A ML-based Approach for HTML-based Style Recommendation

Ryan Aponte
CMU

Ryan A. Rossi
Adobe Research

Shunan Guo
Adobe Research

Jane Hoffswell
Adobe Research

Nedim Lipka
Adobe Research

Chang Xiao
Adobe Research

Gromit Chan
Adobe Research

Eunyee Koh
Adobe Research

Nesreen Ahmed
Intel Labs

ABSTRACT

Given a large corpus of HTML-based emails (or websites, posters, documents) collected from the web, how can we train a model capable of learning from such rich heterogeneous data for HTML-based style recommendation tasks such as recommending useful design styles or suggesting alternative HTML designs? To address this new learning task, we first decompose each HTML document in the corpus into a sequence of smaller HTML fragments where each fragment may consist of a set of HTML entities such as buttons, images, textual content (titles, paragraphs) and stylistic entities such as background-style, font-style, button-style, among others. From these HTML fragments, we then derive a single large heterogeneous hypergraph that captures the higher-order dependencies between HTML fragments and entities in such fragments, both within the same HTML document as well as across the HTML documents in the corpus. We then formulate this new HTML style recommendation task as a hypergraph representation learning problem and propose an approach to solve it. Our approach is able to learn effective low-dimensional representations of the higher-order fragments that consist of sets of heterogeneous entities as well as low-dimensional representations of the individual entities themselves. We demonstrate the effectiveness of the approach across several design style recommendation tasks. To the best of our knowledge, this work is the first to develop an ML-based model for the task of HTML-based email style recommendation.

KEYWORDS

Design style recommendation, graph neural networks, hypergraphs

1 INTRODUCTION

Designing HTML-based email marketing campaigns, posters, or other web-based marketing material is typically very time-consuming, requiring a lot of manual effort, and therefore very costly. Some email authoring tools in the market (e.g., Mailchimp [2], ActiveCampaign [1], Sendinblue [4], Moosend [3], etc.) attempt to help marketers create email campaigns more efficiently by providing drag-and-drop email editors. However, designers still need to manually select the design components (e.g., buttons, texts, images) and customize the design style of each design component. These tools also provide an easy start to HTML-based email creation with a variety of pre-designed email templates, but the choice can be limited and usually fail to align with brand-specific design styles.

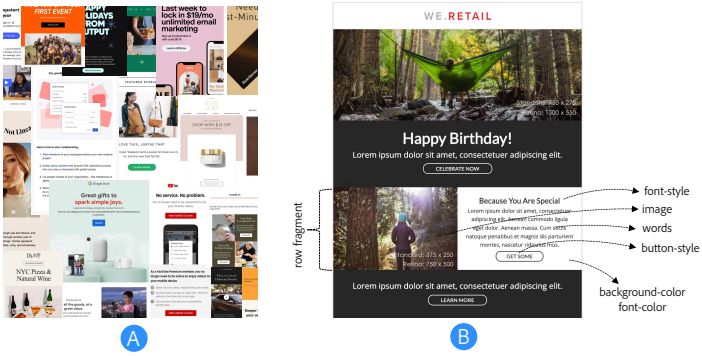


Figure 1: Our approach is trained on a large corpus of HTML documents, that is, marketing emails as shown in (A). Every HTML document is then decomposed into HTML fragments and a set of entities are extracted from each (B).

To address these problems, we develop an ML-based approach that learns to recommend high quality design styles by leveraging a large corpus of HTML-based marketing emails. For a few examples of the HTML documents used for training our model, see Figure 1A. More specifically, given a large corpus of HTML documents (emails, websites [8]), our approach decomposes each of these into a sequence of high-level HTML fragments; see Figure 1B for an example of a fragment. Next, we decompose each fragment into a set of style and design elements such as button, text, and background style, among many others as summarized in Table 1. We encode the fragments as hyperedges in a large heterogeneous hypergraph where each hyperedge is a set of entities of different types. Most importantly, there is often overlap between the different hyperedges, and based on this intuition, we develop a hypergraph neural network framework that learns a model from the hypergraph that captures the similarity between the different HTML fragments, and the higher-order relationships between the individual style and design elements of the fragments (hyperedges). Since the individual entities are often in more than a single HTML fragment (*e.g.*, font or button-style), we must learn a mechanism that enables us to "contextualize" them for any arbitrary HTML fragment, which can be one that the entity participates in already, that is, from the training corpus, or a new HTML fragment design that we may want to recommend to the user. For this, we learn both individual embeddings of the entities (nodes) along with an embedding for each HTML fragment (hyperedge). Most importantly, our learning-based approach can naturally be used to derive scores for

