A Graph Representation of Semi-structured Data for Web Question Answering

Xingyao Zhang[1†] **Linjun Shou**[2†] **Jian Pei**[3‡] **Ming Gong**[2] **Lijie Wen**[1] **Daxin Jiang**[2§]
[1]Tsinghua University
[2]STCA NLP Group, Microsoft
[3]School of Computing Science, Simon Fraser University
xingyaozhangthu@gmail.com, {lisho,migon,djiang}@microsoft.com,
jpei@cs.sfu.ca, wenlj@tsinghua.edu.cn

Abstract

The abundant semi-structured data on the Web, such as HTML-based tables and lists, provide commercial search engines a rich information source for question answering (QA). Different from plain text passages in Web documents, Web tables and lists have inherent structures, which carry semantic correlations among various elements in tables and lists. Many existing studies treat tables and lists as flat documents with pieces of text and do not make good use of semantic information hidden in structures. In this paper, we propose a novel graph representation of Web tables and lists based on a systematic categorization of the components in semi-structured data as well as their relations. We also develop pre-training and reasoning techniques on the graph model for the QA task. Extensive experiments on several real datasets collected from a commercial engine verify the effectiveness of our approach. Our method improves F1 score by 3.90 points over the state-of-the-art baselines.

1 Introduction

Question answering (QA) has become an important feature in most search engines as it delivers information to users in an effective and easy-to-understand manner. Answers to questions are often extracted from Web tables and lists. For example, Figure 1 shows the search result page (SERP) of questions Q_1 "*cities with the highest GDP in the world*" and Q_2 "*the best skydiving locations in the world*", where the answer to Q_1 is from a Web table, while that to Q_2 is from a Web list.

Comparing to unstructured plain text, semi-structured Web data, such as Web tables and lists, are more effective to represent rich relational information. Relations among various elements in a Web table or a list may be useful in answering user questions. According to the statistics from a global commercial search engine, there are hundreds of millions of semi-structured data pieces, including tables and lists on the Web, and the intents of 30% of user queries can be answered by semi-structured data.

Previous attempts towards question answering (QA) using semi-structured data on the Web are mainly IR-based approaches (Balakrishnan et al., 2015; Chakrabarti et al., 2020). Typically, those methods convert semi-structured data into documents by sequentially rearranging text cells to adapt to language models (Chakrabarti et al., 2020; Wang et al., 2018; Zhang and Balog, 2018). Those studies do not make use of inherent structural relationships among components of Web tables or lists. For example, the rearrangement does not consider the vertical relations among cells locating in the same columns, such as the relation among "New York", "Tokyo" and "Los Angeles" in Figure 1(a).

Some recent studies leverage tabular structure implicitly. For example, Nishida et al. (2017) cast tables as matrices of text and apply convolutional neural networks for table embedding. Zhang and Balog (2018) cut tables into smaller fragments. However, the structural information in Web tables and lists is more complex than the simple adjacency relation of matrices. How to take the best advantage of both the text

[†]Equal contribution. Work done when the first author was an intern at Microsoft STCA.

[‡]Jian Pei's research is supported in part by the NSERC Discovery Grant program. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

[§]Corresponding author.

Proceedings of the 28th International Conference on Computational Linguistics, pages 51–61
Barcelona, Spain (Online), December 8-13, 2020

information and the structural relations in Web tables and lists in QA remains a challenge not thoroughly explored.

In this paper, we tackle the problem of Web QA over semi-structured data, and make the following contributions. First, in Section 2, we systematically categorize different components in semi-structured Web data, including captions, headers, subject columns, attribute columns, and cells, as well as their relations, including cell-cell relation, header-cell relation, subject-attribute relation, and caption-content relation. We propose `GraSSLM`, a graph model to jointly represent both text and structural information in semi-structured data for Web QA in Section 3. Our `GraSSLM` model explicitly represents different types of components as nodes in a graph and their relations as edges. Our model integrates heterogeneous information effectively, including text and structures, and reveals hidden semantic correlations across various components naturally.

