Structural Learning for Web Design

Maxine Lim MAXINEL@STANFORD.EDU
Arvind Satyanarayan ARVINDSATYA@CS.STANFORD.EDU
Cesar Torres CTORRES7@STANFORD.EDU
Stanford University, Stanford, CA 94305 USA

Abstract

Recursive neural networks (RNNs) have been successful for structured prediction in domains such as language and image processing. These techniques imposed structure onto sentences or images for more effective learning. However, in domains such as Web design, structure is explicitly embedded in the Document Object Model, so structured prediction can be done using the natural hierarchy of Web pages. This characteristic also allows for features to be inserted at each node of the RNN rather than only using features at the leaves. We show that for structural label prediction, this technique outperforms the baseline by 8%, though it performs poorly for style labels for reasons that are likely a result of suboptimal data.

1. Introduction

Design is difficult to quantify, but in domains such as Web design, it can be represented digitally in the form of HTML, CSS, and its external resources. As a result, we can leverage this data and enable machines to interpret design elements. For example, given a Web page, can we automatically decide if this page looks *minimal* or *modern*? How can we best encode these qualitative characteristics in data representations?

In this paper we investigate using structural prediction techniques to learn design descriptors for Web pages. Based on previous work in Web design and machine learning, we adapt a method using recursive neural networks (RNNs) to more accurately represent Web design by embeding page structure into the feature representations of Web elements. We use these structure-embedded feature representations to train classifiers for two classes of design descriptors. Our results show that structure is indeed an important factor to consider when applying machine learning to Web design, and our technique is able to outperform the baseline.

2. Background

2.1. Motivation

Automatic prediction of design descriptors can enable a new class of tools for Web design. Structural semantic labels, e.g., *sidebar* or *comment* can enable tools that automatically manipulate Web content such as page-to-page or page-to-mobile retargeting. Leveraging style and content-based keywords in design search can help inspire and direct design work by helping designers find relevant inspiration. Currently, finding this inspiration is limited to manually curated galleries or template libraries. These search techniques do not provide flexible or scalable navigation of the full Web.

2.2. Previous Work

From previous work we point out three important insights: first, training off-the-shelf binary SVM classifiers for structural semantics using crowdsourced labels with a set of 1,679 visual features is feasible and achieves a 76% average accuracy (Lim et al., 2012). Second, when applying machine learning to Web design, page structure is important (Kumar et al., 2011). And third, structured prediction using RNNs has proven successful in domains such as natural language and image processing, whose entities do not possess an explicit structure (Socher et al., 2011). Combining these insights, we adapt an existing structured prediction technique for a domain where structure is explicit, structure has been shown to be important in applying machine learning, and learning has been shown to be feasible.

3. Method

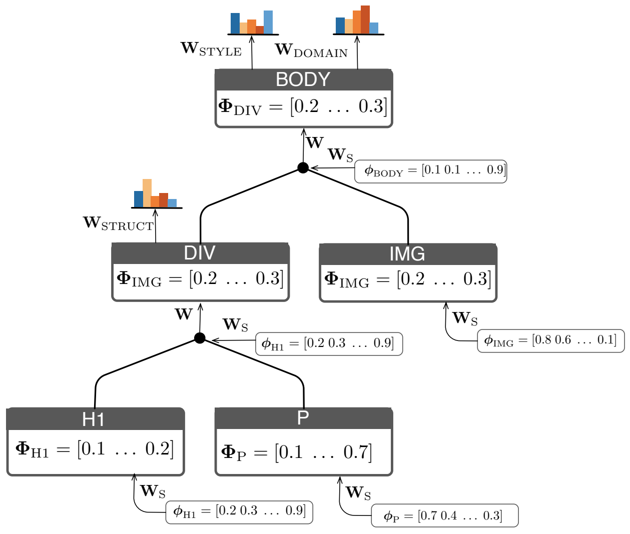


Figure 1. Our adapted RNN model that incorporates raw feature vectors at every node.

3.1. Our Model

Previous work used RNNs for structured prediction in domains such as language and image processing (Socher et al., 2011). Using the underlying DOM tree of web pages, we eliminate the need to develop the structural extraction techniques used in other domains. Furthermore, this explicit structure allows us to incorporate an additional set of features corresponding to the parent node at every neuron in the RNN. An overview of our model is shown in Figure 1.

3.2. Adaptation for Web Design

Our adaptation of RNNs for structured prediction involves modifying the feed-forward step of the algorithm. Typically, the activations for each node of the RNN is computed by:

$$\mathbf{p} = f\left(\mathbf{W}\begin{bmatrix}\mathbf{c_1}\\ \mathbf{c_2}\\ 1\end{bmatrix}\right),$$

where f is the sigmoid function; $\mathbf{c_1}, \mathbf{c_2}, \mathbf{p} \in \Re^{n\times 1}$; and, $\mathbf{W} \in \Re^{n\times(2n+1)}$.