WWW '23 Companion, April 30-May 4, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9419-2/23/04…$15.00
https://doi.org/10.1145/3543873.3587300

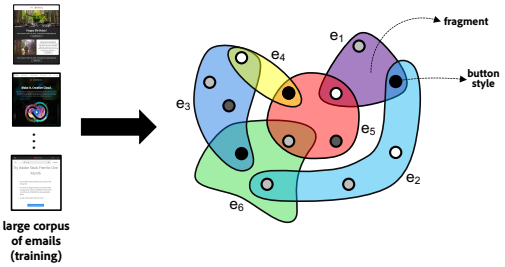


Figure 2: Given a corpus of HTML documents, we derive a large heterogeneous hypergraph that succinctly encodes the dependencies between the various sets of entities. Node color encodes entity type, e.g., ● represents a button-style.

entire HTML designs or fragments. The inferred scores naturally induce a ranking of the designs and we can then recommend the top-k most effective ones to the user. Our approach is able to make such complex recommendations since it encodes the entire fragment along with its style and the design elements as a hyperedge where each element is represented as a node and the set of them as a hyperedge. Thus, our approach can recommend entire fragments by simply solving a hyperedge ranking task. It is straightforward to see that such fragments do not need to be present in the training set, and thus our approach is naturally inductive as it can be used to infer the quality of any HTML fragment.

The key contributions of this work are as follows:

- **Problem Formulation:** We formulate the HTML document style recommendation task as a hypergraph learning problem and apply our model to perform style recommendation.
- **Approach:** We propose an ML-based approach for style recommendation. To the best of our knowledge, this work is the first to develop an ML-based model for email style recommendation.
- **Effectiveness:** The experiments demonstrate the utility of the proposed approach for design style recommendation tasks.
- **Benchmark Data:** We derived a heterogeneous hypergraph from a large corpus of HTML documents representing marketing emails and release the benchmark dataset for others to use.

2 APPROACH

2.1 HTML Email Document Corpus

We first collected a large-scale HTML document corpus from Really Good Emails (https://reallygoodemails.com/), and then extract the HTML fragments from each email document in the corpus (Figure 1B). Such fragments may consist of buttons, background-style, text, images, and so on. Words and images are extracted, but not used. There are 7 unique node types, 24,614 hyperedges (fragments) from 949 HTML documents (emails). See Table 1 for a summary of all such entities and their statistics. To enable others to investigate this important task and further advance the state-of-the-art, we make our curated benchmark dataset accessible at:

https://doi.org/10.6084/m9.figshare.21266253

This curated dataset can be used for benchmark comparisons, as well as for training new models for these new HTML style recommendation tasks, which have applications in designing effective websites, posters, and emails (for better marketing).

2.2 Hypergraph Extraction

Given a large corpus of HTML documents (*i.e.*, promotional marketing emails), we derive a heterogeneous hypergraph from the corpus by first decomposing each HTML email into a set of fragments as shown in Figure 1B. Now, for every fragment, we decompose it further into smaller fine-grained entities such as buttons and background style. These entities are included as nodes in the hypergraph and the set of all entities extracted from the fragment are encoded as a hyperedge as shown in Figure 2. Hence, every hyperedge in the heterogeneous hypergraph represents a fragment from some HTML document. To capture the spatial relationship present between fragments in an HTML document, we also include a node for each fragment along with an edge connecting each fragment to the fragment immediately below or beside it. Hence, this captures the sequence of such fragments in the larger email. Notice that hyperedges in this hypergraph are heterogeneous in that they consist of a set of heterogeneous nodes of various types as shown in Table 1. Most importantly, the entities of a fragment (hyperedge) are not unique to the specific fragment, and can be connected to other fragments (hyperedges) as shown in Figure 2. Each fragment representing a hyperedge in Figure 2 may contain multiple overlapping nodes. Notably, we see that two HTML fragments represented as hyperedges e_1 and e_2 in Figure 2 contain buttons with the same style. This overlap in button-style often implies other stylistic similarities between the two fragments.

