# CSSSketch2Code: An Automatic Method to Generate Web Pages with CSS Style

Yi Han
East China Normal University
School of Data Science and Engineering
Shanghai,China
(86)13817482309
727781157@qq.com

Jun He
Harbin Institute of Technology
School of Mechatronics Engineering
Harbin,China
(86) 0451-86413111
hejun_hit@126.com

Qiwen Dong
East China Normal University
School of Data Science and Engineering
Shanghai,China
(86)021-62224960
qwdong@dase.ecnu.edu.cn

## ABSTRACT
With the constantly increasing scale of the Internet and the users, the Internet applications have higher demands on the front-end web pages. Some web pages have single lattice structure and a relatively fixed HTML code template, which can be automatically generated. There have been research abroad using the deep learning on the task of automatically generating the web pages. However due to the basic encoder-decoder model adopted, the generalization ability of the model is not very robust. In this paper, we propose a novel method based on object detection and attention mechanism to automatically generate a web page with CSS style information. We use object detection to extend the original problem, which makes the model possible to detect the CSS style contents in the web page. Meanwhile we use attention mechanism to strengthen the model. Finally we propose our own dataset, based on which the experiment results show that method we proposed outperforms other existing methods.

## CCS Concepts
• **Computing methodologies → Scene understanding**

## Keywords
Object detection; attention mechanism; convolutional neural network; bidirectional long short term memory; CSS; front-end pages automatic generation

## 1. INTRODUCTION
With the coming of the web times, every kind of web application bloom together. Our aim is to design and implement the straightforward and attractive front-end pages rapidly. Some web pages have single lattice structure and don't have so much complex logic. The manual transformation from web pages into HTML code is time-consuming, making the transformation possible being the automation process. In this paper, we propose a method to automatically transform the HTML code from web pages based on Deep Learning in order to save the time cost consuming on the task. We can achieve the expected effect after the finetune of the automatically generated HTML code. The process to transform the HTML code from web page can be viewed as the problem of image caption. In most image caption problems, given an image, the task is to output a descriptive sentence of the image. In our problem, the task is to output HTML code given a web page. However there are few research on the problem. The existing method is to simply interpret a web page as a page with lattice structure and output the HTML code through CNN and LSTM. Usually a complete web page is consisted with elements like HTML, CSS and JS. However the above method only consider the lattice structure and the characters of the web page, ignoring CSS style information. On the basis of the above method, object detection is used to detect the lattice structure, characters and controls in a web page, making it possible to consider the CSS style information such as the button's color, the character style and so on. In order to make our model more robust, attention mechanism is used to make the model focus on a specific area at one point, ignoring other useless information. Because there are few dataset for the problem of automatically generating the web pages with CSS style, we also manually develop a dataset for such task. The main contribution of the article are as follows:

- In order to make model detect the CSS style information in a web page, object detection is used to consider the CSS style of web pages.
- Attention mechanism is used to let model focus on a specific area at one point and make model more robust.
- A new dataset is proposed to support the new problem we propose.


*ICAAI 2018*, October 6-8, 2018, Barcelona, Spain

ACM ISBN 978-1-4503-6583-3 /18/10…$15.00
DOI: https://doi.org/10.1145/3292448.3292455

## 2. RELATED WORK
The problem of automatically generating the web pages with CSS style information can be divided into two sub problems: image caption and object detection. Image caption's task is to output the corresponding descriptive sentence of the given image, which is the main task of our problem. Object detection's task is to detect the CSS style information from web pages, serving as a supplementary. So in addition to the introduction of the progresses on the problem, image caption and object detection will be introduced as well.

### 2.1 Automatic Generation of the Web Page
There are few research studying on the problem of automatically generating the web page. As we know, there are only two works on it: Pix2Code [1] and Sketch2Code [2]. The input of the Pix2Code is a web front-end page and the output is its corresponding HTML code. The architecture of Pix2Code is encoder-decoder. There exists two problems in Pix2Code. First, the model is not robust owing to the architecture of ordinary

encoder-decoder, without the consideration of using attention mechanism. Second, usually we write the HTML code first and then get web front-end page according to the code. However Pix2Code outputs the HTML code according to its input of web front-end page, which is uncommon in daily life. For the second problem, Sketch2Code replaces the input of web front-end page with web manuscripts, making up for the disadvantage of Pix2Code. However it still doesn't solve the first problem. So on the basis of the existing research, attention mechanism is used to make the model more robust. Meanwhile object detection is used to detect the CSS style information in a web page, greatly extending the existing problem.