In our technique, we apply the semantic transformation to the parent node:

$$\mathbf{c}_{parent} = f\left(\mathbf{W}_{sem}\begin{bmatrix}\mathbf{f}_{parent}\\ 1\end{bmatrix}\right),$$

where m is the number of raw features assigned , and $\mathbf{W}_{sem} \in \Re^{n\times(m+1)}$. (The bias term has been combined into all the parameter matrices.) We select f to be the tanh function rather than the sigmoid function because our normalized data values range from [-1,1] rather than [0,1].

We then incorporate the result into computing the activation:

$$\mathbf{p} = f\left(\mathbf{W}\begin{bmatrix}\mathbf{c_1}\\ \mathbf{c_2}\\ \mathbf{c_{parent}}\\ 1\end{bmatrix}\right),$$

As in previous work, we then add to each RNN parent node a softmax layer to predict class labels.

4. Data Preparation

4.1. Feature Set

Each of our examples is a Web element with its corresponding feature vector. These elements are derived from the Bento segmentation of the DOM tree, which segments visually salient blocks from Web pages (Kumar et al., 2011). The 1,713-dimensional feature vectors for each element include DOM-related properties (e.g. tree level, number of children), CSS attributes (e.g. font, color, size), and computer vision features (e.g. Gist descriptors). These Web elements along with their feature vectors are provided by the Webzeitgeist repository (Kumar et al., 2013).

4.2. Tree Binarization

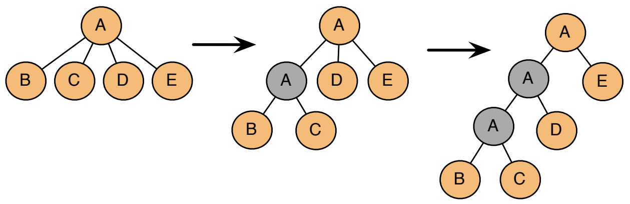


Figure 2. The tree binarization process, which adheres to Chomsky normal form.

Training RNNs required binarizing the DOM trees for each Web page. To maintain the parent-child relationships among the nodes, we constructed post-order trees from their Bento segmentations and binarized them using Chomsky normal form as shown in Figure 2.

This method requires that all product rules follow the form:

$$A \to BC$$
$$A \to \alpha$$
$$S \to \epsilon$$

Thus, if a node has more than two children, a new product rule would be derived of the form:

$$A \to BCD \Rightarrow A \to BE \text{ and } E \to CD$$

where E is a new intermediary dummy node which inherits its feature representation from its parent A.

The resulting trees were assigned a post-order identifier and stored within a NoSQL database along with information describing how to map the original node back to its parent identifier. This mapping is needed to reconstruct the tree for training and evaluation.

4.3. Label Collection

We collected two classes of labels with which to train RNNs: style and structural semantic. Style labels describe the design of the Web page, e.g., *minimal* or *elegant*, and structural semantic labels describe the purpose of each element on the page, e.g., *sidebar* or *comment*. While style labels are page-level descriptors, structural semantic labels describe an individual page node. We ran one study for gathering the page-level labels and another for gathering node-level labels. Overall we recruited over 1,300 US-based participants from Amazons Mechanical Turk and ODesk to apply over 35,000 domain, style, and structural semantic labels to over 4,000 Web pages. These pages were drawn from the Webzeitgeist design repository, which provides visual segmentations and page features for more than 100,000 Web pages (Kumar et al., 2013).

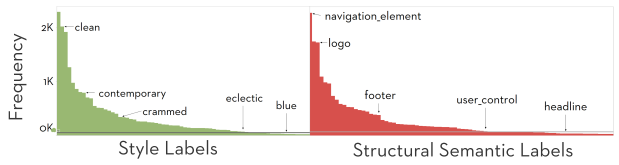


Figure 3. Heavy-tailed distributions of collected label sets.

4.4. Cleaning Data

Allowing users to enter labels in a free-form text field encouraged a wide variety of labels, but it also resulted in a heavily tailed frequency distribution, as shown in 3. Among the thousands of distinct labels, many only occurred one time or had effectively the same meaning as another label. These labels differed only by choice of delimiter, e.g., *eye − catching* and *eye_catching*, or word form, e.g., *religious* and *religion*. We applied a series of transformations to clean the raw data.

To clean the site type and style labels, we first trimmed each entry of extra punctuation at the ends, i.e., *simple_* became *simple*. We proceeded to split long compound labels by the underscore character only if each of the resulting terms had already occurred in our label set. For examine *simple_clean_minimal* would be split into *simple*, *clean*, and *minimal* since each had occurred in the set individually, but *black_and_white* would not, because *and* did not appear in our original label set. Next we merged labels with common stems derived from the Porter2 stemming algorithm, then manually merged labels such as *cartoonish* and *cartoon*. Finally, we removed labels that had only been applied once or only by one person. For the structural semantic labels, we only removed labels that had only been applied once or by a single individual.

