Browserbite: Accurate Cross-Browser Testing via Machine Learning Over Image Features

Nataliia Semenenko, Marlon Dumas
Institute of Computer Science
University of Tartu, Estonia
{nataliia, marlon.dumas}@ut.ee

Tõnis Saar
Browserbite and Software Technology and Applications Competence Center, Estonia
tonis.saar@stacc.ee

Abstract— **Cross-browser compatibility testing is a time consuming and monotonous task. In its most manual form, Web testers open Web pages one-by-one on multiple browser-platform combinations and visually compare the resulting page renderings. Automated cross-browser testing tools speed up this process by extracting screenshots and applying image processing techniques so as to highlight potential incompatibilities. However, these systems suffer from insufficient accuracy, primarily due to a large percentage of false positives. Improving accuracy in this context is challenging as the criteria for classifying a difference as an incompatibility are to some extent subjective. We present our experience building a cross-browser testing tool (Browserbite) based on image segmentation and differencing in conjunction with machine learning. An experimental evaluation involving a dataset of 140 pages, each rendered in 14 browser-system combinations, shows that the use of machine learning in this context leads to significant accuracy improvement, allowing us to attain an F-score of over 90%.**

Keywords—cross-browser testing; machine learning; image processng

I. Introduction

A long-standing issue in Web application development is that a given Web page may be rendered differently in different browsers or systems depending on the rendering engine, screen resolution, font, etc. Some of these differences are minor or even imperceptible, but others constitute layout, formatting or functional incompatibilities. Manual detection of these incompatibilities by means of visual inspection is labor-intensive and error-prone, as fatigue may cause testers to miss incompatibilities. Depending on a Web site's popularity, the number of browser-system combinations that need to be tested in order to cover 90-95% of users can go up to 20-30 [1]. This is a major hindrance for Web application maintenance.

A number of automated cross-browser testing prototypes, such as CrossCheck [2] and WebDiff [3] as well as commercial tools such as Mogotest[1] and Browsera[2] significantly reduce the required amount of manual effort by automating the screenshot capture and comparison steps. However, these systems suffer from over-sensitivity and produce an excessive amount of false positives. For instance, an evaluation of CrossCheck showed 64% of false positives, while for WebDiff this number reached 79% [2]. Fundamentally, this is due to the fact that what constitutes an incompatibility, as opposed to a simple difference, is to some extent subjective [2]. Thus, setting specific thresholds to classify a difference as an incompatibility is far from trivial.

This paper reports our experience building an industrial-strength cross-browser compatibility testing tool, namely Browserbite[3], by combining image processing techniques with machine learning. One of the novelties of Browserbite is that rather than attempting to set thresholds for classifying image discrepancies as incompatibilities during image comparison, the task of classifying differences as incompatibilities is almost entirely pushed to the machine learning phase. We evaluate the accuracy of neural networks and classification trees for this task, based on a dataset of 140 popular Web pages rendered across 14 configurations.

The rest of the paper is structured as follows. Section 2 gives an overview of the Browserbite system. Section 3 introduces the machine learning approach to post-process the output of Browserbite's image comparison module. Next, Section 4 discusses the evaluation results, while Section 5 concludes and discusses directions for future work.

II. Browserbite

The aim of Browserbite is to detect potentially incompatible renderings of a given Web document (identified by its URL) across different browsers and operating systems (OS). At present, Browserbite supports 14 browser-OS combinations (called *configurations*), covering major versions of popular browsers (Chrome, Firefox, IE, Safari) running on Windows XP, Windows 7 and Mac OS.

Browserbite's architecture comprises three main modules: screenshot capture, screenshot comparison and classification. The screenshot capture module consists of a scheduler that controls a number of workers. The workers are instances of different types of Virtual Machines (VMs). Each worker is capable of taking full-page screenshots of Web pages in a given configuration.

