes de confusão representadas pelas Tabelas 13, 14 e 15, demonstram a assertividade dos classificadores *Decision Tree*, *Random Forest* e *SVM* para quando os alvos a serem classificados dentre as amostras de teste são *widgets* do tipo *dropdown menu*, *suggestion*

box e aqueles que não são dos dois tipos anteriores, ou seja, outros tipos de *widgets*.

Tabela 13: Predição: Matriz de confusão para o conjunto de teste dos *widgets* (TEST-DATASET) - Algoritmo: Decision Tree

		Preditos			
		Dropdown menu	**Outros widgets**	**Suggestion box**	Total
Atual	**Dropdown menu**	**1500 (100%)**	0	0	1500
	Outros widgets	82	**10004 (99,06%)**	12	10098
	Suggestion box	0	32	**320 (90,90%)**	352
	Total	1582	10036	332	11950

Tabela 14: Predição: Matriz de confusão para o conjunto de teste dos *widgets* (TEST-DATASET) - Algoritmo: Random Forest

		Preditos			
		Dropdown menu	**Outros widgets**	**Suggestion box**	Total
Atual	**Dropdown menu**	**1500 (100%)**	0	0	1500
	Outros widgets	98	**9976 (98,79%)**	24	10098
	Suggestion box	0	32	**320 (90,90%)**	352
	Total	1598	10008	344	11950

Tabela 15: Predição: Matriz de confusão para o conjunto de teste dos *widgets* (TEST-DATASET) - Algoritmo: SVM

		Preditos			
		Dropdown menu	**Outros widgets**	**Suggestion box**	Total
Atual	**Dropdown menu**	**1164 (77,60%)**	336	0	1500
	Outros widgets	186	**9902 (98,05%)**	10	10098
	Suggestion box	0	0	**352 (100%)**	352
	Total	1350	10238	362	11950

Essas matrizes de confusão corroboram as métricas obtidas pelo procedimento *10 fold cross-validation* realizado com o conjunto de dados de treinamento dos *widgets* e permitem quantificar a assertividade de cada classificador quando exigida a predição sobre os conjuntos de dados de teste. Também evidenciam que o algoritmo de classificação *Decision Tree*, em média, apresentou-se como elegível para ser o primeiro classificador do *pipeline* de aprendizado de máquina, para classificação automática de *widgets* e dos seus subcomponentes, proposto neste estudo.

Ainda como parte das análises da **terceira fase** da avaliação dos resultados, também foram observadas as matrizes de confusão (Tabelas 16 e 17) para o segundo classificador do *pipeline* de aprendizado de máquina, ou seja, o classificador que identifica os subcomponentes dos *widgets*. As matrizes de confusão do classificador dos subcomponentes dos *widgets* sugerem que o algoritmo *Random Forest*, que obteve as melhores métricas no procedimento

10 fold cross-validation, é elegível para atuar como segundo classificador do *pipeline* de classificação automática de *widgets* deste estudo.

Tabela 16: Predição: Matriz de confusão para o conjunto de teste dos subcomponentes dos *widgets* (TEST-DROPDOWN-SUBCOMPONENTS) - Algoritmo: Random Forest - Alvo: SUBCOMPONENTES de *widgets* do tipo *Dropdown menu*

		Preditos		
		Não Subcomp.	**Subcomp.**	Total
Atual	**Não Subcomp.**	**9456 (96,41%)**	352	9808
	Subcomp.	1140	**6272 (81,32%)**	7712
	Total	10896	6624	17520

Tabela 17: Predição: Matriz de confusão para o conjunto de teste dos subcomponentes dos *widgets* (TEST-SUGGESTION-SUBCOMPONENTS) - Algoritmo: Random Forest - Alvo: SUBCOMPONENTES de *widgets* do tipo *Suggestion box*

		Preditos		
		Não Subcomp.	**Subcomp.**	Total
Atual	**Não Subcomp.**	**7456 (94,33%)**	448	7904
	Subcomp.	384	**4416 (92,00%)**	4800
	Total	7840	4864	12704