Table 1. Number of total and distinct raw labels for all three label classes.

Label Type	Total Raw	Distinct Raw
Site Type	18,881	2,011
Style	24,021	1,718
Structural	21,995	2,657

Table 2. Number of total and distinct used labels for all three label classes.

Label Type	Total Used	Distinct Used
Site Type	17,883	65
Style	22,564	70
Structural	15,897	80

4.5. Label Results

The number of total and distinct labels from our study for the two label classes is shown in Table 1. Even after cleaning the data, there were still hundreds of distinct labels for each class, many of which would be difficult to for a person, much less a learning algorithm, to distinguish between. Furthermore, many labels only occurred a handful of times, producing too few examples to expect our RNN to learn. Therefore we selected the most frequent set of labels to learn, maintaining only the labels that occurred fifty or more times. This final filter reduced each set of labels to less than a hundred distinct elements, as shown in Table 2.

5. Learning

5.1. Unit Tests

Given the changes we introduced to the RNN framework, we constructed a set of simple unit tests to verify that gradient descent would converge correctly. These test cases involved simple tree structures of 3-10 nodes, 2 raw features and 1-3 label classes which were fed through the RNN with "gradient check" enabled. With this flag, the gradient descent library would calculate an expected gradient and compare it to the output of the RNN. Once our unit tests passed the gradient check, we had some level of confidence on the validity of our code.

5.2. Experimental Setup

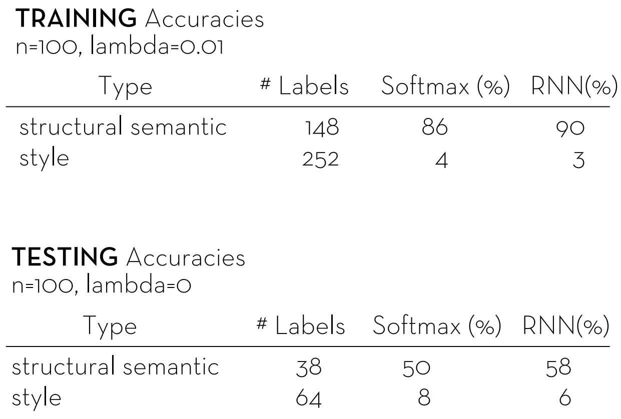
To train and evaluate the RNN, we constructed a hold-out set with 80% of the data used for training and 20% for testing. For a baseline comparison, we developed a "softmax only" classifier which was trained only on labeled nodes (i.e., ignored the structural hierarchy of a web page tree). After several rounds of training and testing, we determined two suitable metrics for judging the accuracy of label classifiers: cross entropy and whether ground truth labels were in the top N predicted labels, where N is the number of ground truth labels. The latter is a more human understandable measure of accuracy that remains robust against multiple co-occurring labels without penalizing predicted ordering.

5.3. Parameter Tweaking

Adjusting the parameters of the RNN significantly affected its performance. In particular, we tweaked two parameters: n, the number of hidden nodes in the RNN, beginning at 50 and in increments of 50; and λ, the regularization constant, by orders of magnitude starting at 0.0001. We found the optimal parameters, determined by the highest average test accuracy of labels, to be $n = 100, \lambda = 0.01$. On average, the RNN would take 15-20 hours to train. Thus due to time constraints, we were only able to adjust parameters for a limited set of permutations.

6. Results and Discussion

At the optimal parameters of $n = 100, \lambda = 0.01$, the average test accuracy of structural RNN-classifiers is 58% with *search* being the most accurately classified label at 96% and *user_control* at 0%. In comparison, the baseline average test accuracy is 50%. On the other hand, style labels perform significantly worse. The average test accuracy is 6% for RNNs compared to a baseline average of 8%. Many of the style labels were not accurately predicted whatsoever (0%) by either. However, *clean* and *modern* performed well under both conditions: 81% and 57% respectively for RNN; 96% and 81% for the baseline. These accuracies are listed in Figure 5.



TRAINING Accuracies
n=100, lambda=0.01

Type	# Labels	Softmax (%)	RNN(%)
structural semantic	148	86	90
style	252	4	3

TESTING Accuracies
n=100, lambda=0

Type	# Labels	Softmax (%)	RNN(%)
structural semantic	38	50	58
style	64	8	6

Figure 5. Average training and test accuracies for structural semantic and style labels. For structural labels, RNN outperforms the baseline.