## 2.2 Image Caption

MS Captivator [3] detects the entity sets involved in an image through a visual detector, generates multiple sentences and finally re-ranks them to choose a sentence with highest rank as result. NIC [4] uses encoded-decoder architecture. The encoder uses convolutional neural network to encode the image and the decoder uses the long short-term memory. However, the originally accepted information will vanish gradually with the forward of the time step. gLSTM [5] uses three kinds of different semantic guiding information as the input of LSTM at every time step. It improves NIC but it doesn't consider that the semantic guiding information at every time step should be different. Sentence-

condition [6] combines image feature and text feature as the input of gLSTM and applies the method of text-conditional on image feature, making the input information different at every time step.

## 2.3 Object Detection

The object detection involves two branches of methods. One is called R-CNN branch. The typical methods are as follows: R-CNN [7] uses selective search to create ROIs (region of interests), gets feature vectors through conducting convolutional operation on each ROI and finally feeds vectors into SVM [8] to determine whether there exists object or not. R-CNN leads the CNN into object detection creatively. However its detecting time is too long because of the fact that every ROI has to experience a convolutional operation. Fast-RCNN [9] conducts a convolutional operation on the image instead of each ROI. Because it decreases the time of convolutional operation, its detecting time is greatly less than R-CNN's. However the use of selective search method becomes the bottleneck of the algorithm. In order to replace the selective search method, Faster-RCNN [10] uses RPN (region proposal network) and further speeds up the detecting time. For the greater accuracy of the object detection regions. Mask-RCNN [11] adds a convolutional layer to generate the image mask on the basis of Faster-RCNN. Another branch method is called end-to-end. YOLO [12] splits the whole image into S*S grids. Each grid will output the bounding box, confidence and class probability vector of the target involved in. Finally NMS(Non-Maximum Supression) algorithm is used to get rid of the targets which have been duplicatedly detected. SSD [13] uses multiple convolutional layers and conducts the object detection on each of them.

# 3. METHOD

The main task of our problem is: given a web script and output the corresponding HTML code. HTML code can be represented by DSL [14] (Domain-Specific Languages), which is consisted with a string of pseudo HTML labels $V = (v^{<1>}, v^{<2>} \dots \dots v^{<T>})$. DSL language can be transformed to HTML code through compiler and different DSL language corresponds to its own web page. To take CSS style information into account when automatically generating web pages, a web page can be viewed from two aspects: the whole lattice structure and the CSS style contents. The whole lattice structure decides the layout of a page and CSS style contents decides the detail styles of a page. The total process of our method is as follows: First given a web script, object detection method is used to detect CSS contents in a web page and corresponding CSS pixel matrix can be achieved. CNN is then used to encode the original web script matrix as well as the CSS pixel matrix to get P and Q, which are feature vectors of the above two matrixes under convolution operations. P and Q are combined as the expression of the original web script image $(P, Q)$.Next, for the time $t \in [1, T]$: Bi-LSTM is used to encode the DSL code $v^{<t>}$ as the language expression $c^{<t>}$, $(P, Q)$ and $c^{<t>}$ are combined to get the combined expression$(P, Q, c^{<t>})$, attention mechanism is used on it, Bi-LSTM is used to decode $(P, Q, c^{<t>})$ and DSL code $v^{<t+1>}$ can be achieved. After T cycles of the above processes, HTML code can be transformed from DSL code $v^{<T>}$. We will introduce our method following this line. The total proposed method is shown in figure 1.