[1] http://mogotest.com
[2] http://browsera.com
[3] http://www.browserbite.com

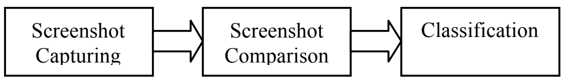

Fig. 1 Browserbite main modules

Web pages are opened and rendered using Selenium.[4] Full-page capture is achieved via image stitching or window resizing depending on the configuration. In the first case, the Web page is automatically scrolled and a screenshot is taken after each scrolling operation. The resulting partial-page screenshots are stitched together into a full Web page image. In the second case the browser window is resized to match the size of the web page document, and a single full Web page screenshot is taken. The latter method has been found to work with IE, while the former method is used for other browsers.

The screenshot comparison module relies on image segmentation and comparison techniques. Its input is a collection of screenshots (images). One of these images is designated by the user as the *baseline image*, while other images are called *Images Under Test (IUTs)*. The baseline image is meant to correspond to a correct rendering of the Web page. IUTs are compared against the baseline image.

Each image is first segmented into smaller rectangular regions called Regions of Interest (ROI), based on borders and color changes in the image. The set of ROIs extracted from the baseline image is matched with the set of ROIs extracted from each of the IUTs. An ROI in the baseline image (called an ROIB) is mapped to at most one ROI in the IUT (an ROIT). The matching of an ROIB to an ROIT is based on a correlation-based image comparison technique. The ROIB and ROIT are correlated and a *correlation index* is extracted. This index captures the similarity between the pair of ROIs. An ROIB and ROIT are declared compatible if their similarity is above a certain threshold. Otherwise, Browserbite reports the pair (ROIB, ROIT) as a *potential incompatibility*. Every ROIB not matched to an ROIT (*missing ROI*) is reported as an incompatibility of the form (ROIB, ⊥), where ⊥ is the null value. Conversely, an ROIT not matched to any ROIB (*additional ROI*) is reported as a pair (⊥, ROIT). When reporting a potential incompatibility, Browserbite superposes the ROIB to the ROIT using 50% transparency, enabling users to check the extent of the incompatibility. For instance, Fig. 2 shows a true positive, while Fig. 3 shows a false positive.

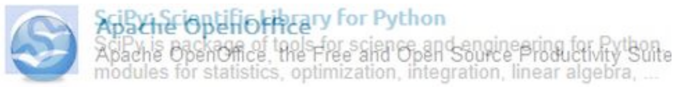
SciPy Scientific Library for Python
SciPy is package of tools for science and engineering for Python modules for statistics, optimization, integration, linear algebra, ...
Apache OpenOffice
Apache OpenOffice, the Free and Open Source Productivity Suite

Fig. 2. Difference corresponding to an actual incompatibility (www.sourceforge.net). The text element presented in the baseline is absent in the IUT; instead IUT contains another text.

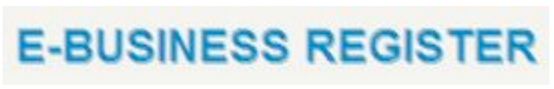

Fig. 3. Minor difference that users did not classify as an incompatibility (www.rik.ee)

An empirical evaluation of an early pre-commercial version of Browserbite by Tamm [4] showed a rate of false positives of around 10% and a rate of false negatives of 44%. Attempts to manually fine-tune the correlation index threshold and other parameters used during image comparison turned out to be unproductive, as decreases in false negatives led to considerable increases in false positives. Given the strong impact that false negatives can have in commercial usage, we decided to increase the sensitivity of Browserbite in the first commercial release of the tool, in such a way that false negatives rate is close to zero, at the expense a high rate of false positives.

Feedback collected during commercial usage confirmed that Bowserbite is oversensitive and practically does not miss any incompatibility. To validate this observation, we conducted another experiment in which the first author of the paper manually compared pairs of Web pages rendered on different configurations. A total of 140 Web pages (dataset described below) were analyzed and each was rendered in four browsers (Chrome 22.0, IE8 and Firefox 3.6 and 16.0.1 on Windows 7). The resulting screenshot pairs were manually compared by the first author. We found that 98% of incompatibilities detected manually were reported by Browserbite in the form of ROI pairs (i.e. 98% recall), but the precision on the other hand was in the order of 66%.

