# FaceOff: Assisting the Manifestation Design of Web Graphical User Interface

Shuyu Zheng
Peking University
zhengshuyu@pku.edu.cn

Ziniu Hu
Peking University
bull@pku.edu.cn

Yun Ma*
Tsinghua University
yunma@tsinghua.edu.cn

## ABSTRACT

Designing desirable and aesthetical manifestation of web graphic user interfaces (GUI) is a challenging task for web developers. After determining a web page's content, developers usually refer to existing pages, and adapt the styles from desired pages into the target one. However, it is not only difficult to find appropriate pages to exhibit the target page's content, but also tedious to incorporate styles from different pages harmoniously in the target page. To tackle these two issues, we propose FaceOff, a data-driven automation system that assists the manifestation design of web GUI. FaceOff constructs a repository of web GUI templates based on 15,491 web pages from popular websites and professional design examples. Given a web page for designing manifestation, FaceOff first segments it into multiple blocks, and retrieves GUI templates in the repository for each block. Subsequently, FaceOff recommends multiple combinations of templates according to a Convolutional Neural Network (CNN) based style-embedding model, which makes the recommended style combinations diverse and accordant. We demonstrate that FaceOff can retrieve suitable GUI templates with well-designed and harmonious style, and thus alleviate the developer efforts.

## KEYWORDS

Web design mining, Web design assistance, Template retrieval

**ACM Reference Format:**
Shuyu Zheng, Ziniu Hu, and Yun Ma. 2019. FaceOff: Assisting the Manifestation Design of Web Graphical User Interface. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3289600.3290610

## 1 INTRODUCTION

For web developers, developing a web page calls for designing not only the page's content and functionality but also the manifestation of graphical user interfaces (GUI), which determines the appearance of the page. Well-designed manifestation can result in an eye candy touch to attract users' attention. Basically, GUI's manifestation considers the neatness and usability of the layout, harmony in color combination, overall design style and so on. Since these considerations require a good taste of art and beauty, it is arduous for developers to efficiently design decent manifestation.

Given a page under design (PUD), a common practice of designing manifestation is to browse through well-designed pages, pick desired styles that satisfy the content, and integrate these styles into the page. However, it is not only difficult to find appropriate pages for exhibiting the PUD's content, but also tedious to incorporate styles from different pages harmoniously in the PUD.

To tackle these two issues, in this paper, we propose FaceOff, a data-driven automation system that assists the manifestation design of web GUI. Given a PUD's source files (including HTML, JavaScript, images, etc.) as input, the goal of FaceOff is to recommend CSS style rules according to the structure (DOM tree after rendering the HTML) and content (text, images or videos) of the input web page.

The design of FaceOff is driven by two ideas. First, according to a previous study [7], a web page can be divided into several templates with different functionalities, such as information cards with pictures, footers with link and copyright information, headers with drop-down menus, etc. Although different web pages may have different contents, they usually share templates that have the same functionality, so that the styles of these templates could be utilized to design new web pages. Second, styles from the same web page should be well accordant with each other. If we could learn how the styles of different templates are combined harmoniously in one web page, then it is possible to select proper combinations of styles from different web pages to be integrated into a new web page.

To this end, we first construct a repository of web GUI based on 15,491 web pages from popular websites and professional design examples. Next, we extract the common templates among different pages in this repository. Then we build a style-embedding model, which uses convolution neural network (CNN) to encode the compiled image of each GUI template. The image of templates with harmonious style will be mapped adjacently in the embedding space. We regard the templates from the same original web page are harmonious with each other, so as to learn the embedding model. Given a PUD, FaceOff segments it into multiple blocks, and retrieves the matched GUI templates for each block. Then FaceOff recommends multiple and diverse style combinations for each retrieved template based on the style-embedding model. Finally, the developer can choose one desired style combination and get a well-designed web page. Note that the output well-designed web page may still need minor changes made manually by developers in order to fit the content more properly.

## 2 RELATED WORK

In this section, we first introduce the related work on design mining and design assisting tools. Then we highlight the differences of our system from the related work.

• **Design Mining.** The data-driven approaches have been applied to design mining for revealing visual evolution and design patterns. Chen et al. [5] revealed the important landmarks in the aesthetic evolution of web pages from three aspects: information architecture

*Corresponding author.


