Detecção Automática de Incompatibilidades Cross-Browser utilizando Redes Neurais Artificiais

Fagner Christian Paes
Universidade Tecnológica Federal do Paraná - UTFPR
Brazil

Willian Massami Watanabe
Universidade Tecnológica Federal do Paraná - UTFPR
Brazil

Resumo—Incompatibilidades *Cross-Browser* (XBIs) representam inconsistências em aplicaç ões Web quando apresentadas em diferentes navegadores. O número crescente de implementaç ões de navegadores *(Internet Explorer, Microsoft Edge, Mozilla Firefox, Opera, Google Chrome)* e as constantes evoluç ões das especificaç ões de tecnologias Web proporcionaram diferenças na forma como os navegadores se comportam e renderizam as páginas Web. As aplicaç ões Web devem se comportar de forma consistente entre os navegadores, portanto, os desenvolvedores web devem superar as diferenças ocorridas durante a renderizaç ão em diferentes ambientes, detectando e evitando XBIs durante o processo de desenvolvimento. Muitos desenvolvedores web dependem de inspeç ão manual das páginas web em vários ambientes de processamento para detectar XBIs, independentemente do custo e a demora que os testes manuais representam ao processo de desenvolvimento. Ferramentas para detecç ão automática de XBIs aceleram o processo de inspeç ão nas páginas web, porém as ferramentas existentes possuem pouca precisão e suas avaliaç ões reportam uma grande porcentagem de falsos positivos. Nesta pesquisa, pretende-se avaliar o uso de redes neurais artificiais para diminuir a quantidade de falsos positivos na detecç ão automática de XBIs através do CSS *(Cascading Style Sheets)* e a comparaç ão relativa do elemento na p ágina web.

Index Terms—Cross-browser, XBI, teste automático.

1 Introdução

Nos últimos anos as aplicações Web se tornaram mais populares, principalmente com o advento da Web 2.0 que proporcionou aos usuários maior interação na organização do conteúdo, pelo fato dos ambientes online estarem mais dinâmicos. Entretanto, os recentes avanços em termos de tecnologias, elevou a complexidade de desenvolvimento de aplicações Web. As aplicações Web seguem um tradicional modelo cliente-servidor, onde parte da aplicação contém componentes sendo executados no lado do servidor e outra parte dos componentes sendo carregados e executados do lado do cliente. Para carregar os componentes do lado do cliente, os usuários podem utilizar uma ampla variedade de navegadores web, como: Internet Explorer, Microsoft Edge, Mozilla Firefox, Opera, Google Chrome, entre outros [1]. A flexibilidade na escolha dos navegadores, por outro lado, aumenta a complexidade, o custo e o esforço durante o ciclo de desenvolvimento do software, pois uma mesma aplicação Web precisa garantir a portabilidade em múltiplos navegadores. Isso significa que cada elemento de uma aplicação web deve ser carregado corretamente e apresentar o mesmo comportamento do conteúdo dinâmico, independente do navegador, versão ou sistema operacional [2]. Neste contexto, as diferenças observadas em aplicações web quando carregadas em múltiplos navegadores são referenciadas como Incompatibilidades Cross-Browser *(XBIs)* em recentes estudos[1], [3].

Uma página Web simples é construída basicamente com três principais tecnologias do lado do cliente: o HTML *(Hypertext Markup Language)*, que contém a estrutura de marcações de um texto, o CSS *(Cascading Style Sheets)* utilizado para definir o estilo da página e o *JavaScript* que lida com o comportamento dinâmico e a interatividade de uma página Web. Diante disso, XBI representa uma característica em cada uma dessas tecnologias (HTML, CSS e JS) que não é suportada consistentemente nos navegadores e se uma aplicação web faz uso desse recurso, então ela apresentará um XBI. A Figura 1 ilustra uma página web que é carregada diferentemente em três navegadores distintos: *Internet Explorer, Google Chrome e Firefox*. Esta incompatibilidade é o resultado de fato de que embora existam várias especificações do navegador que devem orientar o seu desenvolvimento, ainda existem lacunas e necessidades que podem ser implementadas diferentemente pelas equipes de desenvolvimento.