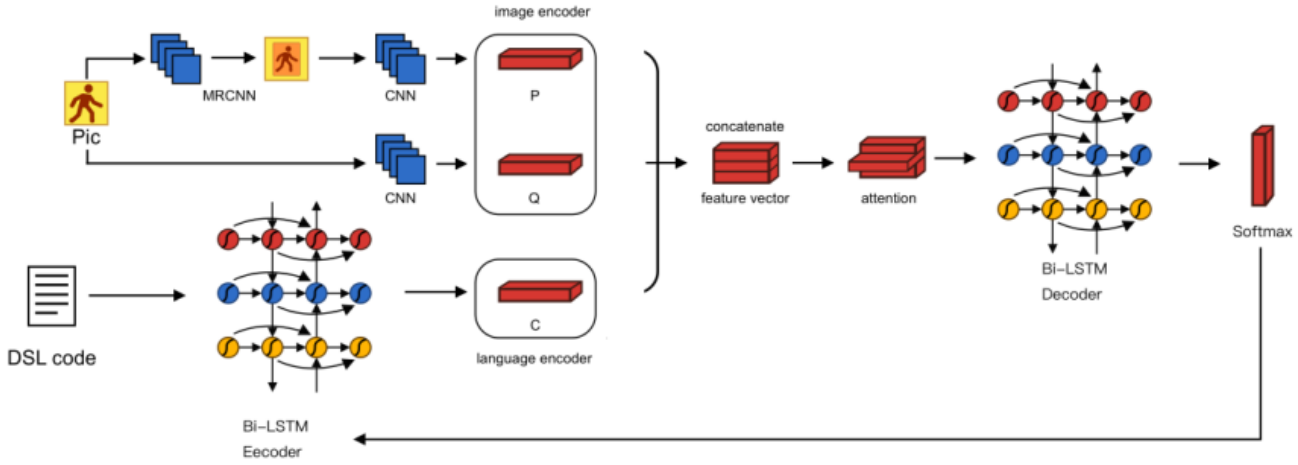
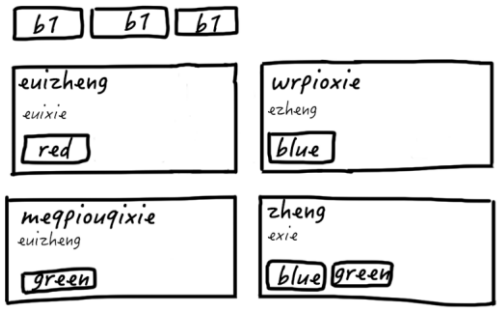



**Figure 1. Architecture of the proposed method.**

## 3.1 CSS Pixel Matrix

In order to consider the CSS style contents when generating HTML codes, object detection is used to detect the CSS contents from original web scripts as CSS pixel matrix. Here Mask-RCNN [11] is a kind of choice . Mask-RCNN feeds the original image to CNN to get the feature map, uses RPN on it to select the ROIs, fixes them to the same size through ROI layers and puts them into fully connected layer. Finally classification result, location and mask of the image can be achieved. The original web scripts and the CSS pixel matrix generated by Mask-RCNN are shown in figure 2.




(a) Original web script


(b)CSS pixel matrix generated through Mask-RCNN

**Figure 2. Original web script and its detailed CSS style content(CSS pixel matrix). In original web script, the characters of "b1", "red", "green", "blue" represent the color of button, the characters which end up with "zheng" or "xie" represent the font style.**

## 3.2 Encoder

Encoder is used on the image and text to extract the result as features. The encoder has three inputs: original web script matrix, CSS pixel matrix generated by Mask-RCNN and its corresponding HTML code. CNN is used to encode the original web script matrix and CSS pixel matrix, Bi-LSTM [15] is used to encode the HTML code.

### *3.2.1 Image Encoder*

As for the image, traditional convolutional neural network, CNN, which is an effective feature extractor, is used to encode. It can learn high order feature from images easily. CNN learns the whole layout feature and the detailed style content from original web script matrix and CSS pixel matrix. After convolutional operation two matrixes $P$ and $Q$ can be achieved and combined as the expression of the original web script image$(P, Q)$. The main reason of this process is to consider the layout and CSS style in a web script simultaneously.

### *3.2.2 Language Encoder*

HTML code is first transformed into DSL code. DSL code is consisted with a string of pseudo HTML labels $V = (v^{<1>}, v^{<2>} \dots \dots v^{<T>})$ and every DSL language corresponds to its own web page. The task of language encoder is: given a DSL label $v^{<t>}$ at time $t$, word embedding [16] is used to get the embedding expression $v_{emb}^{<t>}$ and $v_{emb}^{<t>}$ is encoded to get DSL expression $c^{<t>}$. Finally the image expression$(P, Q)$ and $c^{<t>}$ are combined to get combined expression $(P, Q, c^{<t>})$, which is used to transform into the DSL label $v^{<t+1>}$ at time $t + 1$. The general method is to use a encoder-decoder model consisted with RNN (Recurrent Neural Network) and DSL label $v^{<t>}$ can be viewed as the input of RNN. The Encoder-Decoder architecture is shown in figure 3. However there exists two problems in RNN. First, the location of the word to be predicted maybe far from its related information word and RNN is not good at capturing the distant information. Second, in some tasks, the output of a time is related with the previous and next information state and ordinary RNN can only capture the previous information. So Bi-LSTM is used to replace the ordinary RNN. Compared to ordinary LSTM [17], Bi-LSTM involves forward CNN and backward CNN, which makes it possible for the model to get previous and later information at one time. Meanwhile Bi-LSTM uses the concept of the gate. It controls the flow of information through three gates: input gate $i_t$, forget gate $f_t$ and output gate $o_t$. Input gate controls inflow of previous and later information, forget gate controls outflow of current information and output gate controls the output of current information.