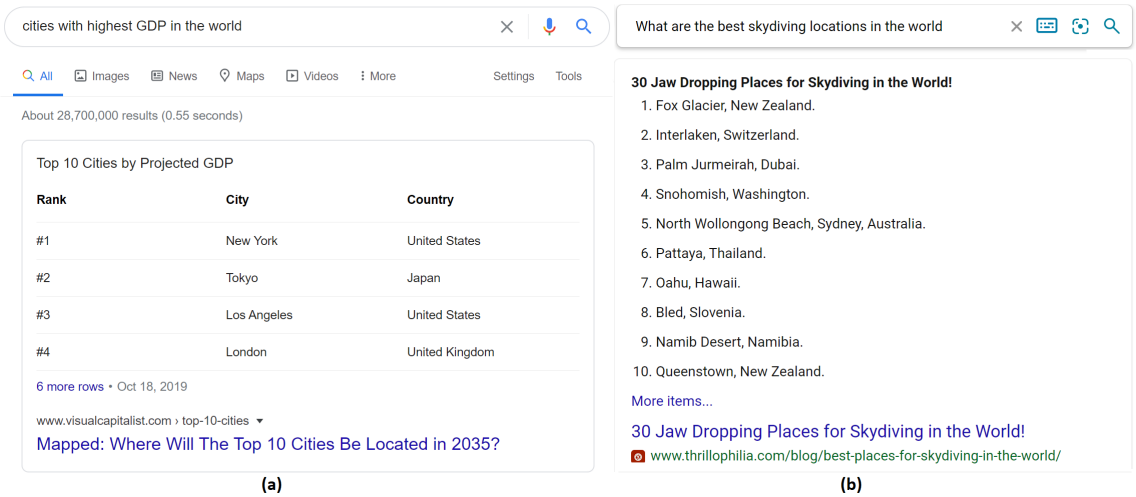

Figure 1: Examples of Web table (a) and Web list (b) from a commercial search engine. Two queries are non-factoid queries, which can be answered by the information from semi-structured data like tables and lists.

Second, in Section 3, we apply two pre-training techniques for graph models. In particular, we design a novel node prediction objective (NPO) to leverage graph structure in node embedding. This pre-training task requires a model to predict the entire content for a masked node from the unmasked neighbor nodes, which guides the model to learn attention over the context. The attention is used in the graph reasoning stage, where the representation of each node is updated by the aggregated information from the neighbor nodes. In this way, the inherent semantic correlations among neighbor nodes are propagated via the structural connections in the graph. Comparing to the previous methods, this graph pre-training and reasoning mechanism better exploits structural information in tables and lists.

Last, to compare our model with the state-of-the-art methods, we create new datasets which contain real-world queries collected from a global commercial search engine paired with table and list data mined from the Web. Each pair $\langle$query, table/list$\rangle$ is further labeled by crowdsourcing annotators with consensus on relevance. The experimental results on the test sets, reported in Section 4, show that `GraSSLM` outperforms the best state-of-the-art baselines up to 1.77 in F1 score on average.

2 Problem Statement

Let S be a semi-structured data example, either a Web table T or a Web list L. There are different types of tables and lists, for example, relational tables, entity tables, matrix tables, enumerate lists and group lists (Lautert et al., 2013). Our method is generally applicable to all those types, therefore, we do not distinguish them in this paper.

Figure 2: Components of Web Semi-Structured Data.

2.1 Components of Web Semi-structured Data

Following Crestan and Pantel (2011) and Eberius et al. (2015), we divide a Web semi-structured data example into various components, as illustrated in Figure 2.

A **caption** C is a direct description that is usually adjunct to the content body of the semi-structured data. For example, in Figure 1 "Top 10 Cities by Projected GDP" and "30 Jaw Dropping Places for Skydiving in the world!" are captions for the table and the list, respectively.

