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

Shuyu Zheng[1], Ziniu Hu[1], Yun Ma[2]
[1]Peking University [2]Tsinghua University
{zhengshuyu,bull}@pku.edu.cn,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 UI 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 UI 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

## 1 INTRODUCTION

Well-designed manifestation of graphic user interfaces (GUI) 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 rather 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 designs that satisfy the content, and integrate these designs into the page. However, it is not only difficult to find appropriate pages for exhibiting the PUD's content, but also tedious to incorporate designs 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. We focus on the web GUI due to the cross-platform feature of the Web where both desktop and mobile devices can exhibit web GUI consistently according to Web standards on browsers.

FaceOff is designed driven by two ideas. First, according to a previous study [4], 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 content, they usually share templates that have the same functionality, so that the styles of these templates could be utilized to design new web page. Second, styles from the same web page should be well accordant with each other. If we could learn how the style 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 in the original web pages are harmonious with each other, so as to learn the embedding model. Finally, FaceOff can use the source HTML file of the PUD along with the selected combination of templates to compile and generate a web page with well-designed and harmonious style.

We next demonstrate the work flow of FaceOff. Given a PUD represented by its HTML and corresponding resources (images, videos, etc.), FaceOff segments the given HTML into multiple blocks, and retrieves the matched GUI templates for each block. Then FaceOff recommends multiple and diverse style combinations for each retrieved template. Finally, the developer can manually choose one 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.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the details of our proposed system, FaceOff. Section 4 demonstrates how our system works and the effectiveness of two key components: template retriever and style recommender. Section 5 concludes the paper.

## 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.

### 2.1 Design Mining

The data-driven approaches have been applied to design mining for revealing visual evolution and design patterns. Chen et al. [2] revealed the important landmarks in the aesthetic evolution of web pages from three aspects: information architecture models, visual flavor and the media composition. Doosti et al. [3] adopted convolution neural network 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. [7] revealed that people judge visual appeal as well as trust and usability of homepages consistently. Reinecke et al. [10] implemented image metrics to quantify colorfulness and visual complexity of web pages, and developed a model to predict perceived visual appeal.

## 2.2 Design Assisting Tool

The idea of assisting developers to conduct design has long been attached great importance. One line of work assists developers to write GUI code more efficiently. Kumar et al. [6] focused on the design mapping where the content of a web page can be transferred into a given template. Nguyen et al. [9] 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. [1] further advanced this idea of generating GUI code from image by adopting deep learning model. Another line of work explores the design style itself, and can assist generating better design. Kumar et al. [5] 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. [11] 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 a HTML subtree, 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 Cascading Style Sheets (CSS) code of each node in the HTML tree. It determines the visual representation of the content, such as width, height, color, background, and border type.

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

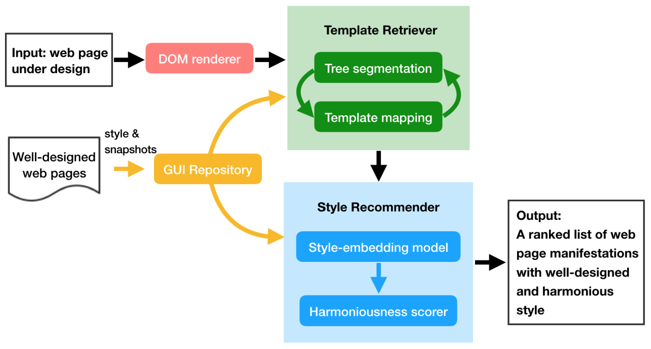


**Figure 1: The system architecture overview of FaceOff.**

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 range, making it impossible to enumerate the styles. Therefore, we resort to existing template of current well-designed web pages to narrow our optimized space, which leads to our approach.

Our basic idea is to find suitable and harmonious GUI templates from a large-scale web page repository. For a new page to be designed, we retrieve templates that fit the structure of the page, and recommend combinations of style 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; then construct a repository of web GUI. Given a page under design as input, FaceOff first renders the web page in a headless browser to acquire the DOM tree structure 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 designs for the developer to choose from.

## 3.3 GUI Repository

We collect a large scale of web pages with their HTML tree, style information (CSS rules of each node), and manifestation (screenshot image of the web page). Totally, we collect 15,491 different web pages from popular websites and professional design examples. 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 HTML tree, and use the subtree to identify a template. Afterwards, we group and index the templates with the same subtree.