Accordingly, we decided to supplement the screenshot comparison module with a classification module in which machine learning is used to reduce the false positive rate. The next section discusses the classification module.

III. CLASSFICATION MODULE

The aim of the classification module is to classify potential incompatibilities reported by Browserbite's screenshot comparison module into two categories: true positives (the potential incompatibility is perceived as such by a user) and false positives (the potential incompatibility is not perceived as such by a user). Below we present the datasets used for training/testing classification models, the employed features and machine learning techniques.

A. Dataset and Golden Standard

We collected a dataset consisting of home pages of the top 140 Websites of Estonia according to Alexa[5]. Each Web page was given as input to Browserbite, which generated around 20000 potential incompatibilities (ROI pairs). The first author trimmed down this set to 1200 potential incompatibilities by manually identifying 600 pairs that were likely to be true incompatibilities and 600 pairs that were likely to be false positives. This classification by the first author was only used to extract a balanced subset of samples. The judgments made by the first author were discarded in the subsequent evaluation – only the set of 1200 ROI pairs was kept.

We recruited 40 subjects through social media and asked them to classify pairs of ROIs into the two classes: "no difference or insignificant difference" and "major difference". Subjects were asked to put a pair in the first class if either they

[4] http://seleniumhq.org

[5] http://alexa.com

noticed no difference at all, or they noticed a minor layout difference, which in their opinion would not affect their perception of a Web page containing that segment. Otherwise they were instructed to classify the pair in the second category.

Respondents were University students in the range of 20-25 years from 6 countries (Estonia, Russia, Ukraine, Germany, Italy and Hungary). The subjects came from different specialties: 60% with the IT background, 20% with economics and business background, 10% with philological background, and 10% others. Each respondent classified between 200 and 400 (ROIB, ROIT) pairs randomly sampled from the dataset with replacement. On the end, we obtained at least 8 classifications for each pair (up to 15 in some cases). For uniformity, we randomly trimmed the dataset so that each (ROIB, ROIT) had exactly 8 judgments (i.e. 8 subjects per pair). The final dataset contained 50.4% of true positives and 40.6% of false positives.[6] The inter-rater reliability of the resulting dataset is 0.94[7], indicating little disagreement between judges. We aggregated the judgments by marking a potential incompatibility as a true incompatibility if at least 5 subjects rated it as a "major difference".

B. Feature Set

We recall that an incompatibility reported by Browserbite consists of a pair (ROIB, ROIT) where ROIB is an ROI in the baseline image and ROIT is a corresponding ROI in the IUT. In case of a missing or additional ROI, ROIT and ROIB can take null values. Given a pair (ROIB, ROIT), we extract, the following 17 features to build a sample for constructing classification models:

- 10 *histogram bins* (h0, h1, … h9). These 10 integers encode the image histogram of the ROIB. 10 discrete bins represent pixel intensity distribution across the entire ROI image;
- Correlation between the ROI in the baseline image and ROI in the IUT. This is a number between zero and one. It is close to zero in case of very low correlation between ROIB and ROIT. It is zero in case of a missing or additional image.
- Horizontal and vertical position of the ROIB (X and Y coordinates) on the baseline image;
- Horizontal and vertical size of ROIB (width and height) of the baseline image;
- Configuration index – a numerical identifier of the browser-platform combination of the IUT. Browserbite supports 14 browser-platforms combinations, thus this is an integer between 1 and 14;
- Mismatch Density MD = E / T, where E is the number of ROIs in the IUT that are not matched 100% to an ROI in the baseline image, and T is the total number of ROIs in the IUT. This is a feature of the IUT itself rather than of an ROI inside the IUT. However, for the sake of convenience when constructing the machine learning models, we make the MD a feature of each ROI. All ROIs extracted from the same IUT will have the same MD (the MD of their enclosing IUT).