Data content refers to the body of semi-structured data, which consists of multiple rows and columns. A special row, the **header**, often locates at the top of the table. The elements in a header often describe the classes that the content of the table belongs to. For example, in Figure 2, the first row consisting of "Rank", "City", and "Country" is the header of the table. The elements in the remaining rows of the table are **cells**. Vertically, cells are grouped into columns, where we identify subject columns and attribute columns. **Subject columns** refer to one or more key subjects or entities of the table, while **attribute columns** list the attribute information of the corresponding subjects or entities. In Figure 2, the column of "City" is a subject column, while the columns of 'Rank" and "Country" are both attribute columns. To recognize subject columns, we adopt a heuristic method (Nishida et al., 2017), which calculates the distinct string ratio as seeds for subject classification. Our empirical study finds that this simple method achieves an accuracy over 95%. Besides, a schema classification method (Eberius et al., 2015) is applied to detect and transpose vertical Web tables into horizontal ones.

A list can be regarded as a special type of table, which has only one single column, and has no header.

2.2 Relations among Components in Tables and Lists

Different components in tables and lists bear inherent semantic relations. Modeling those relations in a graph model fusions semantics among components and achieves a rich representation of semi-structured data. Particularly, we are interested in the following four types of relations.

Caption-Content Relation. A caption is often a summary of the context and content in a table or a list. The words in a caption are often reliable evidences to determine the relevance between a query and a semi-structured data example.

Header-Cell Relation. Since a header often outlines the classes that the cells belong to, a header-cell relation is usually a class-instance relation. For example, the cell, "Los Angeles" in Figure 1 is an instance of the class "City".

Subject-Attribute Relation. More often than not tables store entity information. In such a table, each row, except for the header, corresponds to one entity, where the cells in the subject columns contain the entity names, and the remaining cells in the attribute columns consist of the attributes for that entities. For example, in Figure 1(a), the third row corresponds to a "City" entity "Los Angeles", and "#3" and "United States" are the values of attributes "Rank" and "Country" of the entity, respectively. The subject-attribute relation is usually an entity-attribute relation.

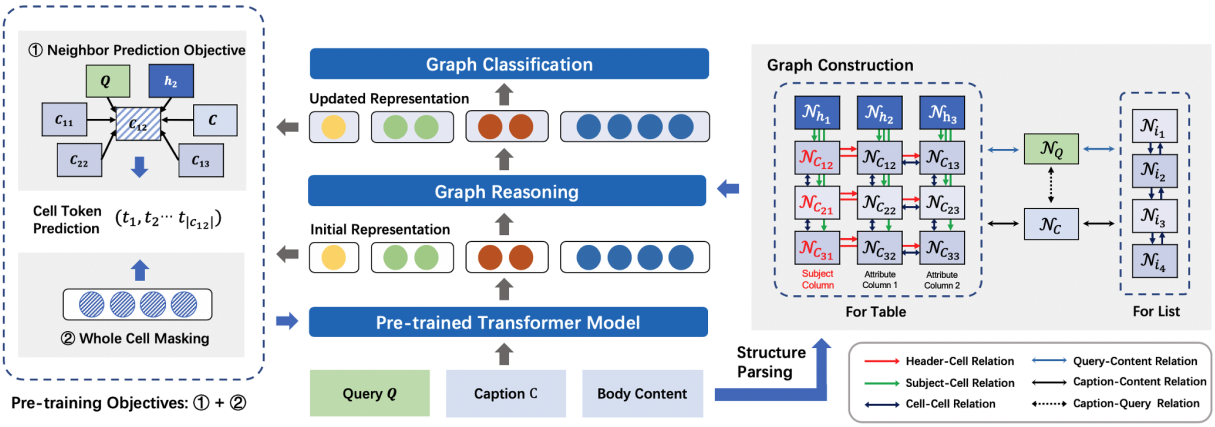

Figure 3: GraSSLM contextually encodes query and semi-structured data via a pre-trained transformer model. It then builds a graph for the semi-structured data into a graph, and updates the node embeddings by the graph reasoning module. Finally the graph classification module aggregates the nodes information to predict the QA match score. In addition, GraSSLM applies two graph pre-training objectives to encourage the model to learn attention over contextual nodes.