*WSDM '19, February 11–15, 2019, Melbourne, VIC, Australia*

ACM ISBN 978-1-4503-5940-5/19/02…$15.00
https://doi.org/10.1145/3289600.3290610

models, visual flavor and the media composition. Doosti et al. [6] adopted CNN to characterize the web design style, and further analyzed the style shift through the long history. In addition, previous work has made efforts to quantify visual appeal. Lindgaard et al. [10] revealed that people judge visual appeal as well as trust and usability of homepages consistently. Reinecke et al. [13] implemented image metrics to quantify colorfulness and visual complexity of web pages, and developed a model to predict perceived visual appeal.

• **Design Assisting Tools.** The idea of assisting developers to conduct design has attracted research attentions in recent years. One line of work assists developers to write GUI code more efficiently. Kumar et al. [9] focused on the design mapping where the content of a web page can be transferred into a given template. Nguyen et al. [12] used Optical Character Recognition (OCR) and Computer Vision (CV) techniques to analyze the design sketch, in order to help generate GUI code automatically. Beltramelli et al. [4] further advanced this idea of generating GUI code from images by adopting deep learning models. Another line of work explores the design style itself, and can assist generating better design. Kumar et al. [8] implemented Webzeitgeist, which supports to access the page elements and their properties with the goal of conducting large-scale machine learning and statistical analysis on web design. Yang et al. [14] presented automatic generation in the field of visual-textual presentation layout. They train the model by explicitly learning some aesthetic principles with domain knowledge, including topic-dependent emotion, typography and color harmonization.

Our system, FaceOff, differs from the previous assisting tools mainly in the following two points. First, FaceOff transforms the task of design into a template retrieval problem, which takes both the structure and design style into consideration. Second, FaceOff learns the design style basically in an unsupervised manner, rather than the human defined rules, so that the learned knowledge can be more diverse.

## 3 SYSTEM

In this section, we first discuss the problem of manifestation design. Next, we present the overview of our proposed system. Then, we show how to collect and construct the GUI repository. Finally, we describe the two main components of FaceOff in detail, which are template retriever and style recommender.

### 3.1 Problem Statement

In this paper, we denote the manifestation of a web GUI as the combination of *Structure* (T), *Content* (C) and *Style* (S). *Structure* (T) is the DOM tree obtained after rendering an HTML, which records how different nodes are organized in a tree structure. *Content* (C) represents the text, image or video of each node in the tree. *Style* (S) can be inferred from the CSS rules of each node in the DOM tree. It determines the visual representation of the content, such as width, height, color, and border type.

The ultimate goal of our system is to assist the manifestation design. More specifically, given the input of $\{T, C\}$, which indicates how a web page is organized and its content information, our system should provide a referenced style $S$ to optimize the overall manifestation.

However, such a problem cannot be solved directly. On one hand, there is no quantitative definition of how fitness a manifestation is since the judgment of manifestation is more or less subjective. On the other hand, the searching space of style $S$ is the whole space of CSS rules, making it impossible to enumerate the styles.

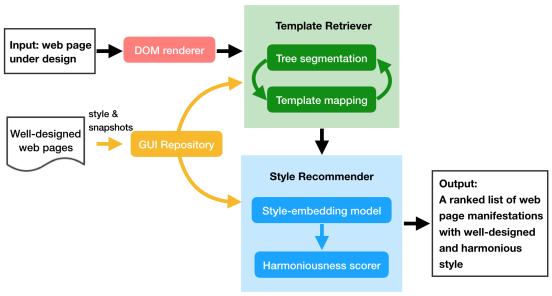


**Figure 1: The system architecture overview of FaceOff.**

Therefore, we resort to current well-designed web pages to narrow our searching space, which leads to our approach.

Our basic idea is to find suitable and harmonious GUI templates from a large-scale web page repository. A template is a common substructure that appears on multiple web pages, such as information cards, headers and footers. For a new page to be designed, we aim to retrieve templates that fit the structure of the page, and recommend combinations of these templates which are harmonious in style. In this way, we can come up with a well-designed web page.

### 3.2 System Overview

Figure 1 shows an overview of FaceOff. We capture the styles and snapshots of web pages from popular websites and professional design examples, and then construct a repository of web GUI. Given a PUD as input, FaceOff first renders the web page in a headless browser to acquire the DOM tree of the web page. The template retriever segments the DOM tree into multiple subtrees and retrieves the matched GUI templates for each subtree. FaceOff recursively finds out the optimized segmentation as well as several top matched templates. The next step is to recommend template combinations. The style recommender consists of a style-embedding model to encode the style of each template, and a pairwise harmoniousness scorer to calculate whether two templates are accordant with each other in style. With these two parts, the style recommender can find out a set of combinations sorted by their total harmoniousness score. The output of the system is a series of ranked well-designed pages for the developer to choose from.