Em geral, a métrica *F-measure* do classificador dos *widgets* e do classificador dos subcomponentes dos *widgets*, para os conjuntos de dados de teste foi de **0,967** e **0,894** respectivamente. Esses valores sugerem que a abordagem desta dissertação de mestrado pode ser considerada como plausível para a identificação de *widgets* do tipo *dropdown menu* e *suggestion box*, bem como dos seus respectivos subcomponentes. Além disso, essa abordagem pode ser considerada como um recurso para o desenvolvimento de ferramentas automáticas para avaliação de conformidade de RIAs para com as regras da ARIA. Também é importante ressaltar que os conjuntos de dados de teste são compostos apenas por dados que não estão compreendidos nos conjuntos de dados para treinamento, o que reduz possíveis vises no contexto dessa análise.

4.4 DISCUSSÃO DOS RESULTADOS

O estudo experimental relatado avaliou a abordagem proposta nesta dissertação para classificação automática de *widgets* do tipo *dropdown menu* e *suggestion box*, bem como dos seus respectivos subcomponentes, por meio do uso de um *pipeline* de aprendizado de máquina. Os resultados obtidos sugerem viabilidade e efetividade do estudo, no entanto, para generalizar os resultados novas pesquisas para outros tipos de *widgets* é algo a ser realizado.

Apesar do *Logger* (Seção 3.1.1) poder coletar todos os registros de mutações do DOM das páginas de quaisquer sites web, o *pipeline* de aprendizado de máquina proposto (Seção 3.2) foi treinado exclusivamente para a classificação de *widgets* do tipo *dropdown menu* e *suggestion box*, bem como dos subcomponentes que os compõem. Adicionalmente, o *Logger* foi instrumentalizado para coletar dados de sites web específicos (Tabelas 2 e 3) para o estudo experimental desta dissertação de mestrado. Assim sendo, antes que os resultados possam ser generalizados para outros tipos de *widgets*, é essencial a condução de um novo estudo de caso para este fim. Por exemplo, para classificar um *widget* do tipo "Calendário" (Figura 1) e seus subcomponentes, o *pipeline* de aprendizado de máquina proposto precisa ser treinado para essa nova classe de *widget*, validar se o conjunto de caracterísicas que compõem atualmente os conjuntos de dados são condizentes para a nova situação e por fim, verificar a qualidade de suas métricas de predição.

O estudo experimental realizado fez uso de conjuntos de dados de teste imparciais para a predição e obtenção dos resultados da avaliação do *pipeline* de aprendizado de máquina, ou seja, os dados contidos nos conjuntos de dados de teste não participaram do processo de treinamento dos algoritmos de aprendizado de máquina utilizados pelo *pipeline* (Figuras 11, 12 e 13). A condução do estudo experimental foi executada desta forma para evitar vieses de tendência de assertividade no modelo de aprendizado de máquina, ocasionados por dados de treinamento e teste não heterogêneos ou por indução na seleção dos dados/características.

Além dos resultados favoráveis apresentados pelo estudo, também foram percebidas ocorrências de falsos positivos nas matrizes de confusão utilizadas como parte da avaliação dos resultados, sendo assim, estudos futuros são necessários para aprimorar a estratégia de classificação automática de *widgets* e dos seus subcomponentes.

4.5 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado um estudo experimental para avaliar a efetividade da abordagem de classificação automática de *widgets* e dos seus subcomponentes, por meio de um *pipeline* de aprendizado de máquina. Foram coletados dados de 34 dos 50 sites mais acessados nos Estados Unidos (Tabela 1) fazendo uso das ferramentas implementadas especialmente para a condução deste trabalho.