Cell-Cell Relation. If we ignore the subject columns, the remaining cells within the same rows or columns are also semantically related. The cells in the same row often describe the various attributes of the same entity, while the cells in the same column are often instances of the same class.

As mentioned in Section 2.1, lists can be considered as a special type of tables. They only have Caption-Content relation and Cell-Cell relation.

The problem of QA over semi-structured data is that, given a query Q and a semi-structured data example S, return the **QA match score** $d(Q, S)$, which predicts the likelihood that S answers Q.

3 Method

In this section, we proposed GraSSLM, which is a graph model of semi-structured data on the Web for QA. Figure 3 shows the overall structure of GraSSLM.

GraSSLM is composed of three components. First, a *pre-trained language model* generates token-level contextual embedding for the concatenation of an input query Q and a semi-structured data example S. Second, a *graph construction module* converts the initial plain text embedding into graphs. Last, a *graph reasoning and classification module* predicts matching results.

3.1 Graph Construction

Given a query Q and a semi-structured data example S of M rows and N columns, we construct a graph based on the components and their relations described in Section 2. Figure 3 illustrates the graph constitution: (i) *Query Node* $\mathcal{N}_Q$; (ii) *Caption Node* $\mathcal{N}_C$; (iii) *Header Nodes* $\{\mathcal{N}_{h_j}\}$, where $1 \leq j \leq N$; (iv) *Cell Nodes* $\{\mathcal{N}_{c_{ij}}\}$, where $2 \leq i \leq M, 1 \leq j \leq N$.

The edges in the graph are created as follows. The first group of edges are formed based on the structural relations in the semi-structured data example. These structural relations carry the inherent semantic relations between the components, and define the context to better represent the elements in a semi-structured data example through the following four types of edges. (i) Caption-Content Relation: edges between caption nodes and cell nodes. (ii) Header-Cell Relation: edges between header nodes and the cell nodes in the corresponding columns. (iii) Subject-Attribute Relation: edges between subject cell nodes and the attribute cell nodes in the corresponding rows. (iv) Cell-Cell Relation: edges between neighbour cell nodes in the same row or in the same column.

The second group of edges connect the query and the semi-structured data example by connecting $\mathcal{N}_Q$ with all nodes in S. The weights of these edges will be derived in the graph reasoning stage to represent

the bi-directional attention between the query words and data components, including the edges between query node and cell node, as well as the edges between query node and caption node.

The graph for a list is a simplified version of that for a table, where there are no Header-Cell edges or Subject-Cell edges.

3.2 Graph Initialization

To obtain the initial representation of graph nodes, we first concatenate the query Q and the text in the semi-structured data example S. The concatenated string consists of $G = (Q, C, \{h_j\}, \{c_{ij}\})$, where C is the caption, $\{h_j\}$ are the tokens in the header, and $\{c_{ij}\}$ are the cells in S. We feed this concatenated string G into a pre-trained BERT model (Vaswani et al., 2017) and derive a contextual embedding for each token in G. We use $\mathcal{LM}(G)$ to denote this representation. In this paper, $\text{BERT}_{\text{base}}$ (Devlin et al., 2018) is used for contextual embedding.

We further derive the initial representation for each node in the graph. Since different nodes may contain various lengths of token spans, we adopt the method in (Fang et al., 2019), which applies a BiLSTM (Chen et al., 2017) on top of the transformer output and a multi-layer perceptron MLP to convert various lengths of token spans into a fix-sized vector as the node representation. We write the BiLSTM model as a function $\mathcal{B}$. We denote by $\mathcal{B}(\mathcal{LM}(G))$ the model on top of the transformer output, and by $\mathcal{B}(\mathcal{LM}(G))[s;t]$ the sequence of hidden states in the model for span extremes in position s and t. We use the subscripts $start$ and end to denote the start and end positions of the tokens of the corresponding components. The initial representation for the nodes are as follows, where normal fonts are used for the text of the corresponding nodes, and bold fonts for the embedding.