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \quad (1)$$

$$i_t = \sigma(W_i[a^{<t-1>}, x^{<t>}] + b_i) \quad (2)$$

$$f_t = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \quad (3)$$

$$o_t = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_a) \quad (4)$$

$$c^{<t>} = i_t \cdot \tilde{c}^{<t>} + f_t \cdot c^{<t-1>} \quad (5)$$

$$a^{<t>} = o_t \cdot \tanh c^{<t>} \quad (6)$$

Where $\tilde{c}^{<t>}$ are candidate cells, $c^{<t>}$ are cells, $a^{<t>}$ are outputs, $\sigma$ are activation functions, $x^{<t>}$ are inputs, $b$ are biases and $W$ are weight matrixes.

## 3.3 Attention Mechanism

At time $t$, encoder is used to encode the original web script matrix, the CSS pixel matrix and the DSL label $v^{<t>}$ to get their expression $(P, Q)$ and $c^{<t>}$, then they are combined together and used to achieve the DSL label $v^{<t+1>}$ at time $t$+1. Considering that at different time $t$, the combined expression $(P, Q, c^{<t>})$ is different. We hope that model will change its attention when generating different DSL labels. So attention mechanism [18] is used on $(P, Q, c^{<t>})$,making model selectively focus its attention on a specific area.

$$e_i^{<t>} = f_{att}((P, Q, c^{<t>})_i, a^{<t>}) \quad (7)$$

$$\alpha_i^{<t>} = \frac{\exp(e_i^{<t>})}{\sum_i^M \exp(e_i^{<t>})} \quad (8)$$

$$s^{<t>} = \sum_{i=1}^{M} \alpha_i^{<t>} (P, Q, c^{<t>})_i \quad (9)$$

Where $f_{att}$ is the similarity matching function, $(P, Q, c^{<t>})_i$ is the ith feature vector of $(P, Q, c^{<t>})$ at time $t$, $a^{<t>}$ is the output of Bi-LSTM, $\alpha_i^{<t>}$ is the attention on $(P, Q, c^{<t>})_i$ and $s^{<t>}$ is the sum of different feature vector under their own attention, as the input of Bi-LSTM at time $t + 1$. The view of attention mechanism is shown in figure 4.

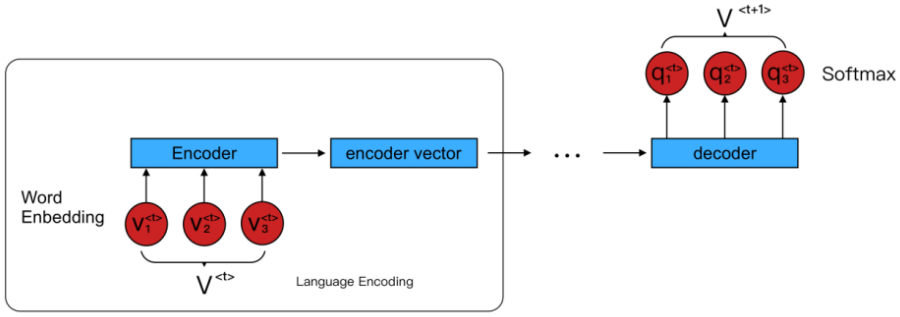



**Figure 3. Encoder-Decoder model: DSL label $v^{<t>}$ is embedded and fed into encoder to get the encoder vector, which consists the step of language encoding. After several steps, the encoder vector is fed into decoder and is transformed into the DSL label $v^{<t+1>}$.**