Os resultados obtidos no processo de avaliação deste estudo, sugerem que o *pipeline* de aprendizado de máquina bem como as características das mutações do DOM utilizadas para composição dos conjuntos de dados, tando dos *widgets* quanto dos seus subcomponentes, deram

suporte para responder a Questão de Pesquisa. Eles também sugerem que o *pipeline* proposto pode contribuir para o desenvolvimento de ferramentas de avaliação de conformidade de RIAs para com as regras da ARIA ou mesmo para o desenvolvimento de ferramentas que façam a adaptação automática de código HTML para os padrões de acessibilidade propostos pela ARIA.

O capítulo a seguir conclui esta dissertação, reforçando as principais contribuições, limitações e possibilidades para trabalhos futuros. Também apresenta uma seção com artigos já publicados na comunidade científica e *links* para *download* da ferramenta ARIA Observer e dos conjuntos de dados utilizados no estudo experimental desta dissertação.

5 CONCLUSÃO

A especificação ARIA detalha uma ontologia com base em atributos de regras, estados e propriedades para propiciar semântica aos elementos HTML dos sites web. A inclusão desses atributos em elementos genéricos do HTML, como por exemplo, DIVs, ULs, SPANs e outros, provê informação adicional às Tecnologias Assistivas, o que as tornam mais eficientes e possibilitam que pessoas com deficiência interajam apropriadamente com as RIAs (W3C, 2017a). Embora a especificação ARIA esteja disponível e muitos esforços venham sendo feitos para sua popularização perante a comunidade dos desenvolvedores web, os requisitos da ARIA não estão sendo considerados à contento no processo de desenvolvimento de muitos sites web (FREIRE et al., 2008b, 2008a; WATANABE et al., 2014).

A não inclusão de informações adicionais sobre semântica, proporcionada pela ontologia da ARIA nos sites web, reforça a necessidade de disponibilizar ferramentas de suporte para os desenvolvedores com intuito de alertar possíveis falhas e facilitar a implementação de RIAs acessíveis (ANTONELLI et al., 2018). Nesse sentido, esta dissertação apresentou uma abordagem que propõe o uso de um *pipeline* de aprendizado de máquina para classificar automaticamente *widgets* do tipo *dropdown menu* e *suggestion box*, bem como os seus respectivos subcomponentes. A abordagem baseia-se na coleta de mutações do DOM em RIAs e, para tanto, foram selecionados 34 dos 50 sites web mais acessados nos EUA segundo o portal de informações Alexa. As amostras coletadas foram organizadas em conjuntos de dados para treinamento do *pipeline* de classificação e teste de sua efetividade. Uma ferramenta chamada **Aria Observer** foi desenvolvida para apoiar a abordagem e estratégias de captura, persistência e geração dos conjuntos de dados.

No processo de avaliação, os resultados evidenciaram que o conjunto de classificadores utilizados no *pipeline* foram capazes de identificar corretamente os *widgets* alvo deste trabalho, bem como sugeriram que as características utilizadas para composição dos conjuntos de dados submetidos ao aprendizado de máquina foram suficientes para proporcionar assertividade aos algoritmos de classificação, no entanto, mostraram-se mais efetivos na identificação dos *widgets* do que dos seus respectivos subcomponentes.

A abordagem nesta dissertação não resulta em uma ferramenta a ser utilizada diretamente pelos desenvolvedores web, no entanto, introduz uma estratégia para classificar automaticamente *widgets* e seus respectivos subcomponentes. Indiretamente, o resultado deste estudo vem a contribuir com a acessibilidade web, no sentido de que o mesmo pode ser utilizado para o desenvolvimento de ferramentas de avaliação automática de RIAs em detrimento as suas respectivas conformidades com as regras da ARIA, ou mesmo em ferramentas para adaptação automática do código HTML de sites web, de acordo com as regras da ARIA, para torná-los acessíveis.

5.1 LIMITAÇÕES E TRABALHOS FUTUROS

Abrangência: Este estudo descreveu uma abordagem para classificação automática de *widgets* e dos seus subcomponentes a partir de uma amostra composta por 34 dos 50 sites mais acessados nos Estados Unidos, segundo o portal de informações Alexa, consequentemente, não é possível estender os resultados obtidos para todos os outros sites da Internet.