### 3.3 GUI Repository

We collect a large number of well-designed web pages with their DOM tree, style information (279 CSS rules of each node through JavaScript `getComputedStyle` method), and manifestation (screenshot of the web page). Totally, we collect 15,491 different web pages from professional design examples including Bootstrap examples [2] and Webby Awards gallery [3], and popular websites from Alexa top sites [1]. Then, we extract templates from these web pages. We assume that the structure can reflect the functionality of the GUI. Therefore, for each page, we cut out all subtrees from the original DOM tree, and use the subtree as index to identify a template. We group all the subtrees with the same structure together, forming a GUI template. Therefore, subtrees in a GUI template have the same structure but different styles.

### 3.4 Template Retriever

After getting the DOM tree by rendering the PUD, a template retriever will retrieve templates in the GUI repository that match

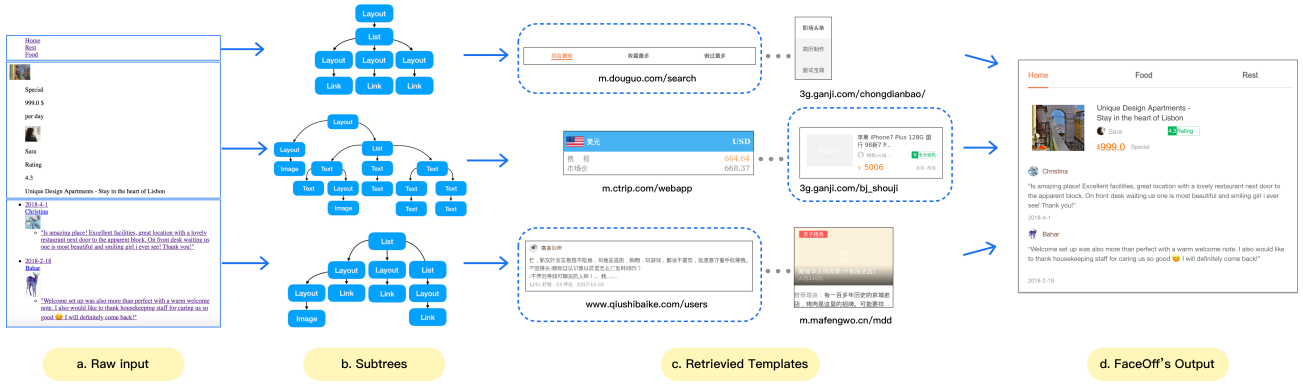


**Figure 2: The overall workflow of FaceOff, to transform a poor-designed hotel-booking page into a well-designed one.**

the given structure. It first segments the PUD into several blocks by dividing the DOM tree into multiple subtrees. The segmentation is based on the structural similarity of subtrees between blocks and GUI templates in the GUI repository. We use a variant of tree edit distance algorithm proposed by Zhang et al. [15] to define the structural similarity of subtrees.

Furthermore, in order to find out the segmentation faster, we adopt a top-down searching algorithm. First, we can find the template in our repository with the least matching gap with the given DOM tree. Then, we divide all the children into subtrees, and apply this algorithm recursively to these subtrees. If the sum of matching gap of all these subtrees is less than that of the complete tree, we should segment it. In this way, we can recursively find out the optimized segmentation to minimize the overall matching gap, and meanwhile retrieve all the templates.

## 3.5 Style Recommender

After the above operations, FaceOff can retrieve a GUI template for each segmented block. As mentioned before, a GUI template contains several subtrees with different styles. However, not all style combinations of GUI templates are in accordance with the whole manifestation. Therefore, the style recommder is to recommend subtree combinations whose styles are harmonious with each other.

To this end, we propose a style-embedding model, which uses a CNN to encode the compiled image of a subtree, and uses a pairwise cosine similarity to model the score of style harmoniousness. To train such an embedding model, we assume that subtrees from the same original DOM tree should be accordant in style with each other since web pages in our repository are well designed. Specifically, we extract the subtrees from the same web page as positive data, and randomly sample some subtrees from different web pages as negative data. To represent the style of the subtree, we use the snapshot of the subtree as input, which is rendered based on the CSS rules of each node of the subtree. In this way, the learned style-embedding model will map the subtrees from the same web page closely, and thus can learn to discriminate style harmoniousness.