2.3 Model

We now describe the learning-based model for HTML design style recommendation.

Preliminaries. Let $G = (V, E)$ denote a hypergraph where $V = \{v_1, \ldots, v_N\}$ are the $N = |V|$ vertices and $E = \{e_1, \ldots, e_M\} \subseteq 2^V$ is the set of $M = |E|$ hyperedges. Hence, a hyperedge $e \in E$ is a set of vertices $e = \{s_1, \ldots, s_k\}$ such that $\forall s_i \in e, s_i \in V$. Furthermore, hyperedges can be of any arbitrary size and are not restricted to a specific size, thus, $e_i, e_j \in E$, then $|e_i| < |e_j|$ may hold. Let $\mathbf{H}$ denote the $N \times M$ hyper-incidence matrix of the hypergraph G such that $H_{ik} = 1$ iff vertex $v_i \in V$ is in hyperedge $e_k \in E$ and $H_{ik} = 0$ otherwise. Intuitively, $\mathbf{H} \in \mathbb{R}^{N \times M}$ connects the nodes to their hyperedges and vice-versa. The hyperedge degree vector $\mathbf{d}^e \in \mathbb{R}^M$ is $\mathbf{d}^e = \mathbf{H}^\top \mathbf{1}_N$ where $\mathbf{1}_N$ is the N-dimensional vector of all ones. Then the degree of a hyperedge $e_j \in E$ is simply $d^e_j = \sum_i H_{ij}$. We define the *diagonal hyperedge node degree matrix* as $\mathbf{D} = \texttt{diag}(\mathbf{H1}_M)$ where $\mathbf{D}$ is a $N \times N$ diagonal matrix with the

Table 1: Statistics and properties of the HTML document corpus and the heterogeneous hypergraph derived from it. Note $|V|$ = # nodes of a given type, Δ denotes max hyperedge degree; d_{avg} and d_{med} are the mean/median degree.

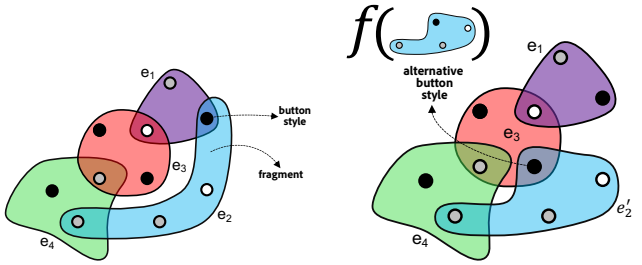
Node Type	$\lvert V\rvert$	Δ	d_{avg}	d_{med}
button-style	2361	21	7.14	6
text-style	14678	1548	173.42	6
background & font color	2798	72	10.46	4
background style	811	73	20.96	8
word	31761	6664	382.77	22
image	7682	200	7.29	1
fragment	24614	235	27.84	16

Table 2: Results for Button Style Recommendation.

	HR@K				nDCG@K			
Model	@1	@10	@25	@50	@1	@10	@25	@50
Random	0.000 ± 0.00	0.000 ± 0.00	0.018 ± 0.00	0.027 ± 0.00	0.000 ± 0.00	0.000 ± 0.00	0.004 ± 0.00	0.006 ± 0.00
Pop.	0.000 ± 0.00	0.000 ± 0.00	0.009 ± 0.00	0.009 ± 0.00	0.000 ± 0.00	0.002 ± 0.00	0.003 ± 0.00	0.005 ± 0.00
HyperGCN	0.008 ± 0.01	0.011 ± 0.00	0.043 ± 0.02	0.066 ± 0.03	0.008 ± 0.01	0.009 ± 0.00	0.017 ± 0.01	0.021 ± 0.01
Our Approach	0.243 ± 0.05	0.477 ± 0.03	0.536 ± 0.06	0.594 ± 0.05	0.243 ± 0.05	0.354 ± 0.04	0.368 ± 0.04	0.379 ± 0.04

Table 3: Results for Background Style Recommendation.

