Hierarchical segmentation of graphical interfaces for Document Object Model reconstruction

Cătălin F. Perțicaș [1] Mihai S. Baba [1] Homa Davoudi [1] Răzvan V. Florian [1]

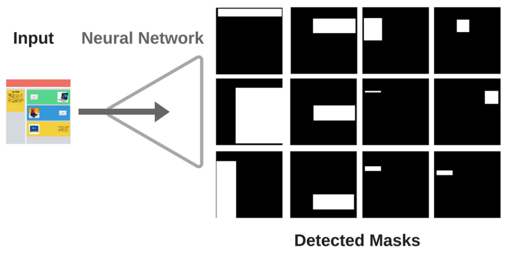

Figure 1. Detection of graphical elements using a deep convolutional neural network.

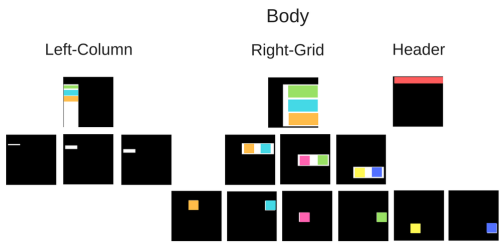

Figure 2. Tree reconstruction based on the hierarchical segmentation.

1. Introduction

Recently, systems that generate source code from bitmap renderings of graphical user interfaces have started to be developed (Beltramelli, 2017). Here we experiment with a new approach that is able to reconstruct the hierarchical structure of elements displayed in the graphical layout. As a case study, we take bitmap snapshots of HTML pages and infer from those the document object model (DOM), which can be then converted straightforwardly to HTML.

We train a specialized deep convolutional neural network, Faster R-CNN (Ren et al., 2015), for segmenting a graphical rendering into bounding boxes (masks) of typical graphical elements and for classifying these elements. This first module of our system outputs the masks and classes for the identified elements (Fig. 1). These masks are further processed, with a second module, to re-create the hierarchical structure and the corresponding source code that generated the graphical layout (Fig. 2).

Differently from Beltramelli (2017), our approach decouples the two learning tasks: (1) inferring the structure of the graphical interface, and (2) mapping the hierarchical structure of elements to source code, thus allowing a better understanding of how the two modules learn. Moreover, with the help of segmented masks generated by the first module, the code generating module can make more informed decisions. In this study, we focus on a machine learning implementation of the first task. The second task is performed with a classical algorithm that creates a tree from the detected masks.

[1]Romanian Institute of Science and Technology, Cluj-Napoca, Romania. Correspondence to: Cătălin F. Perțicaș <perticas@rist.ro>, Răzvan V. Florian <florian@rist.ro>.

Published at the ICML workshop Neural Abstract Machines & Program Induction v2 (NAMPI) — Extended Abstract, Stockholm, Sweden, 2018. Copyright 2018 by the author(s).

2. Methodology

The experiments are performed on a synthetic dataset which is defined through a domain specific language (DSL). We have a total of 12 HTML elements to detect and include in the reconstructed DOM. We use the Selenium WebDriver to retrieve screenshots of our synthetically generated HTML pages. For acquiring the ground-truth masks of individual elements, we execute Javascript through the WebDriver.

2.1. Detection of graphical elements

For the specific case of HTML pages, we need a type of detection and segmentation of the elements which allows overlaps and hierarchies. It is very common to have nested elements, such as an image inside a row `div`, inside a grid `div`. The Faster R-CNN (Ren et al., 2015) can handle such cases with a Region Proposal Network (RPN). A RPN is a convolutional network which learns to predict multiple regions of interest (rectangular masks) for a given image. The model outputs the coordinates and an "objectness" score for each region. This is done via regression on region coordinates and classification for determining if the data in the region is an object or background. A second model, the

IoU threshold	Mean average precision (%)
0.70	**99.5**
0.75	99.4
0.80	99.2
0.85	97.7
0.90	85.3
0.95	45.5

Table 1. Mean average precision of segmentation as a function of IoU thresholds.

Overlap threshold	Accuracy (%)
0.70	**92.8**
0.75	**92.8**
0.80	**92.8**
0.85	91.8
0.90	86.7
0.95	63.6

Table 2. Tree reconstruction accuracy as a function of different overlap thresholds.

Fast-R-CNN detector, shares the convolution weights with the RPN and uses its predictions for distinguishing between different classes of objects and for further detection refinements. We used an ImageNet pre-trained Faster R-CNN system. Then we trained it with 3 600 screenshots and validated on 400 other screenshots. All screenshots had an initial size of 900 $\times$ 1 200 pixels and were resized to 128 $\times$ 128 pixels. The number of epochs was 1 000.

For assessing the quality of the resulting segmentations and detections, we used the mean average precision over recall values. Thus, an average precision was computed for predicting the elements from a sample and then we reported the mean of these averages across samples. The precision was computed based on the Intersection over Union (IoU). Thus, if the IoU between a detected element and the ground-truth was larger than a threshold and the two elements had the same class, then it was considered a hit. The precision, as a function of the IoU threshold, is presented in Table 1.

2.2. DOM tree reconstruction

For mapping the masks detected at the previous step to a tree of elements, such as a DOM in the case of HTML, we have developed a simple conventional, heuristic algorithm. For all pairs of detected regions, we counted the number of pixels in the intersection of the two regions. If the intersection, measured as the percentage of the area of the smaller region, was larger than an overlap threshold, we considered the node of the smaller region to be a child of the node assigned to the larger region. This process leads to the creation of a directed acyclic graph. However, we would like to obtain a tree structure, so, if one node has several parent nodes, we only kept the edges to the parent node with the smallest area, thus creating a tree structure. Finally we sorted all children nodes by their position in the detected masks (top to bottom and left to right), such that we could compare the resulting tree with the ground-truth.

The performance was computed as either a hit or miss, the same way it was computed by Beltramelli (2017). The predicted and ground-truth trees needed to be identical in order to count as a hit. The accuracy of prediction as a function of different overlap thresholds is reported in Table 2. To compare our approach with the existing one (pix2code) (Beltramelli, 2017), we have run the pix2code model (using the code at `https://github.com/tonybeltramelli/pix2code`) on our synthetically generated dataset and we got an accuracy of 80.0%, significantly lower than our best result of 92.8%.

3. Conclusions

The segmentation method reports an average precision of above 97% for an IoU up to 85% (Table 1). The DOM reconstruction accuracy was more than 92% (Table 2), significantly better than the state-of-the-art. Therefore, the hierarchical structure behind graphical interfaces can be reconstructed with high precision using our approach. The DOM can be then converted in a straightforward way to HTML code.

In future work, we will extend these preliminary results to study the performance of our system on more general datasets, including real websites. Our system can be used in the future as part of a software that automatically generates websites from user-provided mockups.

Acknowledgements

This work was supported by the European Regional Development Fund and the Romanian Government through the Competitiveness Operational Programme 2014–2020, project ID P_37_679, MySMIS code 103319, contract no. 157/16.12.2016.

References

Beltramelli, Tony. pix2code: Generating code from a Graphical User Interface screenshot, 2017. URL `https://arxiv.org/abs/1705.07962`.

Ren, Shaoqing, He, Kaiming, Girshick, Ross, and Sun, Jian. Faster R-CNN: Towards real-time object detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pp. 91–99, 2015.