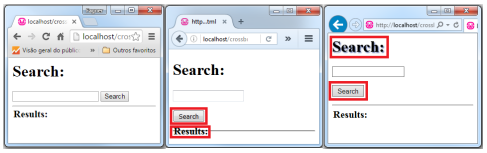

Figura 1. Exemplo XBI entre os Navegadores.

XBIs variam desde problemas simples na interface do usuário até falhas críticas das funcionalidades, portanto, segundo [4], essas incompatibilidades XBIs podem ser classificadas em dois grupos:

- *Incompatibilidades em Layout*: são causadas por diferenças em como os navegadores carregam a página web e são imediatamente visíveis pelo usuário. Essas incompatibilidades podem ser classificadas como diferenças na posição do elemento,

tamanho, visibilidade ou aparência do elemento na página web.

- *Incompatibilidades em Funcionalidade*: estas incompatibilidades estão associadas a funcionalidade ou comportamento de uma aplicação web e a maneira que ela é executada em diferentes navegadores. Essas incompatibilidades podem limitar a maneira que o usuário interage com a aplicação web dependendo do navegador que o usuário está utilizando.

Com base nestes fatores, pretende-se avaliar o uso de redes neurais artificiais para diminuir a quantidade de falsos positivos na detecção automática de XBIs através do CSS e a comparação relativa do elemento na página web.

Esta pesquisa foi organizada da seguinte forma: Seção 2 apresenta as diferentes abordagens para detecção automática de XBIs, a Seção 3 descreve as etapas realizadas na técnica utilizada neste trabalho, na Seção 4 são apresentados os resultados e por fim, são apresentadas as conclusões e trabalhos futuros(Seção 5).

2 Trabalhos Relacionados

A fim de detectar XBIs, os desenvolvedores precisam carregar as aplicações web em diferentes navegadores e inspecionar manualmente se as aplicações web são apresentadas corretamente e se comportam de forma consistente. Atualmente, ferramentas comerciais que visam reduzir o custo desta inspeção manual, tais como, Microsoft Expression Web[1] e Adobe Lab[2], geram automaticamente várias imagens de cada navegador, mas dependem dos desenvolvedores na inspeção manual das imagens para detectar XBIs. Assim, estas abordagens de detecção XBIs ainda são demoradas, exigindo um grande esforço da equipe de desenvolvimento.

O estado da arte nesta área apresenta a implementação de várias ferramentas para detecção automática de XBIs, tais como: WebDiff [5], CrossCheck [3], X-Pert [4], WebMate [6] e Broswerbite [7]. As ferramentas possuem o propósito de utilizar a análise da estrutura DOM, comparação de imagens, isomorfismo de grafos, aprendizado de máquina e comparação da posição relativa para detecção de XBIs. Estas técnicas foram implementadas separadamente nas ferramentas com o objetivo de reduzir o número de falsos positivos reportados por cada ferramenta. Falsos positivos são incompatibilidades identificadas pelas ferramentas mas que não são XBIs e que devem ser avaliadas manualmente pelo desenvolvedor web, o que afeta negativamente o processo de desenvolvimento.

WebDiff, por exemplo, implementou ambas as técnicas, análise de estrutura DOM e comparação de imagens. A avaliação conduzida relatou taxa de 17% de falsos positivos [1]. Em outro estudo, tendo diferentes amostras de grupos de páginas web, Choudhary et al. desenvolveram a ferramenta CrossCheck com as técnicas de análise de modelo de navegação e aprendizado de máquina [3]. Neste estudo, a ferramenta CrossCheck obteve um desempenho melhor que a ferramenta WebDiff, apresentou uma taxa de 64% de falsos positivos, enquanto o WebDiff obteve uma taxa de 79% de falsos positivos.