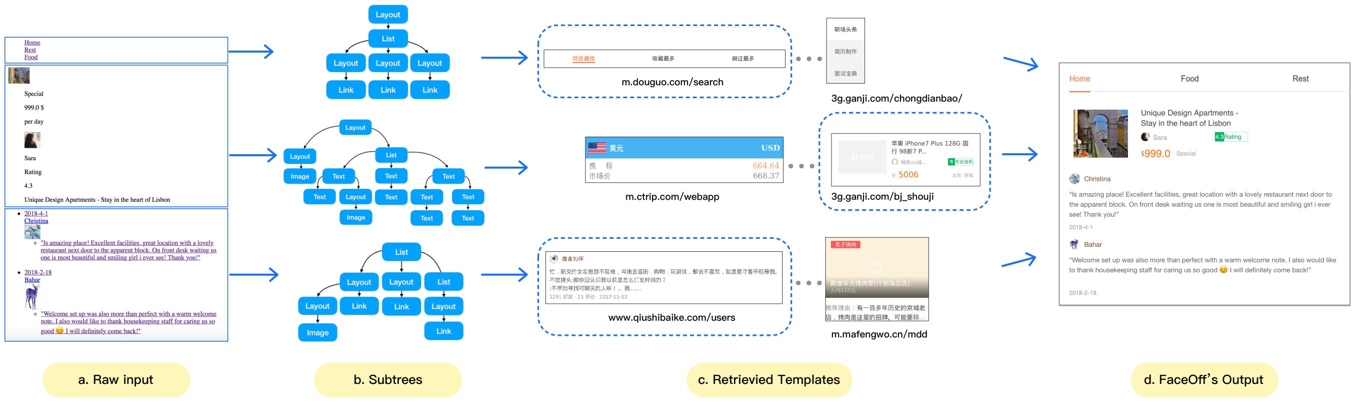


**Figure 2: The overall work flow of FaceOff, to transform a poor-designed hotel-booking page into a well-designed one.**

## 3.4 Template Retriever

Given the HTML of the page under design, FaceOff first segments it into multiple blocks based on the structural similarity to the templates in the GUI repository, and then matches the corresponding UI style templates for each block. We use the tree edit distance proposed by Zhang et al. [12] to define the similarity of HTML 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. Then, we try to segment 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 also retrieve all the templates that have been found during the segmentation.

## 3.5 Style Recommender

After the above operations, we can get a series of templates for the different blocks of the input HTML. The next step is to recommend template combinations whose styles are harmonious with each other. We achieve this by training an embedding model to map templates with similar style adjacently in the embedding space, and find some template combinations whose embeddings are close to each other.

In order to represent the UI style, we propose a style-embedding model, which uses a convolution neural network (CNN) to encode the compiled image of a template, and uses a pair-wise cosine similarity to model the score of style harmoniousness. To train such an embedding model, we assume that the templates from the same web page should be accordant in style with each other since web pages in our repository are well designed. Specifically, we extract the templates in the same web page as positive data, and randomly sample some templates in different web pages as negative data. In this way, the learned style-embedding model will map the templates in 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 templates, and can thus recommend a set

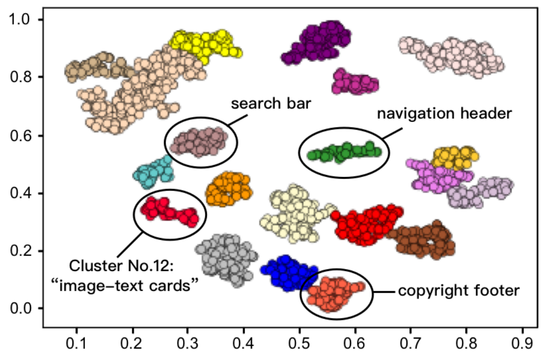


**Figure 3: 20 clusters of the sampled templates, each represents a functionality group.**

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

## 4 DEMONSTRATION

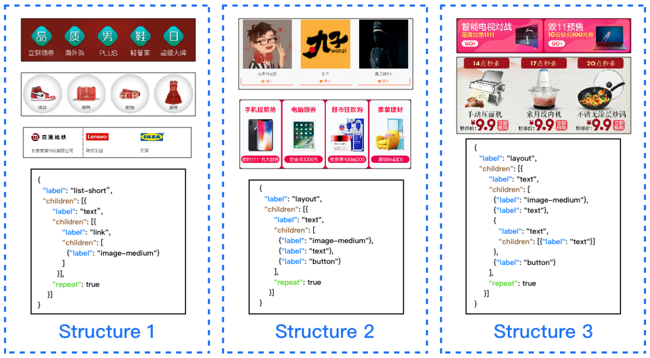
In this section, we first demonstrate how our system can assist manifestation design with a concrete example, and then illustrate the effectiveness of the template retriever and style recommender.