$$\boldsymbol{Q} = MLP\Big(\mathcal{B}(\mathcal{LM}(\boldsymbol{G}))[Q_{start}; Q_{end}]\Big) \qquad \boldsymbol{C} = MLP\Big(\mathcal{B}(\mathcal{LM}(\boldsymbol{G}))[C_{start}; C_{end}]\Big)$$

$$\boldsymbol{h_j} = MLP\Big(\mathcal{B}(\mathcal{LM}(\boldsymbol{G}))[h_{j;start}; h_{j;end}]\Big) \qquad \boldsymbol{c_{i,j}} = MLP\Big(\mathcal{B}(\mathcal{LM}(\boldsymbol{G}))[c_{i,j;start}; c_{i,j;end}]\Big)$$

3.3 Graph Reasoning and Prediction

After generating the dense representation of graph nodes, `GraSSLM` leverages a two-layer graph convolutional network (GCN) (Kipf and Welling, 2016) to perform message passing over the graph. At each layer, the graph convolutional neural network aggregates the neighbors' representations of one node and further transforms the aggregated representation with a linear projection. Let $\boldsymbol{L}^{(0)} = \{\boldsymbol{Q}, \boldsymbol{C}, \{\boldsymbol{h_j}\}, \{\boldsymbol{c_{ij}}\}\} \in \mathbb{R}^{K \times d}$, where $K = 2 + M \times N$ is the total number of nodes in the graph including the query node, caption nodes, header nodes and cell nodes, and d is the output dimensionality of the MLP in Section 3.2. The graph reasoning process is formalized as

$$\boldsymbol{L}^{(l)} = \sigma\Big(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{L}^{(l-1)}\boldsymbol{W}^{(l-1)}\Big),$$

where $\boldsymbol{L}^{(l)}$ denotes the l-th ($l = 1, 2$) layer of GCN, σ is the non-linear activation function, which is ReLU in our case and $\boldsymbol{W}^{(l-1)}$ is the weight matrix of the $(l-1)$-th layer. $\boldsymbol{D} \in \mathbb{R}^{K \times K}$ denotes the graph degree matrix, which records the amount of edges for every node and $\boldsymbol{A} \in \mathbb{R}^{K \times K}$ denotes the graph adjacency matrix, which records the graph edge information. Symbol $\sim$ here indicates a renormalization trick of adding a self-connection to each node of the graph and building the corresponding degree and adjacency matrix. After two rounds of convolution, $\boldsymbol{L}^{(2)}$ denotes the node features updated.

Graph prediction is derived by a mean pooling operation on the nodes of the graph, followed by an MLP, that is, $y = MLP(Pooling(\boldsymbol{L}^{(2)}))$, where y is the predicted QA match score for the corresponding input query Q and data example S.

3.4 Pre-training Strategy

Pre-training (Erhan et al., 2010) has become a new paradigm of natural language processing, and various pre-training techniques have been proposed (Devlin et al., 2018; Joshi et al., 2020). However, most previous pre-training techniques were designed for plain text. Due to the structural characteristics of Web semi-structured data, they cannot be applied directly to such data. In this paper, we propose a novel

pre-training method that allows the model to learn representations from semantics embedded in both text and structures of tables and lists. Following the successful pre-training experience of transformer-based models (Devlin et al., 2018), we used two pre-training objectives designed specifically for semi-structured data.