1. veja: https://www.microsoft.com/expression/eng/
2. veja: http://labs.adobe.com/

Alguns anos mais tarde, Choudhary et al. implementaram a ferramenta X-Pert com um algoritmo de comparação da posição relativa [4]. Neste último estudo, em outra avaliação com uma amostra distinta de páginas web de outros estudos, X-Pert apresentou taxa de 24% de falsos positivos contra uma taxa de 82% de falsos positivos apresentados pelo CrossCheck.

O trabalho de Saar et al., que apresentou a ferramenta Browserbite [7], relata a comparação entre o uso de árvores de decisão e redes neurais para detectar XBIs em aplicações web através de regiões de interesse, diferentemente de outras abordagens que utilizam elementos do DOM da página web. Nos estudos, Browserbite apresentou taxa de apenas 4% de falsos positivos.

Vale destacar que todos os trabalhos relacionados fazem o uso de uma variedade de técnicas, com o objetivo de reduzir o número de falsos positivos. Esta proposta de pesquisa também tem o objetivo de reduzir o número de falsos positivos, mas diferentemente das técnicas existentes, foi proposto o uso de redes neurais artificiais para análise dos atributos CSS e a comparação da posição relativa dos elementos na página web.

3 Proposta

Esta pesquisa tem como objetivo avaliar o uso de redes neurais artificiais para detecção automática de XBIs. A técnica consiste no uso de um conjunto de páginas web com XBIs e sem XBIs para ensinar a rede neural artificial multicamada *(Backpropagation)* para identificar diferenças nos valores dos atributos CSS e na posição relativa do elemento na página web.

A técnica utiliza como conjunto de dados para treinar a rede neural as características extraídas do resultado da classificação dos screenshots de cada elemento HTML da página web, os valores dos atributos CSS relacionados a posição do elemento HTML e a diferença da posição do elemento e suas relações de pai, filho e irmãos na estrutura da página web. A pesquisa pretende contribuir com o estado da arte através da diminuição de falsos positivos, utilizando redes neurais artificiais treinadas com as características extraídas da página web.

As seguintes seções apresentam em detalhes cada etapa realizada para o aprendizado supervisionado da rede neural artificial proposto na pesquisa (Figura 2).

3.1 Etapa de Coleta de Dados

Nesta primeira etapa, são coletados os dados que serão classificados para formar o conjunto de dados para treinamento da rede. A coleta de dados foi realizada através de uma lista de 50 páginas web.

3.1.1 Valores dos atributos CSS e Posição Relativa dos Elementos

Nesta etapa, o crawler lê a lista de 50 páginas web e uma lista de atributos CSS (A lista de CSS foi formada por atributos CSS relacionados a posição do elemento HTML na página web), em seguida o crawler carrega uma página web por vez no navegador referência para iniciar a procura dos atributos CSS em cada elemento HTML e armazena os