	HR@K				nDCG@K			
Model	@1	@10	@25	@50	@1	@10	@25	@50
Random	0.000 ± 0.00	0.000 ± 0.00	0.000 ± 0.00	0.074 ± 0.00	0.000 ± 0.00	0.000 ± 0.00	0.000 ± 0.00	0.013 ± 0.00
Pop.	0.000 ± 0.00	0.000 ± 0.00	0.000 ± 0.00	0.000 ± 0.00	0.001 ± 0.00	0.006 ± 0.00	0.010 ± 0.00	0.016 ± 0.00
HyperGCN	0.000 ± 0.00	0.031 ± 0.04	0.061 ± 0.06	0.147 ± 0.14	0.000 ± 0.00	0.018 ± 0.02	0.025 ± 0.03	0.041 ± 0.04
Our Approach	0.181 ± 0.11	0.457 ± 0.14	0.552 ± 0.10	0.741 ± 0.08	0.181 ± 0.11	0.308 ± 0.11	0.333 ± 0.11	0.369 ± 0.10



(a) (b)

Figure 3: Our approach naturally handles the inductive setting where we have new unseen fragments (set of styles and design elements). This is important for our recommendation tasks since we need to score alternative HTML fragment designs for recommendation. In (a) we have the original fragment whereas (b) is an alternative design using different button styles. See text for discussion.

hyperedge degree $d_i = \sum_j H_{ij}$ of each *vertex* v_i on the diagonal and $\mathbf{1}_M = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^\top$ is the all ones vector. Furthermore, we define the diagonal hyperedge degree matrix $\mathbf{D}_e = \texttt{diag}(\mathbf{H}^\top \mathbf{1}_N) = \texttt{diag}(d_1^e, d_2^e, \ldots, d_M^e)$ where $\mathbf{D}_e$ is a $M \times M$ matrix with hyperedge degree $d_j^e = \sum_i H_{ij}$ of each hyperedge $e_j \in E$ on the diagonal.

Hyperedge-Dependent Convolutions. Intuitively, we desire the individual entities to be embedded close together in the d-dimensional space if they are highly connected *and* occur in many of the same hyperedges, which are also close to one another. More formally,

$$\mathbf{Z}^{(k+1)} = \sigma\big((\mathbf{D}^{-1}\mathbf{H}\mathbf{P}_e\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}^{-1}\mathbf{Z}^{(k)} + \mathbf{D}^{-1}\mathbf{H}\mathbf{Y}^{(k)})\mathbf{W}^{(k)}\big) \quad (1)$$

$$\mathbf{Y}^{(k+1)} = \sigma\big((\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{P}\mathbf{D}^{-1}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{Y}^{(k)} + (\mathbf{H}\mathbf{D}_e^{-1})^\top\mathbf{Z}^{(k+1)})\mathbf{W}_e^{(k)}\big) \quad (2)$$

where Eq. 1-2 defines the hypergraph convolutional layers of our model, resulting in the updated node $\mathbf{Z}^{(k+1)}$ and hyperedge embeddings $\mathbf{Y}^{(k+1)}$ at layer $k+1$. In Eq. 2, the *random walk hyperedge transition matrix* $\mathbf{P}_e = (\mathbf{D}^{-1}\mathbf{H})^\top\mathbf{H}\mathbf{D}_e^{-1} \in \mathbb{R}^{M\times M}$ is used whereas Eq. 2 uses the *random walk node transition matrix* $\mathbf{P} = \mathbf{H}\mathbf{D}_e^{-1}(\mathbf{D}^{-1}\mathbf{H})^\top \in \mathbb{R}^{N\times N}$. Importantly, the node embeddings at each layer are updated using the hyperedge embedding matrix $\mathbf{D}^{-1}\mathbf{H}\mathbf{Y}^{(k)}$. Similarly, the hyperedge embeddings at each layer are updated using the $(\mathbf{H}\mathbf{D}_e^{-1})^\top\mathbf{Z}^{(k+1)}$ node embedding matrix. The process repeats until convergence. Note that the initial node $\mathbf{Z}^{(1)}$ and hyperedge features $\mathbf{Y}^{(1)}$ are derived as:

$$\mathbf{Z}^{(1)} = \phi(\mathbf{H}\mathbf{H}^\top - \mathbf{D}) \quad (3)$$

$$\mathbf{Y}^{(1)} = \phi(\mathbf{H}^\top\mathbf{H} - \mathbf{D}_e) \quad (4)$$

where ϕ is a function that maps $\mathbf{H}\mathbf{H}^\top - \mathbf{D}$ and $\mathbf{H}^\top\mathbf{H} - \mathbf{D}_e$ to a low-dimensional embedding. In this work, we used SVD as in [7].