Generalização: As amostras foram analisadas com objetivo de classificar apenas dois tipos específicos de *widgets*, que foram os do tipo *dropdown menu* e *suggestion box*. Dessa forma, há que se ampliar o estudo para validar sua eficiência para outros tipos de *widgets*, como por exemplo, calendários, *sliders*, *tooltips*, entre outros.

***Logger*:** O *plugin* desenvolvido para o navegador Google Chrome foi instrumentalizado para coletar apenas uma amostra de cada mutação ocorrida no DOM de uma determinada página web, dessa forma, caso uma mesma mutação DOM tenha ocorrido mais de um vez e tenha gerado dados diversos ao da primeira ocorrência, os mesmos não foram analisados pelo *pipeline* de classificação.

Classificação manual: O trabalho de classificação manual das amostras para treinamento do aprendizado de máquina não foi revisado por nenhum outro indivíduo, o que aumenta a possibilidade dos conjuntos de dados resultantes conterem amostras classificadas incorretamente.

Aplicação real: Um novo estudo poderia fazer uso da proposta desta dissertação para implementar o *pipeline* de aprendizado de máquina em uma aplicação real. Os resultados poderiam possibilitar a avaliação da aplicabilidade da abordagem.

5.2 DIVULGAÇÃO DOS RESULTADOS

Durante o período de pesquisa e desenvolvimento desta dissertação de mestrado, alguns resultados parciais foram publicados em conferências internacionais. São eles:

- RIZO, E. H. **Classification pipeline for automatic identification of widgets and their parts**. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. Pau, France: ACM, 2018 (SAC '18). ISBN: 978-1-4503-5191-1. Disponível em: https://doi.acm.org/10.1145/3167132.3167460

- RIZO, E. H. et al. **Automatic identification of widgets and their subcomponents based on a classification pipeline for DOM mutation records**: In: Proceedings of the 16th International Web for All Conference: Addressing Information Barriers. San Francisco, CA, USA: ACM, 2019 (W4A '19). ISBN 978-1-4503-6716-5. Disponível em: https://doi.org/10.1145/3315002.3317555

Estão disponíveis para download os conjuntos de dados das amostras utilizadas para treinamento e teste do *pipeline* de classificação e também a ferramenta **ARIA Observer**, que é composta pelo *Logger* (Subseção 3.1.1), pelo Módulo Receptor (Subseção 3.1.2) e pelo Pipeline para classificação automática de *widgets* e dos seus subcomponentes (Seção 3.2). O *pipeline* de classificação foi implementado sob a versão 3.20 da ferramenta Orange Tool.

- Download do *Logger* (Plugin do navegador Google Chrome): https://bitbucket.org/ehrizo/ariaobserver/src/master/Plugin-Logger/

- Download do Módulo Receptor: https://bitbucket.org/ehrizo/ariaobserver/src/master/Tool/

- Download dos conjuntos de dados e do pipeline de classificação: https://bitbucket.org/ehrizo/ariaobserver/src/master/DataSets/

REFERÊNCIAS

ABNT. **Norma Brasileira ABNT NBR 9050**. 2015. Acessibilidade a edificações, mobiliário, espaços e equipamentos urbanos. [Obtido em 25 de Setembro de 2018]. Disponível em: <http://www.ufpb.br/cia/contents/manuais/abnt-nbr9050-edicao-2015.pdf>.

ANTONELLI, H. L. et al. Drop-down menu widget identification using html structure changes classification. **ACM Trans. Access. Comput.**, ACM, New York, NY, USA, v. 11, n. 2, p. 10:1–10:23, jun. 2018. ISSN 1936-7228. Disponível em: <http://doi.acm.org/10.1145/3178854>.

BRASIL. **Decreto no 5.296/2004**. 2004. Regulamenta as Leis nos 10.048, de 8 de novembro de 2000, que dá prioridade de atendimento às pessoas que especifica, e 10.098, de 19 de dezembro de 2000, que estabelece normas gerais e critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência ou com mobilidade reduzida, e dá outras providências. [Obtido em 25 de Setembro de 2018]. Disponível em: <http://www.planalto.gov.br/ccivil-03/-Ato2004-2006/2004/Decreto/D5296.htm>.