Whole Cell Masking (WCM). We follow the masked language model proposed by BERT (Devlin et al., 2018), but with different masking schema. Extended from whole word masking (Joshi et al., 2020), *Whole Cell Masking* firstly masks every token of the word if any of its pieces is masked. Additionally, it masks the whole cell content if any token in table cells or headers is masked. We mask 15% of all cells in total through replacing 80% of the masked cell tokens by a special mask token [MASK], 10% by random tokens and 10% with the original tokens. Given input $G = (Q, C, \{h_j\}, \{c_{ij}\})$, let $T = (t_1, \ldots, t_{|X|})$ be the sequence of tokens for G, where $t_m \in T$ is the m-th token, which is masked, that is, $\boldsymbol{t}_m = MLP(\boldsymbol{e}_m)$, where $\boldsymbol{e}_m \in \mathbb{R}^d$ denotes the token-level embedding of input t_m, which is generated by the contextual language model. After an MLP with one hidden layer, $\boldsymbol{e}_m$ is decoded as a token prediction score $\boldsymbol{t}_m \in \mathbb{R}^V$, where V denotes the vocabulary size.

Neighbor Prediction Objective (NPO). To incorporate the structural information of semi-structured data in the pre-training stage, we propose a novel *neighbor prediction objective* task for graph pre-training. The task is to predict each token inside a masked node using the representations of the neighbor nodes. In order to make the pre-training consistent with fine-tuning stage, we apply the same contextual embedding module and graph reasoning module as used in the fine-tuning process to generate the reasoned node representation.

Formally, denote by $\boldsymbol{L}_n^{(2)}$ the node representation of the n-th node $\mathcal{N}_n$ after contextual embedding and graph reasoning, and by function $neighbour(\cdot)$ the node representations of its neighbor nodes in the constructed graph. We use fixed sinusoidal embedding (Liu et al., 2019) as positional embedding to predict the tokens from $\boldsymbol{L}_n^{(2)}$.

$$\boldsymbol{r}_{n_k} = [Pooling(neighbour(\boldsymbol{L}_n^{(2)})); \boldsymbol{p}_k] \tag{1}$$

$$\boldsymbol{t}_{n_k} = MLP(\boldsymbol{r}_{n_k}) \tag{2}$$

where function $Pooling(\cdot)$ converts the neighbor node representations of $\boldsymbol{L}_n^{(2)}$ into a d-dimensional vector with mean pooling. We concatenate the representations of the neighbors and the k-th positional embedding $\boldsymbol{p}_k \in \mathbb{R}^d$ to get the representation for the k-th token $\boldsymbol{r}_{n_k}$. After an MLP with two hidden layers for decoding, we obtain the prediction result of the k-th token $\boldsymbol{t}_{n_k} \in \mathbb{R}^V$.

`GraSSLM` sums up the loss from both the whole cell masking objective and the neighbor prediction objective as the total loss function. For the m-th token in the input token sequence, we can find the corresponding position k in the n-th node. The total pre-training loss is

$$\mathcal{L}(t_m) = \mathcal{L}_{WCM}(t_m) + \mathcal{L}_{NPO}(t_m) = -logP(t_m|\boldsymbol{t}_m) - logP(t_m|\boldsymbol{t}_{n_k}) \tag{3}$$

Notably, NPO directly uses the masked input from WCM for graph construction and prediction.

4 Experiments

We evaluate the `GraSSLM` model and other baselines on three datasets, including one table QA dataset, one list QA dataset and one small dataset of complex query-table pairs. In addition, we leverage two other large-scale datasets for pre-training. We describe the three datasets as follows.

- **Table Query Matching dataset (Table-QM)** is an English table QA task dataset from one commercial Q&A system, which has about 34k labeled cases. Each case consists of three parts, a question, a table, and a binary label (0 or 1) by crowdsourcing judges indicating whether the question can be answered by the table. The dataset is collected as follows. First, for each question, the top 10 relevant documents returned by the search engine are selected to form pairs ⟨Question, Url⟩. Then, tables are extracted from those documents to form triples ⟨Question, Url, Table⟩. Those ⟨Query, Table⟩ pairs are sampled and sent to crowdsourcing judges. Specifically, each pair ⟨Query, Table⟩ is

Statistic	Table-QM	List-QM	DTable-QM	Table-Pretrain	List-Pretrain
Dataset size	34,276	62,581	2,457	6,994,996	100,954,174
Avg. query length	28.0	30.5	28.3	27.9	30.5
Avg. title length	134.1	34.9	124.3	103.4	30.5
Avg. # of rows	7.6	5.8	10.3	4.7	11.4
Avg. # of columns	2.8	1	5.6	2.4	1
Avg. # of cells	21.3	5.8	57.7	11.3	11.3
Positive/Negative	1:2.1	1:1.3	1:2.1	-	-

Table 1: Dataset Statistics.

required to be labeled by three judges. Those cases with 2/3 or higher positive labels receive positive final labels, otherwise negative.