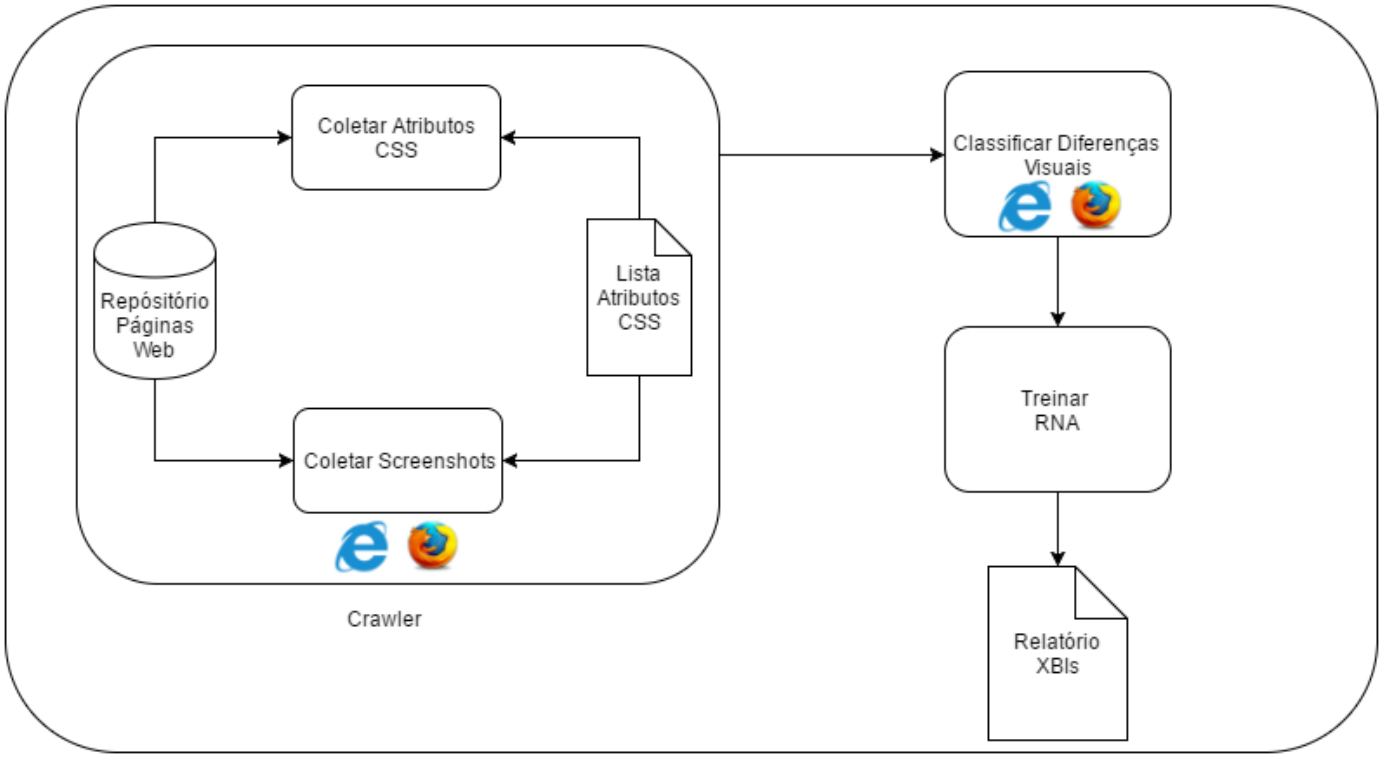

Figura 2. Visão Geral

valores dos atributos CSS e o posicionamento do elemento em um arquivo. Este procedimento é repetido em todas as páginas web existentes na lista. No final dessa etapa, existe um arquivo com todas as páginas web, seus respectivos elementos HTML, os valores de atributos CSS e a posição relativa.

3.1.2 Screenshots

Nesta etapa, o crawler captura o screenshot individual de cada elemento identificado na página web no navegador referência e no navegador em teste apenas para na etapa de extração das características poder realizar a classificação manual dos dados que serão utilizados posteriormente no treinamento. Este procedimento também é realizado em todas as páginas web existentes na lista. No final dessa etapa, existe um repositório com todos os screenshots dos elementos HTML da página web.

3.2 Etapa de Extração das Características

Nesta etapa, são extraídas as características dos dados coletados na etapa anterior.

3.2.1 Classificar Diferenças Visuais

O procedimento de classificar as diferenças visuais ocorreu neste momento da pesquisa de forma manual. O objetivo da classificação foi preparar o conjunto de dados que serviu de entrada para o treinamento da rede neural. Nesta etapa, os elementos capturados pelo crawler foram analisados e os respectivos screenshots de ambos os navegadores comparados. Quando encontrado diferenças entre os screenshots o registro do conjunto de dados era classificado com XBI, conforme exibido na Figura 3 diferenças entre o navegador referência (FF) e o navegador em teste (IE).