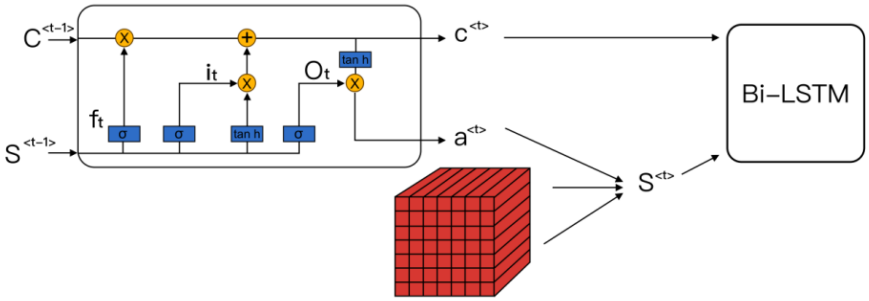



**Figure 4. Attention Mechanism: the red box in the pic represents the combined feature $(P, Q, c^{<t>})$. Attention mechanism is used to compute the similarity between the combined feature and hidden state $a^{<t>}$. The similarity is finally used to compute the weighted sum of combined feature, $s^{<t>}$ as the input of Bi-LSTM at next time step.**

## 3.4 Language Decoder

After attention mechanism be applied on the combined expression$(P, Q, c^{<t>})$ at time $t$, the language decoder is applied on it to get the DSL code expression $q^{<t+1>}$,which is used to predict the DSL label $v^{<t+1>}$ through softmax classifier.

$$p\left(y^{(i)} = j|x^{(i)};\ \theta\right) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \tag{10}$$

Where $\theta$ are parameters, $x$ is input, $y$ is output. Cross-entropy is used as the loss function of our model.

$$L(\tilde{y}, y) = -\sum_{j=1}^{c} y_j \log \tilde{y}_j \tag{11}$$

$$J(W, b) = \frac{1}{m}\sum_{i=1}^{m} L(\tilde{y}^{(i)}, y^{(i)}) \tag{12}$$

Where $y$ is ground truth, $\tilde{y}$ is predicted output, $L$ is single loss function and $J$ is whole loss function.

# 4. EXPERIMENTS

In this paper we propose a new method to automatically generate the web pages. We conduct several experiments to answer the following questions.

**RQ1** Does Mask-RCNN make model recognize the CSS style information in the web page?

**RQ2** Does the use of attention mechanism make the model focus its attention on useful information?

**RQ3** Does our model outperform baselines?

Experimental settings will be introduced first and then above questions will be answered, finally the choice of hyperparameters will be under discussion.

## 4.1 Experimental Settings

### *4.1.1 Dataset*

As we all know, there are few dataset for the problem of automatically generating the web pages with CSS style. So we develop our own dataset. We prepared 1500 datasets, 1050 for training set and 450 for test set. The dataset involves original web script, the CSS pixel matrix and its corresponding DSL label sequences. Photoshop and graphics tablet are used to draw the web scripts, which involves the whole layout, some basic controls like buttons, characters and the CSS style of these basic controls such as the font style and the button's color. They are shown in figure 2(a). The opensource software labelImg [19] is used to annotate the original web scripts. The annotations are characters and buttons in the script. With the dataset of original web scripts and annotated web scripts, an object detector can be trained to detect the CSS style content, as shown in figure 2(b). Finally DSL label sequences of web scripts are generated and these three kinds of data can be composed as dataset of our model.

### *4.1.2 Experimental Environment and Model Settings*

Python is used to implement the experimental code, Linux is used to run the code and GPU Tesla-K80 is used to train the model. The GPU size is 12 GB and training time lasts 12 hours. RMSprop is used to optimize the model parameters, the learning rate is set to 1e-4 and the method of learning rate decay is used to make the model converge more accurately. Dropout [20] is used to avoid the phenomenon of being overfit, cross-entropy is used to be the loss function and the callback function is set to let us get the most effective model.

### *4.1.3 Evaluation Metrics*

Here three kinds of evaluation metrics in machine translation are used: BLEU [21], ROUGE-L [22] and METEOR [23], The fourth metric: $SUM$ is the average of BLEU,ROUGE-L and METEOR.