Using this embedding model, we can calculate the style harmoniousness score of two subtrees, and can thus recommend a set of combinations sorted by their total harmoniousness score. Afterwards, the developer can choose the desired combination, and FaceOff will use the content of the source HTML file along with the

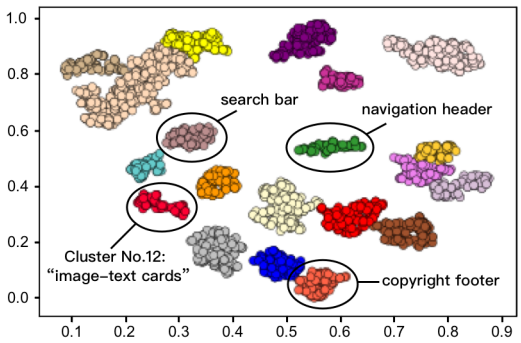


**Figure 3: 20 clusters of the sampled templates, each representing a functionality group.**

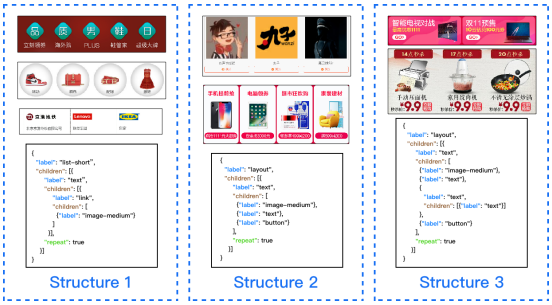


**Figure 4: Three template examples of cluster No.12 (Image-Text Cards), with their subtree and several styles.**

selected combination of template styles to compile and generate a well-designed web page with harmonious style.

# 4 DEMONSTRATION

In this section, we first demonstrate how our system can assist manifestation design with a concrete example, and then illustrate the effectiveness of the template retriever and style recommender.

## 4.1 Overall Workflow Demonstration

First, we use an illustrative example to show the overall workflow of our system[1]. As shown in Figure 2, suppose a developer is

[1]A video of this workflow demonstration is provided at https://youtu.be/lvGaiSSVcyM.

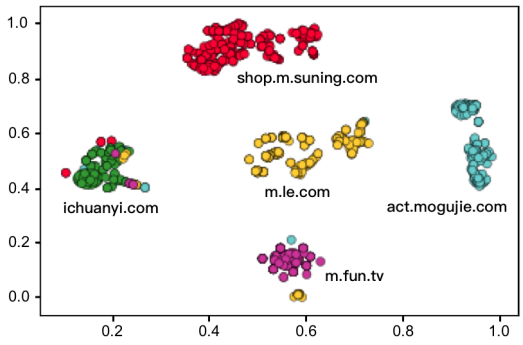


**Figure 5: Image embedding results of 5 sampled web pages. The color indicates the original web page.**

going to design a web page for hotel booking. She has already prepared the content, including some links, a picture of the hotel, brief introduction, room prices, and several comments from customers. The HTML file without any style information is visualized in part *a* of Figure 2.

Part *b* and *c* in Figure 2 show the intermediate results of FaceOff given the raw input in *a*. FaceOff divides the DOM tree into subtrees, and retrieves templates with the structure of these subtrees, which are most similar to the input one in functionality. Part *c* shows two template examples of each segmented subtree, which share the same structure, but exhibit different designs. From these template examples, we can figure out that the input web page exhibits the functionality of a header, an image-text card containing two images, and a list of image-paragraph cards. The optimal style combination comprises the three template examples with dotted frames, which is selected by style recommender from the repository.

Part *d* in Figure 2 presents the final output of FaceOff after compiling and rendering in the browser. The visualization of the output web page is more attractive and harmonious in style compared with the raw input. The result demonstrates that FaceOff can recommend well-designed templates and greatly alleviate the developer efforts.

### 4.2 Effectiveness of Template Retriever

We assume that each template may reflect a certain kind of functionality of web pages. Thus we sample 2000 subtrees from the repository, calculate the distance matrix and apply T-SNE [11] to visualize them. We next use K-means to find 20 clusters among these subtrees. As shown in Figure 3, the subtrees are clustered into separated groups, each of which represents a particular functionality. For example, the groups highlighted by circles are image-text cards, navigation header, search bar and copyright footer. This result demonstrates our assumption on the correspondence between structure and functionality.