BRASIL. **Decreto no 6.949/2009**. 2009. Promulga a Convenção Internacional sobre os Direitos das Pessoas com Deficiência e seu Protocolo Facultativo, assinados em Nova York, em 30 de março de 2007. [Obtido em 25 de Setembro de 2018]. Disponível em: <http://www.planalto.gov.br/ccivil-03/-Ato2007-2010/2009/Decreto/D6949.htm>.

BREIMAN, L. Random forests. **Mach. Learn.**, Kluwer Academic Publishers, Norwell, MA, USA, v. 45, n. 1, p. 5–32, out. 2001. ISSN 0885-6125. Disponível em: <https://doi.org/10.1023/A:1010933404324>.

BROWN, A. et al. The uptake of web 2.0 technologies, and its impact on visually disabled users. **Universal Access in the Information Society**, v. 11, n. 2, p. 185–199, Jun 2012. ISSN 1615-5297. Disponível em: <https://doi.org/10.1007/s10209-011-0251-y>.

CARVALHO, L. P.; FERREIRA, L. P.; FREIRE, A. P. Accessibility evaluation of rich internet applications interface components for mobile screen readers. In: **Proceedings of the 31st Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2016. (SAC '16), p. 181–186. ISBN 978-1-4503-3739-7. Disponível em: <http://doi.acm.org/10.1145/2851613.2851680>.

CHEN, A. Q. Widget identification and modification for web 2.0 access technologies (wimwat). **SIGACCESS Access. Comput.**, ACM, New York, NY, USA, n. 96, p. 11–18, jan. 2010. ISSN 1558-2337. Disponível em: <http://doi.acm.org/10.1145/1731849.1731851>.

CHEN, A. Q. et al. Widget Identification: A High-Level Approach to Accessibility. **World Wide Web**, v. 16, n. 1, p. 73–89, jan 2013. ISSN 1386-145X. Disponível em: <http://link.springer.com/10.1007/s11280-012-0156-6>.

DEMSAR, J. et al. Orange: Data mining toolbox in python. **Journal of Machine Learning Research**, v. 14, p. 2349–2353, 2013. Disponível em: <http://jmlr.org/papers/v14/demsar13a.html>.

DOUSH, I. A. et al. The design of RIA accessibility evaluation tool. **Advances in Engineering Software**, v. 57, p. 1–7, mar 2013. ISSN 09659978. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0965997812001512>.

FRATERNALI, P. et al. Engineering rich internet applications with a model-driven approach. **ACM Trans. Web**, ACM, New York, NY, USA, v. 4, n. 2, p. 7:1–7:47, abr. 2010. ISSN 1559-1131. Disponível em: <http://doi.acm.org/10.1145/1734200.1734204>.

FREIRE, A. P.; BITTAR, T. J.; FORTES, R. P. M. An approach based on metrics for monitoring web accessibility in brazilian municipalities web sites. In: **Proceedings of the 2008 ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2008. (SAC '08), p. 2421–2425. ISBN 978-1-59593-753-7. Disponível em: <http://doi.acm.org/10.1145/1363686.1364259>.

FREIRE, A. P.; RUSSO, C.; FORTES, R. P. M. The perception of accessibility in Web development by academy, industry and government: a survey of the Brazilian scenario. **New Review of Hypermedia and Multimedia**, v. 14, n. 2, p. 149–175, dec 2008. ISSN 1361-4568. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/13614560802624241>.

GIBSON, B. Enabling an accessible web 2.0. In: **Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A) - W4A '07**. New York, New York, USA: ACM Press, 2007. p. 1–6. ISBN 159593590X. Disponível em: <http://portal.acm.org/citation.cfm?doid=1243441.1243442>.