#### *4.1.3.1 BLEU*

BLEU is used to compute the co-occurrence frequency of N grams in candidates and references.

$$\mathrm{Count}_{\mathrm{w_{i,j}}}^{\mathrm{clip}} = \min\left(\mathrm{Count}_{\mathrm{wi}}, \mathrm{Ref_j\_Count_{wi}}\right) \quad (13)$$

$$\mathrm{Count}^{\mathrm{clip}} = \max\left(\mathrm{Count}_{\mathrm{w_{i,j}}}^{\mathrm{clip}}\right) \ \mathrm{i} = 1,2,3 \ldots. \quad (14)$$

$$\mathrm{p_n} = \frac{\sum_{\mathrm{n\_grams}\,\epsilon\,\tilde{\mathrm{y}}} \mathrm{Count}^{\mathrm{clip}}(\mathrm{n\_grams})}{\sum_{\mathrm{n\_grams}\,\epsilon\,\tilde{\mathrm{y}}} \mathrm{Count}(\mathrm{n\_grams})} \quad (15)$$

$$\mathrm{BLEU} = \exp\left(\frac{1}{4}\sum_{\mathrm{i=1}}^{4} \mathrm{p_i}\right) \quad (16)$$

Where $Count_{wi}$ is the number of word $w_i$, $Ref_j\_Count_{wi}$ is the time of the word $w_i$ occurs in the $j$th reference. $Count_{w_{i,j}}^{clip}$ is the clip number of word $w_i$ in the $j$th reference. $Count^{clip}$ is the max clip number of word $w_i$ in all references. $p_n$ is the $BLEU_n$ metric based on N-gram and $BLEU$ is the average score of n $BLEUs_n$.

#### *4.1.3.2 METEOR*

In order to make up for the disadvantage of BLEU metric, METEOR considers the metric of recall and uses the weighted harmonic mean based on single-precision.

#### *4.1.3.3 ROUGE-L*

ROUGE is a kind of similarity matching method based on recall. One of the ROUGE metric, ROUGE-L, uses the method of LCS(Longest Common Subsequence) to compute.

$$\mathrm{R_{lcs}} = \frac{\sum_{\mathrm{i=1}}^{\mathrm{u}} \mathrm{LCS_U(r_i,c)}}{\mathrm{m}} \quad (17)$$

$$\mathrm{P_{lcs}} = \frac{\sum_{\mathrm{i=1}}^{\mathrm{u}} \mathrm{LCS_U(r_i,c)}}{\mathrm{n}} \quad (18)$$

$$\mathrm{F_{lcs}} = \frac{(1+\beta^2)\mathrm{R_{lcs}P_{lcs}}}{\mathrm{R_{lcs}}+\beta^2\mathrm{P_{lcs}}} \quad (19)$$

The measurement of ROUGE-L is F-value. $\mathrm{LCS_U(r_i, c)}$ is the length of longest common subsequence between each sentence $\mathrm{r_i}$ in the references and sentence set c of candidates. m is the number of words involved in references and $n$ is the number of words involved in candidates. $\beta$ is a hyperparameter.

#### *4.1.3.4 Average Weighting Metric: SUM*

Normalize the above three metrics and average them to get the average weighting metric: $SUM$.

$$\mathrm{SUM} = \frac{1}{3}(\mathrm{BLEU} + \mathrm{METEOR} + \mathrm{ROUGE\_L}) \quad (20)$$

### *4.1.4 Baseline*

Owing to the fact that Sketch2Code is familiar with our problem, it is taken as our model's baseline. The architecture used by Sketch2Code is consisted with CNN and LSTM. On this basis we propose four different model architectures, they are as follows:

- **CSSSketch2Code-V1(CNN+Bi-LSTM)**.The ordinary LSTM is replaced with Bi-LSTM to make model can consider previous and later information at one point.
- **CSSSketch2Code-V2(CNN+LSTM+Mask-RCNN)**.Object detection is used to make model detect the CSS style information in a web page.
- **CSSSketch2Code-V3(CNN+LSTM+Attention)**.Attention mechanism is used to make model selectively focus on some useful information when generating HTML code.
- **CSSSketch2Code-V4(CNN+Bi-LSTM+Attention+Mask-RCNN)**.Make model more robust with the combined advnatages of the above three.

## 4.2 Comparison between Different Models

The comparative experiment results of the five models are shown in Table 1 and we will answer the above questions according to Table 1.