To further illustrate the functionality, we explicitly show three subtrees of cluster No.12, which represents image-text cards in Figure 4. We can see that all the subtrees contain image and text in an organized manner. The image of template examples can further illustrate that these templates are indeed of image-text card functionality. In addition, it shows that there are diverse design styles for the same subtree, laying the foundation of the style recommender.

### 4.3 Effectiveness of Style Recommender

Based on our learning objective, the learned style-embedding model should map the templates in the same web page closer. To evaluate this objective, we randomly choose five web pages in our repository, calculate the embedded vectors of all the templates in these pages, and visualize them using T-SNE. As indicated from Figure 5, the templates from the same web page are grouped together, except for a small quantity of special cases. This result indicates that our style-embedding model can capture the diverse design style among different web pages.

## 5 CONCLUSION AND FUTURE WORK

In this paper, we propose to assist manifestation design by retrieving suitable and harmonious GUI templates. To do so, we construct a large-scale web design template repository, and use the structure matching algorithm to implement a template retriever. We further design a matching task to train a CNN-based style embedding model to find the optimal combination of templates with harmonious style. We finally demonstrate how our system can effectively assist manifestation design of web GUI.

As for the future work, we plan to conduct a user study to evaluate the effectiveness of FaceOff, by applying it to existing web pages, and inviting real developers to compare the manifestation recommended by FaceOff and the original manifestation of the page. In addition, we plan to estimate time savings resulting from automatically generating CSS rules against relying on human designers.

## ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under grant number 61725201, China Postdoctoral Science Foundation, and an Open Project Fund of National Engineering Lab of Big Data System Software of China.

## REFERENCES

[1] Alexa top sites. https://www.alexa.com/topsites/, 2018.
[2] Bootstrap featured examples. http://www.youzhan.org/, 2018.
[3] The webby awards. https://www.webbyawards.com/, 2018.
[4] T. Beltramelli. pix2code: Generating code from a graphical user interface screenshot. In *Proceedings of EICS 2018*, pages 3:1–3:6, 2018.
[5] W. Chen, D. J. Crandall, and N. M. Su. Understanding the aesthetic evolution of websites: Towards a notion of design periods. In *Proceedings of CHI 2017*, pages 5976–5987, 2017.
[6] B. Doosti, D. J. Crandall, and N. M. Su. A deep study into the history of web design. In *Proceedings of WebSci 2017*, pages 329–338, 2017.
[7] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *Proceedings of WWW 2013*, pages 830–839, 2005.
[8] R. Kumar, A. Satyanarayan, C. Torres, M. Lim, S. Ahmad, S. R. Klemmer, and J. O. Talton. Webzeitgeist:design mining the web. In *Proceedings of CHI 2013*, pages 3083–3092, 2013.
[9] R. Kumar, J. O. Talton, S. Ahmad, and S. R. Klemmer. Bricolage: example-based retargeting for web design. In *Proceedings of CHI 2011*, pages 2197–2206, 2011.
[10] G. Lindgaard, C. Dudek, D. Sen, L. Sumegi, and P. Noonan. An exploration of relations between visual appeal, trustworthiness and perceived usability of homepages. *ACM Transactions on Computer-Human Interaction*, 18(1):1–30, 2011.
[11] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
[12] T. A. Nguyen and C. Csallner. Reverse engineering mobile application user interfaces with remaui. In *Proceedings of ASE 2015*, pages 248–259, 2015.
[13] K. Reinecke, T. Yeh, L. Miratrix, R. Mardiko, Y. Zhao, J. Liu, and K. Z. Gajos. Predicting users' first impressions of website aesthetics with a quantification of perceived visual complexity and colorfulness. In *Proceedings of CHI 2013*, pages 2049–2058, 2013.
[14] X. Yang, T. Mei, Y. Q. Xu, Y. Rui, and S. Li. Automatic generation of visual-textual presentation layout. *ACM Transactions on Multimedia Computing Communications & Applications*, 12(2):1–22, 2016.
[15] Zhang and Shasha. Simple fast algorithms for the editing distance between trees and related problems. *Siam Journal on Computing*, 18(6):1245–1262, 1989.