- **List Query Matching dataset (List-QM)** is an English list QA task dataset from one commercial Q&A system, which has about 62k labeled cases. The data collection process is similar to Table-QM. For query selection, we include unordered lists, ordered lists and description lists (Consortium and others, 1999) to increase diversity.

- **Deep Tables Query Matching dataset (DTable-QM)** is a subset of the Table-DM dataset, including those tables with relatively complicated structures and larger amounts of information. Specifically, we selected the tables with more than 50 cells and 10 columns, which may bring challenges for semantic table modeling. This results in 2,457 instances in the dataset.

For pre-training of `GraSSLM`, we leverage the following semi-structured datasets.

- **Large-scale Table Pre-training dataset (Table-Pretrain)** is a large-scale unlabelled Web table dataset, which is extracted from 10 million Web pages sampled from the index of one commercial search engine. The sampling is conducted from the set of pages seen by United State users in the period from Feb 2019 to Dec 2019. After filtering out incomplete or empty Web tables, we extract 7M Web tables.

- **Large-scale List Pre-training dataset (List-Pretrain)** is a large-scale unlabelled Web list dataset. Similar to the process of deriving Table-Pretrain, 100M Web lists are extracted.

We compare `GraSSLM` with several strong baselines. **Single-field document retrieval (SDR)** (Cafarella et al., 2009; Cafarella et al., 2008) and **Multi-field document retrieval (MDR)** (Pimplikar and Sarawagi, 2012) are two representative methods that treat a semi-structured data example as a single document or multi-fielded document. They apply an IR approach for QA (Zhang and Balog, 2018). **Semantic table retrieval (STR)** (Zhang and Balog, 2018) introduces a semantic representation for Web tables. The representation includes sets of extracted concepts and entities. **BERT** (Devlin et al., 2018) is a powerful Transformer-based model, which has demonstrated impressive performance in the semantic matching task. We apply this model to a concatenation of the query and the sequential tokens in a semi-structured data example, and then use a multi-layer perception for classification. All the previous methods do not consider the structure information in tables or lists. In this paper, we applied Bert-base as our backbone model and baseline. The last baseline is **TAPAS** (Herzig et al., 2020), a recent state-of-the-art approach of QA over tables. This method encodes rows and columns to embed structural information of tables.

To measure the accuracy of matching, we use the average F1 as our metric. Precision, Recall, and F1 score are computed on the number of true positives (TP), false positives (FP), and false negatives (FN). F1 score is the harmonic mean of precision and recall. Since the matching prediction task is casted as a binary classification task, we consider F1 score as the metric and calculate average F1 based on that.

All methods are implemented in PyTorch (Paszke et al., 2017) and trained on an Ubuntu 16.04 with 64GB memory and eight GTX 1080 Ti GPU. For all data-sets, we randomly select 80% of the records as training set, 10% as validation set and the remaining 10% as test set. We train the model using training

data, and fix model parameters based on the best model performance on validation set. We then test the model on test set. We perform three random runs and report both mean and standard deviation for testing performance.

We use stochastic gradient descent (SGD) with a learning rate of 2e-5. We use mini-batches of size 64, with batch size 8 for each of 8 GPUs, we use with 1 hidden-layer of 768 hidden units. We use dropout with a rate of 0.5, which is applied to all feedforward neural networks. For the pre-trainng process, We use a batch size of 64 and fine-tune for 4 epochs over the large-scale data-set for two unsupervised task. For each task, we selected the fine-tuning learning rate of 2e-5. For the graph convolutional network, we applied a Bi-LSTM with hidden-layer with 768 hidden units on the top of transformer output. The GCN contains two convolutional layers with the hidden size of 1,536. After node-level convolution, we adapted mean-pooling for graph representation. As to the positional embedding, we created a fixed sinusoidal embedding with 768 hidden units.