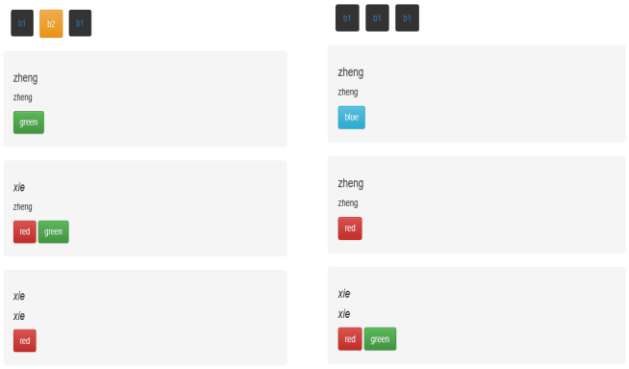
### *4.2.1 RQ1: Does Mask-RCNN Make Model Recognize the CSS style Information in the Web Page?*

Figure 5 shows the original web page script, ground truth and web pages generated by different algorithms. Figure 5(a) is the original web script. Figure 5(b) is the web page corresponding to Figure 5(a). Figure 5(c) is the web page generated by CSSSketch2Code-V2 and figure 5(d) is the web page generated by Sketch2Code. We can find that when applying Sketch2Code, the model only have a knowledge of the whole lattice structure of a page, ignoring the CSS style content. Compared to it, CSSSketch2Code-V2 can capture the CSS style information more accurately, such as top button's color and the font style in web page. Although the model cannot predict perfectly after adding MRCNN, the effect is obvious compared to Sketch2Code.

**Table 1. Algorithm comparison under four metrics**

| Algorithm | BLEU | METEOR | ROUGE-L | SUM |
|---|---|---|---|---|
| Sketch2Code | 0.669 | 0.493 | 0.782 | 0.648 |
| CSSSketch2Code-V1 | 0.669 | 0.500 | 0.788 | 0.652 |
| CSSSketch2Code-V2 | 0.683 | 0.503 | 0.785 | 0.657 |
| CSSSketch2Code-V3 | 0.672 | 0.518 | 0.785 | 0.658 |
| CSSSketch2Code-V4 | 0.679 | 0.513 | 0.783 | 0.658 |

(a) Original web script (b) web page corresponding to the script



(c) web page generated by CSSSketch2Code-V2 (d) web page generated by Sketch2Code

**Figure 5. web scripts, ground truth pages and web pages generated by different algorithms.**

### *4.2.2 RQ2: Does the Use of Attention Mechanism Make the Model Focus Its Attention on Useful Information?*

As shown in Table 1, CSSSketch2Code-V3 outperforms Sketch2Code in almost all metrics. When generating HTML code such as the label <button>, CSSSketch2Code-V3 will focus more attention on the pixels of button area and ignore other useless information. However Sketch2Code will not be concerned with this information and thus have a low effect.

### *4.2.3 RQ3: Does Our Model Outperform Baselines?*

Table 2's results show that when we use CSSSketch2Code-V1 and change the LSTM into Bi-LSTM, the metric *SUM* increases 0.4% compared to Sketch2Code. Bi-LSTM can let model know the previous and after information at one point and LSTM cannot do this. When using CSSSketch2Code-V3, the metric *SUM* increases 1%. The model uses attention mechanism, which lets the model increase the weights of useful information and decrease the weights of useless information when generating HTML code. When using CSSSketch2Code-V2 and CSSSketch2Code-V4, the metric *SUM* increases about 1%. They add object detector to the model and let model detect the whole lattice structure and CSS style content in a web page, which make the HTML code generated automatically more accurate. All of these results state that our methods really outperform the baseline's methods.

## 4.3 Hyperparameter Settings

### *4.3.1 The Analysis of Conv Layers*

Different number of layers will lead to different models. We carry out seven check experiments and the experimental results are shown in figure 6. The experimental results show that when we take 7 as the number of conv layers, the model has the best effect on the test set and the metric SUM can reach 0.657. We can observe that when the conv net is shallow(eg.4 layers), the model cannot greatly capture the feature of input and lead to underfit. With model being deeper, it can have a better knowledge of input data and the effect gradually increases. Finally when the conv layer is too deep, the model become overfit and makes the effect decrease again.