## 4.1 Overall Work Flow Demonstration

First, we use an illustrative example to show the overall work flow of our system[1]. As is shown in Figure 2, suppose a developer is going to design a web page for hotel booking. He has already prepared the content, including some links, a picture of the hotel, short information, room prices, and several comments from customers. The HTML file without any style information or CSS can be visualized as the part *a* in Figure 2.

Part *b* and *c* in Figure 2 show the intermediate results of FaceOff given the raw HTML input in *a*. FaceOff divides the HTML into subtrees, and retrieves templates with the structure of these subtrees, which are most similar to the input one in functionality.

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


```
{
  "label": "list-short",
  "children": [{
    "label": "text",
    "children": [{
      "label": "link",
      "children": [
        {"label": "image-medium"}
      ]
    }],
    "repeat": true
  }]
}
```

```
{
  "label": "layout",
  "children": [{
    "label": "text",
    "children": [
      {"label": "image-medium"},
      {"label": "text"},
      {"label": "button"}
    ],
    "repeat": true
  }]
}
```

```
{
  "label": "layout",
  "children": [{
    "label": "text",
    "children": [
      {"label": "image-medium"},
      {"label": "text"},
      {
        "label": "text",
        "children": [{"label": "text"}]
      },
      {"label": "button"}
    ],
    "repeat": true
  }]
}
```

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

Part *c* shows two template examples of each retrieved structure, which share the same structure, but exhibit different designs. From these template examples, we can figure out that the input web page conducts 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 development 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 [8] to visualize them. We next use K-means to find 20 clusters among these subtrees. As is illustrated 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. As is illustrated in Figure 4, 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 templates 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 is indicated

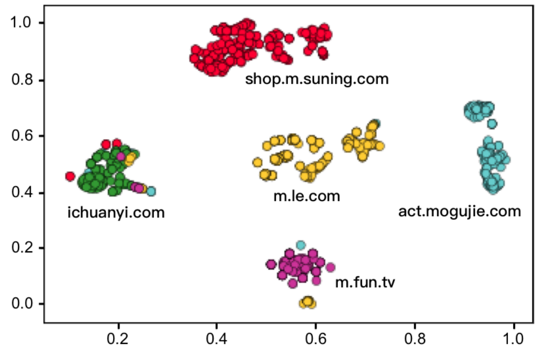


**Figure 5: Image embedding results of 5 sampled web pages. Color indicates their original web page.**

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

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.

## REFERENCES

[1] T. Beltramelli. pix2code: Generating code from a graphical user interface screenshot. *CoRR*, abs/1705.07962, 2017.
[2] W. Chen, D. J. Crandall, and N. M. Su. Understanding the aesthetic evolution of websites: Towards a notion of design periods. In *CHI Conference*, pages 5976–5987, 2017.
[3] B. Doosti, D. J. Crandall, and N. M. Su. A deep study into the history of web design. In *ACM on Web Science Conference*, pages 329–338, 2017.
[4] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *Special Interest Tracks and Posters of the International Conference on World Wide Web*, pages 830–839, 2005.
[5] R. Kumar, A. Satyanarayan, C. Torres, M. Lim, S. Ahmad, S. R. Klemmer, and J. O. Talton. Webzeitgeist:design mining the web. In *Sigchi Conference on Human Factors in Computing Systems*, pages 3083–3092, 2013.
[6] R. Kumar, J. O. Talton, S. Ahmad, and S. R. Klemmer. Bricolage: example-based retargeting for web design. pages 2197–2206, 2011.
[7] 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 (TOCHI)*, 18(1):1–30, 2011.
[8] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
[9] T. A. Nguyen and C. Csallner. Reverse engineering mobile application user interfaces with remaui (t). In *Ieee/acm International Conference on Automated Software Engineering*, pages 248–259, 2016.
[10] 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 *Sigchi Conference on Human Factors in Computing Systems*, pages 2049–2058, 2013.
[11] 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.
[12] Zhang and Shasha. Simple fast algorithms for the editing distance between trees and related problems. *Siam Journal on Computing*, 18(6):1245–1262, 1989.