GIBSON, B.; SCHWERDTFEGER, R. Dhtml accessibility: Solving the javascript accessibility problem. In: **Proceedings of the 7th International ACM SIGACCESS Conference on Computers and Accessibility**. New York, NY, USA: ACM, 2005. (Assets '05), p. 202–203. ISBN 1-59593-159-7. Disponível em: <http://doi.acm.org/10.1145/1090785.1090830>.

GIRAUD, S. et al. Accessibility of rich internet applications for blind people: A study to identify the main problems and solutions. In: **Proceedings of the 9th ACM SIGCHI Italian Chapter International Conference on Computer-Human Interaction: Facing Complexity**. New York, NY, USA: ACM, 2011. (CHItaly), p. 163–166. ISBN 978-1-4503-0876-2. Disponível em: <http://doi.acm.org/10.1145/2037296.2037335>.

GOOGLE. **Chrome Develop Extensions**. 2016. [Obtido em 07 de Fevereiro de 2017]. Disponível em: <https://developer.chrome.com/extensions/devguide/>.

HEARST, M. A. Support vector machines. **IEEE Intelligent Systems**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 13, n. 4, p. 18–28, jul. 1998. ISSN 1541-1672. Disponível em: <http://dx.doi.org/10.1109/5254.708428>.

HOSMER, D. W.; LEMESHOW, S. **Applied logistic regression**. [S.l.]: John Wiley and Sons, 2000. ISBN 0471356328, 9780471356325.

KELLY, B. et al. Accessibility 2.0: People, policies and processes. In: **Proceedings of the 2007 International Cross-disciplinary Conference on Web Accessibility (W4A)**. New York, NY, USA: ACM, 2007. (W4A '07), p. 138–147. ISBN 1-59593-590-8. Disponível em: <http://doi.acm.org/10.1145/1243441.1243471>.

KELLY, B. et al. Forcing standardization or accommodating diversity?: A framework for applying the wcag in the real world. In: **Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)**. New York, NY, USA: ACM, 2005. (W4A '05), p. 46–54. ISBN 1-59593-219-4. Disponível em: <http://doi.acm.org/10.1145/1061811.1061820>.

LUCCA, G. D.; FASOLINO, A. R.; TRAMONTANA, P. Web site accessibility: Identifying and fixing accessibility problems in client page code. In: **Seventh IEEE International Symposium on Web Site Evolution(WSE)**. [s.n.], 2005. v. 00, p. 71–78. ISSN 1550-4441. Disponível em: <doi.ieeecomputersociety.org/10.1109/WSE.2005.22>.

MDN MOZILLA WEB DOCS. **MutationObserver API**. 2012. [Obtido em 06 de Fevereiro de 2017]. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver/>.

MELNYK, V. et al. Look ma, no aria: Generic accessible interfaces for web widgets. In: **Proceedings of the 12th Web for All Conference**. New York, NY, USA: ACM, 2015. (W4A '15), p. 21:1–21:4. ISBN 978-1-4503-3342-9. Disponível em: <http://doi.acm.org/10.1145/2745555.2746666>.

MELNYK, V.; ASHOK, V.; PUZIS, Y. Widget Classification with Applications to Web Accessibility. **Web Engineering, Icwe 2014**, p. 341–358, 2014. ISSN 16113349. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-319-08245-5-20>.

MUNSON, E. V.; PIMENTEL, M. G. da. Specialized documents. In: ____. **Web Accessibility: A Foundation for Research**. London: Springer London, 2008. p. 274–285. ISBN 978-1-84800-050-6. Disponível em: <https://doi.org/10.1007/978-1-84800-050-6-16>.

ORANGE. **Open source machine learning and data visualization tool for novice and expert**. 2013. [Obtido em 16 de Novembro de 2017]. Disponível em: <https://orange.biolab.si/>.

PAGANELLI, L.; PATERNÒ, F. Intelligent analysis of user interactions with web applications. **Proceedings of the 7th international conference on Intelligent user interfaces - IUI '02**, p. 111–118, 2002. Disponível em: <http://dl.acm.org/citation.cfm?id=502735%5Cnhttp://dl.acm.org/citation.cfm?id=502716.502735>.