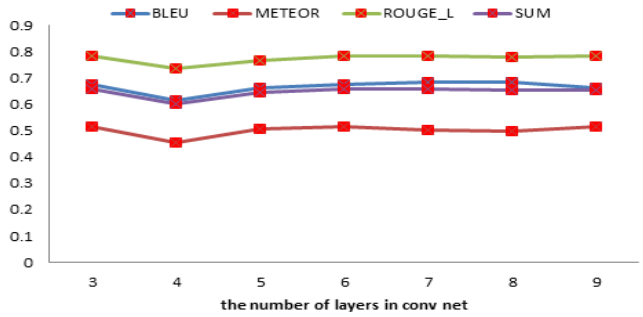



**Figure 6. The relationship between the number of layers and each metric.**

### *4.3.2 The Analysis of MRCNN's Accuracy*

If we add the object detector to the original architecture CNN+LSTM, the model can detect the CSS style content in a web page. The accuracy of MRCNN will affect the model result. We carry out five check experiments under the MRCNN's accuracy of 0.3, 0.4, 0.6, 0.7 and 1. The experimental results are shown in Table 2. The experimental results show that the accuracy of model will increase with the advance of object detector.

**Table 2. Comparison between object detectors with different accuracies**

| Algorithm | BLEU | METEOR | ROUGE-L | SUM |
|---|---|---|---|---|
| CNN+LSTM+0.3MRCNN | 0.608 | 0.500 | 0.768 | 0.625 |
| CNN+LSTM+0.4MRCNN | 0.623 | 0.468 | 0.738 | 0.610 |
| CNN+LSTM+0.6MRCNN | 0.673 | 0.507 | 0.788 | 0.656 |
| CNN+LSTM+0.7MRCNN | 0.671 | 0.509 | 0.785 | 0.655 |
| CNN+LSTM+1.0MRCNN | 0.683 | 0.503 | 0.785 | 0.657 |

## 5. CONCLUSION

In this paper, we use two methods of object detection and attention mechanism to automatically generate the web front-end page. By object detector, model can consider the CSS style information in the page, greatly extending the original problem. At meantime to make model more robust we use attention mechanism, letting model selectively focus its emphasis on some useful information. The technique of deep learning makes this problem become true and we believe there will emerge more techniques to support the development of this task.

## 6. REFERENCES


[1] Beltramelli T. pix2code: Generating Code from a Graphical User Interface Screenshot[J]. 2017.

[2] https://github.com/ashnkumar/sketch-code.

[3] Fang H, Platt J C, Zitnick C L, et al. From captions to visual concepts and back[C]// Computer Vision and Pattern Recognition, IEEE, 2015:1473-1482.

[4] Vinyals O, Toshev A, Bengio S, et al. Show and tell: A neural image caption generator[C]// Computer Vision and Pattern Recognition, IEEE, 2015:3156-3164.

[5] Jia X, Gavves E, Fernando B, et al. Guiding Long-Short Term Memory for Image Caption Generation[J]. 2015.

[6] Zhou L, Xu C, Koch P, et al. Image Caption Generation with Text-Conditional Semantic Attention[J]. 2016.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[8] Suykens J A K, Vandewalle J. Least Squares Support Vector Machine Classifiers[M]. Kluwer Academic Publishers, 1999.

[9] Girshick R. Fast R-CNN[J]. Computer Science, 2015..

[10] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.[J]. IEEE Trans Pattern Anal Mach Intell, 2017, 39(6):1137-1149.

[11] He K, Gkioxari G, Dollar P, et al. Mask R-CNN[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, PP(99):1-1.

[12] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[J]. 2015:779-788.

[13] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector[C]// European Conference on Computer Vision. Springer International Publishing, 2016:21-37.

[14] Fowler M. Domain Specific Languages[M]. Addison-Wesley Professional, 2010.

[15] Chiu J P C, Nichols E. Named Entity Recognition with Bidirectional LSTM-CNNs[J]. Computer Science, 2015.

[16] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and their Compositionality[J]. Advances in Neural Information Processing Systems, 2013, 26:3111-3119.

[17] Graves A. Long Short-Term Memory[M]// Supervised Sequence Labelling with Recurrent Neural Networks. Springer Berlin Heidelberg, 2012:1735-1780.

[18] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate[J]. Computer Science, 2014.

[19] https://github.com/tzutalin/labelImg.

[20] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. Computer Science, 2012, 3(4):págs. 212-223.

[21] Papineni K, Roukos S, Ward T, et al. IBM Research Report Bleu: a Method for Automatic Evaluation of Machine Translation[J]. Acl Proceedings of Annual Meeting of the Association for Computational Linguistics, 2002, 30(2):311--318.

[22] Flick C. ROUGE: A Package for Automatic Evaluation of summaries[C]// The Workshop on Text Summarization Branches Out. 2004:10.

[23] Banerjee S, Lavie A. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments[J]. 2005:228-231.