2.4 Training

The training objective for our approach is formally presented in this section. Let $E = \{e_1, e_2, \ldots\}$ denote the set of known hyperedges in the hypergraph G where every hyperedge $e_t = \{s_1, \ldots, s_k\} \in E$ represents a set of nodes that can be of any arbitrary size $k = |e_t|$. This is important since HTML fragments that represent hyperedges in our heterogeneous hypergraph can be varying sizes, e.g., simple HTML fragments may only have a few entities whereas more complex HTML fragment designs may consist of a large number of entities. Hence, for any two hyperedges $e_t, e_t' \in E$, then $|e_t| \neq |e_t'|$ may hold. Further, let F be a set of sampled vertex sets from the set $2^V - E$ of unknown hyperedges. Given an arbitrary hyperedge $e \in E \cup F$, we define a hyperedge score function f as:

$$f: \quad e = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \rightarrow w \quad (5)$$

Hence, f maps the set of d-dimensional embedding vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ of the hyperedge e to a score $f(e = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\})$ or simply $f(e)$. In this work, we define $f(e)$ to be the mean cosine similarity between all pairs of nodes in the hyperedge $e \in E$. More formally,

$$f(e) = \frac{1}{T} \sum_{\substack{i,j \in e \\ \text{s.t. } i>j}} \mathbf{x}_i^\top \mathbf{x}_j \quad (6)$$

where $T = \frac{|e|(|e|-1)}{2}$ is the number of unique node pairs i, j in the hyperedge e. Intuitively, the hyperedge score $f(e)$ is largest when all nodes in the set $e = \{s_1, s_2, \ldots\}$ have similar embeddings. Then,

the hyperedge prediction loss function is:

$$\mathcal{L} = -\frac{1}{|E \cup F|} \sum_{e \in E \cup F} Y_e \log\big(\rho(f(e_t))\big) + (1 - Y_e) \log\big(1 - \rho(f(e_t))\big)$$

where $Y_e = 1$ if $e \in E$ and otherwise $Y_e = 0$ if $e \in F$. Further, let $\rho(f(e_t)) = \frac{1}{1+\exp[-f(e_t)]}$ where $p(e_t) = \rho(f(e_t))$ is the probability of hyperedge e_t existing in the hypergraph G. Furthermore, our approach naturally extends to the inductive learning setting where we can use it to recommend new unseen HTML fragments representing sets of entities (Figure 3). In particular, given a new unseen fragment, we can leverage the hyperedge score function as shown in Figure 3 to derive a score for this new unseen fragment, which indicates how well these entities go together to form a well-designed HTML fragment. Intuitively, if the embeddings from each entity in the fragment are similar, then the function f will score this fragment higher than a fragment with dissimilar entities where the notion of similarity is learned from the large corpus of professionally designed HTML emails/documents.

3 EXPERIMENTS

In this section, we use the proposed approach for style recommendation tasks. To quantitatively evaluate the effectiveness of our approach for style recommendation tasks, we hold out 20% of links in the hypergraph that occur between a fragment and a style entity (e.g., button-style) to use as ground-truth for quantitative evaluation. Then HNN is trained using the other 80% links. Given the learned embeddings from our approach, we derive a score between HTML fragment i and every button-style $k \in V_B$ as follows:

$$\mathbf{w}_i = f(\mathbf{z}_i, \mathbf{z}_k), \quad \forall k \in V_B \tag{7}$$

where f is a score function (*i.e.*, cosine) and $\mathbf{w}_i = \begin{bmatrix} \mathbf{w}_{i1} & \cdots & \mathbf{w}_{i|V_B|} \end{bmatrix}$ are the scores. We then sort the scores $\mathbf{w}_i$ and recommend top-K styles with largest weight. To quantitatively evaluate the recommendations, we use HR@K and nDCG@K where $K = \{1, 10, 25, 50\}$. We repeat the above for each of the held-out links in the test set and report the mean of the evaluation metrics.