POWERS, D. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. **Journal of Machine Learning Technologies**, v. 2, n. 1, p. 37–63, 2011.

QUINLAN, J. R. Induction of decision trees. **Mach. Learn.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 1, n. 1, p. 81–106, mar. 1986. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A:1022643204877>.

REID, L. G.; SNOW-WEAVER, A. Wcag 2.0: A web accessibility standard for the evolving web. In: **Proceedings of the 2008 International Cross-disciplinary Conference on Web Accessibility (W4A)**. New York, NY, USA: ACM, 2008. (W4A '08), p. 109–115. ISBN 978-1-60558-153-8. Disponível em: <http://doi.acm.org/10.1145/1368044.1368069>.

RIZO, E. H. et al. Automatic identification of widgets and their subcomponents based on a classification pipeline for dom mutation records. In: **Proceedings of the 16th International Web for All Conference: Addressing Information Barriers**. San Francisco, CA, USA: ACM, 2019. (W4A '19). ISBN 978-1-4503-6716-5. Disponível em: <https://doi.org/10.1145/3315002.3317555>.

SANDHYA, S.; DEVI, K. A. S. Accessibility evaluation of websites using screen reader. **Proceedings of the 2011 7th International Conference on Next Generation Web Services Practices, NWeSP 2011**, p. 338–341, 2011.

SCHMIDT, K.-U. et al. An User Interface Adaptation Architecture for Rich Internet Applications. In: **The Semantic Web: Research and Applications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 736–750. Disponível em: <http://link.springer.com/10.1007/978-3-540-68234-9_53>.

THIESSEN, P.; HOCKEMA, S. Wai-aria live regions: Ebuddy im as a case example. In: **Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)**. New York, NY, USA: ACM, 2010. (W4A '10), p. 33:1–33:9. ISBN 978-1-4503-0045-2. Disponível em: <http://doi.acm.org/10.1145/1805986.1806030>.

VARGAS, A.; WEFFERS, H.; ROCHA, H. V. da. A method for remote and semi-automatic usability evaluation of web-based applications through users behavior analysis. **ACM Proceedings of the 7th International Conference on Methods and Techniques in Behavioral Research - MB '10**, p. 1–5, 2010. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-79952500928&partnerID=tZOtx3y1>.

VELASCO, C. a. et al. A web compliance engineering framework to support the development of accessible rich internet applications. **Proceedings of the 2008 international cross-disciplinary workshop on Web accessibility (W4A) - W4A '08**, p. 45, 2008. Disponível em: <http://portal.acm.org/citation.cfm?doid=1368044.1368054>.

W3C. **Web content accessibility guidelines 1.0. W3C Recommendation**. 1999. [Obtido em 29 de Setembro de 2018]. Disponível em: <http://www.w3.org/TR/WCAG10/>.

W3C. **Techniques for Web Content Accessibility Guidelines 1.0**. 2000. [Obtido em 03 de Outubro de 2018]. Disponível em: <https://www.w3.org/TR/WAI-WEBCONTENT-TECHS/>.

W3C. **Introduction to Web Accessibility**. 2005. [Obtido em 25 de Setembro de 2018]. Disponível em: <https://www.w3.org/WAI/fundamentals/accessibility-intro/>.

W3C. **Web Content Accessibility Guidelines (WCAG) Overview**. 2005. [Obtido em 04 de Outubro de 2018]. Disponível em: <https://www.w3.org/WAI/standards-guidelines/wcag/>.

W3C. **Diretrizes de Acessibilidade para Conteúdo Web (WCAG) 2.0**. 2008. [Obtido em 03 de Outubro de 2018]. Disponível em: <https://www.w3.org/Translations/WCAG20-pt-PT/>.