Although the RNN-classified labels do perform, on average, better than the baseline, we expected the addition of structural information to make a larger impact. Moreover, while previous work obtained an average accuracy of 76% for structural labels using binary SVMs, our results are much lower. Granted, given that the data, the label set, and the metric are different, these numbers are perhaps unfairly compared.

However, one reason for this result can a sparse data set. We had many distinct labels, and the frequency was not evenly distributed among them. Labels that performed well tended to have more examples. Collapsing this set of labels even further, or collecting more data would help to reduce the skew in the data set.

Another problem with our data was the miscalculation of negative examples. Due to the manner in which we conducted our crowdsourced label collection, we did not have any *true* negative examples: if a node or page does not have a specific label, it does not imply that it is *not* in that class. Counting a lack of a label as a negative label is likely having a detrimental effect on the learning.

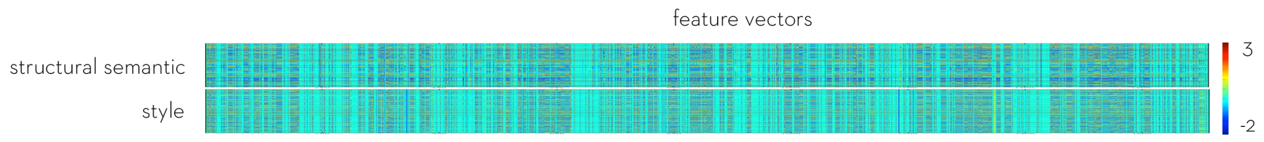


Figure 4. Heat map of raw features grouped by label for structural and style labels. Note the density difference between style and structural semantic feature vectors.

Our data set does not entirely account for why style label classifiers performed so poorly, especially compared to their structural counterparts. We hypothesize that our choice of softmax classifier is also negatively affecting style label accuracy. Typically, a node can only be labeled with any one of the structural labels. Thus, softmax is a good classifier to use here as all probabilities must sum to 1. But in the case of style labels, a page can be described with multiple labels, making softmax cause a "smearing" of predicted probabilities. Training multiple binary classifiers might be a better choice for non-disjoint labels.

Although we do not have optimal results, we have evidence that these descriptors can be learned and that structure is important in this learning. As shown in Figure 4, the heatmap visualizations of raw features grouped by label show clear banding, which implies that there are patterns to be learned for both structural and style labels. Additionally, RNNs do better than the baseline for structural labels by 8%, indicating that structure can further inform the learning.

7. Conclusion

We have implemented an adapted RNN algorithm for embedding structure into the feature representations for Web elements to predict design descriptors for Web design. We've shown that for the optimal parameter combination, our method is able to outperform the baseline by 8% for structural labels. Unfortunately overall our accuracies are low, and for style labels our classifiers consistently performed poorly. We point to parameter adjustment, a richer, better data set, and different classifers types for the different label classes as ways to improve our accuracies.

7.1. Future Work

Looking ahead we are re-running our experiments with a collapsed label set of approximately twenty labels per label class. By merging similar labels together (e.g. *header* and *heading*) and removing ones that were difficult for even humans to understand (e.g. *user_control*), we believe that a more accurate set of softmax classifiers will be learnt on a smaller label set.

We have also launched a new round of crowdsourced label collection to increase the density of our label set. In this study, participants are given the newly collapsed set of labels and are taken through our corpus page-by-page. For each page, participants are asked to select all style labels that apply. They are then taken through each structural label individually and asked to select all visual blocks that fit the label. With this design, we can be more confident that our dataset is complete: in particular, as participants see the whole set of labels, if a node or page does not have a particular label, this does indeed count as a negative example.

With style labels, we are investigating the effects of training multiple, independent logistic classifiers: one per label. The obvious caveat is that different style labels may not necessarily be independent: a page being labeled *clean* may also imply that it is *minimal*. Nonetheless, we believe this may be a first step towards addressing the issue of probability "smearing" discussed previously.

Acknowledgments

This work was done in collaboration with Ranjitha Kumar, Richard Socher, Jerry O. Talton, and Scott R. Klemmer.

References

Kumar, R., Talton, J. O., Ahmad, S., and Klemmer, S. R. Bricolage: Example-based retargeting for web design. In *CHI: ACM Conference on Human Factors in Computing Systems*, 2011.

Kumar, R., Satyanarayan, A., Torres, C., Lim, M., Ahmad, S., Klemmer, S. R., and Talton, J. O. Webzeitgeist: Design mining the web. In *CHI: ACM Conference on Human Factors in Computing Systems*, 2013.

Lim, M., Kumar, R., Satyanarayan, A., Torres, C., Talton, J. O., and Klemmer, S. R. Learning structural semantics for the web. Technical report, Stanford University, 2012.

Socher, R., Lin, C. Chiung-Yu, Ng, A. Y., and Manning, C. D. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.