Since this is the first work to study this problem, there are no immediate baselines for comparison. Nevertheless, we compare our approach to several common-sense baseline methods including random that recommends a style or design element uniformly at random among the set of possibilities, popularity (pop) that recommends the most frequent style, and HyperGCN [9]. The results showing the effectiveness of the various approaches for recommending the top button-styles are provided in Table 2. Notably, our approach performs significantly better than the other models across both HR@K and nDCG@K for all $K \in \{1, 10, 25, 50\}$. In many instances, the simple random and popularity baseline are completely ineffective with HR@K and nDCG@K of 0 when K is small (top-1 or 10). In contrast, our approach correctly recovers the ground-truth button-style 24% of the time in the top-1 (Table 2). Results for recommending useful background-styles are reported in Table 3. Our approach performs best, achieving better HR and nDCG across all K. It is important to note that results at smaller K are more important, and these are precisely the situations where the other models completely fail, that is, for top-1 all approaches have a HR of 0 indicating they are never able to correctly recover the ground-truth background-style that was held-out.

Table 4: Hyperedge Prediction Results.

	AUC	MAP
GCN	0.707 ± 0.03	0.622 ± 0.03
GraphSAGE	0.769 ± 0.02	0.788 ± 0.03
HyperGCN	0.467 ± 0.03	0.478 ± 0.04
HGNN	0.802 ± 0.03	0.731 ± 0.03
Our Approach	**0.846** ± 0.02	**0.810** ± 0.03

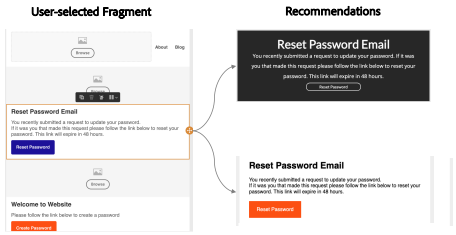


Figure 4: HTML Fragment Recommendations

To further evaluate the effectiveness of our approach, we also leverage it for prediction of entire HTML fragments of an email in Table 4. Recall an HTML fragment consists of a set of entities such as buttons, text, background-style, among others. Therefore, this task is equivalent to predicting hyperedges of arbitrary size. For training, we select 80% of the observed hyperedges (HTML fragments) and use the remaining 20% for testing. The same amount of negative hyperedges $f \in F$ are generated by sampling a hyperedge $e \in E$ and corrupting it by replacing $\frac{|e|}{2}$ nodes of e with $\frac{|e|}{2}$ other nodes sampled uniformly from $V - e$. For this HTML fragment prediction task, we evaluate our approach against HyperGCN [9], HGNN [5], GraphSAGE [10], and GCN [6]. Results are reported in Table 4. Overall, our approach achieves the best predictive performance over all other models. This demonstrates the effectiveness of our approach for HTML fragment prediction that gives rise to many important practical applications such as for facilitating the design of web sites and marketing emails. Finally, we also provide a few example recommendations in Figure 4.

REFERENCES

[1] 2022. ActiveCampaign - Email Marketing, Automation, and CRM. https://www.activecampaign.com
[2] 2022. Mailchimp: Marketing smarts for big ideas. https://mailchimp.com/
[3] 2022. Moosend - Email Marketing Automation Platform for Thriving Businesses. https://moosend.com/
[4] 2022. Sendinblue - All Your Digital Marketing Tools in One Place. https://www.sendinblue.com
[5] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *AAAI*, Vol. 33. 3558–3565.
[6] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907* (2016).
[7] Fayokemi Ojo, Ryan A Rossi, Jane Hoffswell, Shunan Guo, Fan Du, Sungchul Kim, Chang Xiao, and Eunyee Koh. 2022. VisGNN: Personalized Visualization Recommendation via Graph Neural Networks. In *WWW*. 2810–2818.
[8] Qifan Wang, Yi Fang, Anirudh Ravula, Fuli Feng, Xiaojun Quan, and Dongfang Liu. 2022. WebFormer: The Web-Page Transformer for Structure Information Extraction. In *WWW*. 3124–3133.
[9] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In *NeurIPS*. 1509–1520.
[10] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*. 974–983.