W3C. **Packaged Web Apps (Widgets) - Packaging and XML Configuration (Second Edition). W3C Recommendation**. 2012. Obtido em 24 de Setembro de 2018. Disponível em: <https://www.w3.org/TR/widgets/>.

W3C. **Accessible Rich Internet Applications (WAI-ARIA) 1.0**. 2013. [Obtido em 20 de Julho de 2017]. Disponível em: <http://www.w3.org/TR/wai-aria/>.

W3C. **A guide to understanding and implementing Web Content Accessibility Guidelines 2.0**. 2016. [Obtido em 04 de Outubro de 2018]. Disponível em: <https://www.w3.org/TR/UNDERSTANDING-WCAG20/>.

W3C. **Web Accessibility Evaluation Tools List**. 2016. [Obtido em 14 de Novembro de 2017]. Disponível em: <http://www.w3.org/WAI/ER/tools/>.

W3C. **Accessible Rich Internet Applications (WAI-ARIA) 1.1.** 2017. [Obtido em 15 de Janeiro de 2018]. Disponível em: <https://www.w3.org/TR/wai-aria-1.1/>.

W3C. **WAI-ARIA Authoring Practices 1.1**. 2017. [Obtido em 15 de Novembro de 2017]. Disponível em: <https://www.w3.org/TR/wai-aria-practices/>.

W3C. **WAI-ARIA Overview**. 2017. [Obtido em 06 de Outubro de 2018]. Disponível em: <https://www.w3.org/WAI/standards-guidelines/aria/>.

W3C. **About W3C WAI**. 2018. [Obtido em 29 de Setembro de 2018]. Disponível em: <https://www.w3.org/WAI/about/>.

W3C. **Web Accessibility Initiative (WAI)**. 2018. [Obtido em 01 de Outubro de 2018]. Disponível em: <https://www.w3.org/WAI/>.

W3C. **Web Content Accessibility Guidelines (WCAG) 2.1**. 2018. [Obtido em 04 de Outubro de 2018]. Disponível em: <https://www.w3.org/TR/WCAG21/>.

W3C. **What's New in WCAG 2.1**. 2018. [Obtido em 05 de Outubro de 2018]. Disponível em: <https://www.w3.org/WAI/standards-guidelines/wcag/new-in-21/>.

W3C Brasil. **Cartilha de Acessibilidade na Web**. 2013. [Obtido em 23 de Setembro de 2018]. Disponível em: <http://www.w3c.br/pub/Materiais/PublicacoesW3C/cartilha-w3cbr-acessibilidade-web-fasciculo-I.html>.

WATANABE, W. M. **Avaliação automática de acessibilidade em RIA**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - USP, 2014.

WATANABE, W. M.; FORTES, R. P. d. M. Automatic identification of drop-down menu widgets using mutation observers and visibility changes. In: **Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC '16**. New York, New York, USA: ACM Press, 2016. p. 766–771. ISBN 9781450337397. Disponível em: <http://dl.acm.org/citation.cfm?doid=2851613.2851860>.

WATANABE, W. M.; FORTES, R. P. M.; DIAS, A. L. Acceptance tests for validating aria requirements in widgets. **Universal Access in the Information Society**, v. 16, n. 1, p. 3–27, Mar 2017. ISSN 1615-5297. Disponível em: <https://doi.org/10.1007/s10209-015-0437-9>.

WATANABE, W. M.; GERALDO, R. J.; de Mattos Fortes, R. P. Keyboard navigation mechanisms in tab widgets. In: **Proceedings of the 29th Annual ACM Symposium on Applied Computing - SAC '14**. New York, New York, USA: ACM Press, 2014. p. 721–726. ISBN 9781450324694. Disponível em: <http://dl.acm.org/citation.cfm?doid=2554850.2554947>.

ZHANG, S. et al. Learning k for knn classification. **ACM Trans. Intell. Syst. Technol.**, ACM, New York, NY, USA, v. 8, n. 3, p. 43:1–43:19, jan. 2017. ISSN 2157-6904. Disponível em: <http://doi.acm.org/10.1145/2990508>.