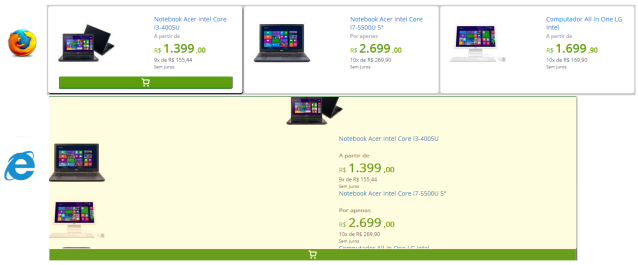

Figura 3. Classificação das Diferenças Visuais

3.3 Etapa de Treinamento

Nesta etapa, exploramos redes neurais artificial *multi-layer perceptron* que é uma técnica de aprendizado de máquina. Foi utilizada para treinar a rede neural o algoritmo *Back-propagation* com três camadas (entrada, oculta, saída) implementado pela ferramenta Weka [3]. As redes neurais suportam um modelo matemático inspirado na estrutura neural que adquirem conhecimento através da experiência. As principais características da redes neurais estão relacionadas com a sua capacidade de aprendizado e a evolução da forma como executam uma tarefa específica através de treinamento contínuo [8]. A regra mais comum para o número de neurônios na camada oculta é o valor médio do número entradas e o número de neurônios na camada de saída [8]. Como o número de entradas são 20 e o número de neurônios na camada de saída são 2, a camada oculta foi treinada com 11 neurônios. A ferramenta Weka disponibiliza essa definição para o número de neurônios na

3. veja: http://www.cs.waikato.ac.nz/ml/weka/

camada oculta como padrão. Os neurônios foram treinados com um conjunto de dados de 2.332 amostras de elementos HTML extraídos das 50 páginas web rastreadas pelo *crawler*. A Tabela 1, apresenta os resultados do treinamento da rede.

Tabela 1
Resultado da etapa de treinamento

Medida	Conjunto Treinamento
Precision(%)	92
Recall(%)	91

3.4 Etapa de Testes

Nesta etapa, foram utilizados dois métodos de particionamento da base de dados disponíveis na ferramenta Weka:

- *Cross-Validation*: O conjunto de dados original foi dividido no primeiro ciclo de teste em 10 subconjuntos do mesmo tamanho, o algoritmo executa o total de subconjunto-1 para treinamento e 1 subconjunto para teste. Também foi realizado um segundo ciclo de teste com 5 subconjuntos para avaliar possíveis diferenças nos resultados.
- Holdout: O conjunto de dados original foi dividido em 2 subconjuntos, 2/3 dos dados para treinamento e 1/3 para teste.

Na próxima seção, serão apresentados os resultados obtidos.

4 Resultados

Os resultados dos testes (Tabela 2), foram avaliados através das medidas de precisão e *recall*. A precisão significa a quantidade de XBIs encontrados em relação a quantidade de XBIs previstos (classe prevista) no conjunto de dados de treinamento. Já a medida *recall* significa a quantidade de XBIs encontrados em relação a quantidade de XBIs real (classe real) existentes no conjunto de dados de treinamento. Os valores percentuais foram extraídos da média ponderada das classes nas matrizes de confusão (Tabelas 3,4 e 5).

Tabela 2
Resultado da etapa de testes

Medida	Cross-Validation 10 subconjunto	Cross-Validation 5 subconjunto	Holdout
Precision (%)	89	88	82
Recall (%)	89	88	82

A Tabela 3 representa a matriz de confusão do método Cross-Validation 10 subconjuntos que apresentou resultado um pouco superior em relação aos outros testes, em seguida as Tabelas 4 e 5 apresentam as matrizes de confusão dos métodos *Cross-Validation* 5 subconjuntos e *Holdout* respectivamente. Em relação ao resultados inferiores na etapa de teste do método *Cross-Validation* e *Holdout* é importante destacar que os métodos durante os testes possuem subconjuntos de dados que não fizeram parte do treinamento, por isso existe a tendência da precisão e recall cair em comparação ao conjunto de treinamento. Importante lembrar que o objetivo das redes neurais são prever o resultado de conjuntos de dados que nunca foram usados no treinamento.