C. Machine Learning Techniques

We explored two popular machine learning techniques for classification: classification trees (i.e. decision tree) [6] and artificial neural networks [7]. Specifically, we used the implementations of these techniques provided in Matlab.

The use of classification trees is motivated by the fact that they provide a convenient way to interpret the model. By analyzing the classification tree, we can obtain insights into the thresholds that determine whether a potential incompatibility is an actual incompatibility or not.

Neural networks imitate the brain's ability to sort out patterns and learn from trials and errors, discerning and extracting the relationships that underlie the data with which it is presented. Studies have shown that neural networks are a promising alternative to standard classification methods [7]. In this respect, a key advantage of neural networks is their ability to adjust themselves to the data without any explicit specification of functional or distributional form.

We selected the 3-layered feed-forward neural network. The first layer (input layer) consists of 17 neurons corresponding to the number of features. The output layer has 2 neurons (binary classification). As the dataset is not linearly separable one or more additional "hidden" layers are needed. In practice, very few problems that cannot be solved with a single hidden layer can be solved by adding another hidden layer [8]. Accordingly, we chose one hidden layer.[8]

The number of neurons in the neural network is another important parameter, as too few hidden neurons can cause underfitting so that the neural network cannot learn the details. Conversely, a too large number of hidden neurons can cause overfitting, as the neural network starts to learn insignificant details. Accordingly, the number of hidden neurons was determined experimentally. In order to determine the appropriate number of hidden neurons we applied empirically derived rules-of-thumb. One of the most common is that the number of hidden neurons should be around the mid-point between the size of the input and size of the output layers [9]. As the number of input neurons equals 17 (the number of features) and the number of output neurons equals 2 and 4 for binary and quaternary classification respectively, we tried to find an optimal number of hidden neurons between 8 and 13. To this end, we trained the neural network with different number of neurons and calculated the F-score for each trained model, using a set of 200 (ROIB, ROIT) samples not used in the subsequent evaluation. We experimentally found that the peak in F-score is reached for a number of hidden neurons of 11. This number was used in the evaluation reported below.

[6] False negatives were identified by the first author separately. External subjects were used to classify true and false positives.

[7] Calculated using the Inter-Rater Reliability Calculator at http://www.med-ed-online.org/rating/reliability.html which implements the measure in [5]

[8] Additional experiments conducted after the evaluation reported here confirmed that adding a hidden layer does not improve F-score.

Using the golden standard described above, we compared the classification accuracy of Browserbite without machine learning post-processing, with that of Browserbite post-processed with a classification tree and Browserbite post-processed with a neural network. Classification accuracy is measured in terms of precision, recall and F-score with their standard definitions [10]. The machine learning models were trained and evaluated using a five-fold cross-validation method. In other words, the dataset was partitioned into five equal parts, four parts were used to train a model and the remaining one was used to test the model. This process was repeated 5 times with each part playing the testing role once. The results from each fold were averaged to produce a single measurement of precision, recall and F-score for each method (classification tree and neural network).

Additionally using the same dataset, we also evaluated Mogotest – a commercial tool for cross-browser compatibility testing based on analysis of Document Object Models (DOM).

IV. Evaluation Results

The evaluation results are summarized in Table 1. It can be seen from the tables that neural networks outperform by far classification trees. The neural network achieves a very high precision at the expense of some degradation in recall. The improvement in precision provided by classification trees is less significant, and comes at the expense of a drop in recall.

Table 1. Accuracy for Browserbite w/out classification, Mogotest, Browsebite + classificaton tree and Browserbite+ neural network

Measure	***Plain Browserbite***	***Mogotest***	***Classification tree***	***Neural network***
Precision	0.66	0.75	0.844	0.964
Recall	0.98	0.82	0.792	0.886
F-score	0.79	0.78	0.81	0.923

These results are a significant improvement with respect to state of the art techniques such as CrossCheck [2], which achieves a precision of 36% (64% of false positives) and its predecessor WebDiff, with a precision of 21% according to an evaluation reported in [2]. This latter evaluation of CrossCheck and WebDiff is focused on false positives (false negatives are not reported). Assuming 100% recall, these results imply an F-score of 52% and 35% respectively. Given the differences in experimental setups and evaluation goal, no conclusive comparative statements can be drawn, but the results suggest that Browserbite enhanced with a neural network-based classification module achieves high accuracy relative to state-of-the-art techniques.

V. Conclusion

This paper presented and evaluated the Browserbite cross-browser testing tool with an emphasis on its classification module. The results show that neural networks for incompatibility classification provide a high level of accuracy in this context (precision of 96% with recall of 89%) outperforming classification trees. Given the improvements achieved, the neural network technique has been productized and included in Browserbite's private beta version (to be released in the public version later in 2013).

While the approach has been framed in the context of Browserbite, the underlying principles may be applied to enhance other image-based cross-browser testing techniques such as CrossCheck. Validating the technique in other settings is a direction for future work. A related direction is to evaluate the proposed technique with different types of stakeholders involved in Web application development (e.g. Web designers versus testers versus developers). In this respect, one can hypothesize that classification models for designers would be different than those for developers, for example.

Browserbite is representative of tools for single-page cross-browser compatibility testing. Prototypes and techniques such as Crosscheck [2], Webmate [11], [12] or the technique reported in [13], address the complementary problem of behavioral testing, meaning that they detect incompatibilities that arise when navigating from a given page. The integration of the techniques explored in this paper with behavioral testing techniques is another avenue for future work.

References

[1] StatCounter Gglobal Stats.. [Online]. http://gs.statcounter.com

[2] S. R. Choudhary, M. R. Prasad, A. Orso, "CROSSCHECK: Combining Crawling and Differencing To Better Detect Cross-browser Incompatibilities in Web Applications," in *Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation* (ICST), Montreal, Canada, 2012, pp. 171–180.

[3] S. R. Choudhary, H. Versee, A. Orso, "WEBDIFF: Automated Identification of Cross-browser Issues in Web Applications," in *Proceedings of the 2010 IEEE International Conference on Software Maintenance (ICSM)*, Timisoara, Romania, 2010, pp. 1–10.

[4] A.-L. Tamm, "Visual testing in different testing approaches", University of Tartu, Bachelor thesis 2012.

[5] R. L. Ebel, "Estimation of the reliability of ratings," *Psychometrica*, vol. 16, no. 4, pp. 407-424, 1951.

[6] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, *Classification and Regression Trees*, 1st ed.: Chapman and Hall/CRC, 1984.

[7] G. P. Zhang, "Neural Networks for Classification: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 30, no. 4, pp. 451-462, 2000.

[8] J. Heaton, *Introduction to Neural Networks for Java*, 2nd ed.: Heaton Research, 2005.

[9] A. Blum, *Neural Networks in C++: An Object Oriented Framework for Building Connections,* NY: John Wiley & Sons, 1992.

[10] D. M. W. Powers, "Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness, and Correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37-63, 2011.

[11] V. Dallmeier, M. Burger, T. Orth, A. Zeller, "WebMate: A Tool for Testing Web 2.0 Applications," in *Proceedings of the Workshop on JavaScript Tools*, Beijing, China, 2012, pp. 11-15.

[12] V. Dallmeier, M. Burger, T. Orth, A. Zeller, "WebMate: Generating Test Cases for Web 2.0," in *Software Quality. Increasing Value in Software and Systems Development*, S. Biffl D. Winkler, Ed.: Springer, 2013, pp. 55-69.

[13] A. Mesbah, M. R. Prasad, "Automated Cross-Browser Compatibility Testing," in *Proceedings of the 33rd International Conference on Software Engineering, ICSE'11*, Honolulu, HI, USA, 2011, pp. 561–570.