Tabela 3
Matriz de Confusão *Cross-Validation* 10 Subconjuntos

Classe Real	Classe Prevista		
		Não XBI	**XBI**
	Não XBI	421	142
	XBI	121	1648

Tabela 4
Matriz de Confusão *Cross-Validation* 5 Subconjuntos

Classe Real	Classe Prevista		
		Não XBI	**XBI**
	Não XBI	382	181
	XBI	87	1682

Tabela 5
Matriz de Confusão *Holdout*

Classe Real	Classe Prevista		
		Não XBI	**XBI**
	Não XBI	113	74
	XBI	66	524

O resultado da nova técnica foi significante, apesar de não superar o estado da arte. Dado as diferenças na configuração do experimento e avaliação de uma técnica diferente, não são conclusões comparativas, mas o resultado sugere a continuação da pesquisa em redes neurais com conjunto de dados gerado através dos atributos CSS e a posição relativa de elementos para diminuição de falsos positivos.

5 Conclusão

Essa pesquisa avaliou o uso de redes neurais artificiais como uma técnica para detecção automática de XBIs em atributos CSS e na comparação da posição relativa dos elementos HTML. Os resultados obtidos credenciam as redes neurais com alto nível de precisão na detecção de XBIs. Em trabalhos futuros, tendo em vista que esta pesquisa utilizou um conjunto de dados limitado para o treinamento, serão necessários a revisão dos critérios de seleção dos atributos CSS e o aumento da qualidade do conjunto de dados na etapa de coleta de elementos e *screenshots* para aumentar a precisão da técnica.

Referências

[1] S. Choudhary, H. Versee, and A. Orso, "Webdiff: Automated identification of cross-browser issues in web applications," in *Proc. of the 2010 IEEE International Conference on Software Maintenance (ICSM 2010)*, Sept 2010, pp. 1–10.

[2] V. Dallmeier, M. Burger, T. Orth, and A. Zeller, *WebMate: Generating Test Cases for Web 2.0*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 55–69. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35702-2$_5$

[3] S. Choudhary, M. Prasad, and A. Orso, "Crosscheck: Combining crawling and differencing to better detect cross-browser incompatibilities in web applications," in *Proc. of the 5th International Conference on Software Testing, Verification and Validation (ICST 2012)*, April 2012, pp. 171–180.

[4] ——, "X-pert: Accurate identification of cross-browser issues in web applications," in *Proc. of the 35th International Conference on Software Engineering (ICSE 2013)*, May 2013, pp. 702–711.

[5] S. Choudhary, H. Versee, and A. Orso, "A cross-browser web application testing tool," in *Proc. of the 2010 IEEE International Conference on Software Maintenance (ICSM 2010)*, Sept 2010, pp. 1–6.
[6] V. Dallmeier, M. Burger, T. Orth, and A. Zeller, "Webmate: A tool for testing web 2.0 applications," in *Proc. of the Workshop on JavaScript Tools (JSTools 2012)*. New York, NY, USA: ACM, 2012, pp. 11–15. [Online]. Available: http://doi.acm.org/10.1145/2307720.2307722
[7] N. Semenenko, M. Dumas, and T. Saar, "Browserbite: Accurate cross-browser testing via machine learning over image features," in *Proc. of the 29th IEEE International Conference on Software Maintenance (ICSM 2013)*, Sept 2013, pp. 528–531.
[8] J. Heaton, *Introduction to Neural Networks for Java, 2Nd Edition*, 2nd ed. Heaton Research, Inc., 2008.