For all baseline models, we use pre-trained corresponding transformer models as word embedding and using the output of token [CLS] as sentence embedding. Out-of-vocabulary (OOV) words are hashed to one of 100 random embedding each initialized to mean 0 and standard deviation 1. All other hidden layer weights were initialized from random Gaussian distribution with mean 0 and standard deviation 0.01. Each hyperparameter setting was run on a same machine as the `GraSSLM`, using Adagrad for optimization with initial accumulator value of 0.1.

4.1 Overall Performance

We compare `GraSSLM` against the state-of-the-art baselines on the Table-QM, List-QM and DTable-QM datasets. The results are reported in Table 2. As `GraSSLM` is complementary to language models, we use `GraSSLM` (BERT) to denote the language models used by `GraSSLM` as the backbone model.

Method	Table-QM	List-QM	DTable-QM	Average
SDR (Cafarella et al., 2009)	64.39	62.94	57.95	61.76
MDR (Pimplikar and Sarawagi, 2012)	65.82	62.79	59.21	62.61
STR (Zhang and Balog, 2018)	72.95	69.93	65.30	69.39
BERT (Devlin et al., 2018)	76.22	72.15	68.54	72.30
TAPAS (Herzig et al., 2020)	77.92	73.69	70.49	74.03
`GraSSLM`	**79.04**	**77.61**	**71.19**	**75.95**

Table 2: Performance comparison on the three datasets. The % signs are omitted. The best results are highlighted in bold.

`GraSSLM` consistently achieves the best performance against all baselines. `GraSSLM` outperforms the baselines BERT by up to 5.44% (List-QM). Our model captures both the text-level and structure-level information via explicitly modeling the inherit building components and their semantic correlation from Web semi-structured data. Comparing to the best IR-based method STR, our model is up to 7.68% better on the List-QM dataset. It demonstrates that the heterogeneous graph model in `GraSSLM` uses structural features more effectively than those IR-based methods, which focus on slicing Web semi-structured data into different documents but ignore the potential correlations among them. Besides, `GraSSLM` outperforms TAPAS, the newest baseline for QA on tables, by up to 3.90% in the List-QM dataset. It illustrates that the graph-based pre-training objectives strengthen the representation capability of models for semi-structured data, which will be further discussed in Section 4.3.

Notably, all the baselines display servere performance drops in the DTable-QM dataset, while `GraSSLM` still holds the best performance (71.19%). The explicit graph modeling guides the model to learn attention over noisy contexts, which benefit semantic reasoning on complicated tables.

4.2 Ablation Studies

We conduct ablation studies on GraSSLM to empirically examine the contribution of every components, particularly, the semantic relations we proposed, which includes the following steps.

Semantic Relation Ablation To further study the contribution of the semantic relations defined in Section 2.2, we removed the edges representing *Caption-Content Relation*, *Header-Cell Relation*, *Subject-Attribute Relation* and *Cell-Cell Relation* from graphs respectively and keep the other components untouched.

LSTM Ablation We replace Bi-LSTM, which generates node representation from the output of language model, by average pooling to obtain the fix-sized initial embedding as the inputs of graph neural network.

GCN Ablation We remove GCN, which aggregates and updates node-level representation and outputs the final prediction. We also remove the LSTM part as there is no need to generate node inputs. Instead, we use a MLP for classification, which makes the model same as one of our baselines BERT.

Method	**Table-QM**	**List-QM**	**DTable-QM**	Average
GraSSLM	79.04	77.61	71.19	
w/o Caption-Content Rel.	78.55 (0.49 ↓)	76.04 (1.57 ↓)	70.90 (0.29 ↓)	0.78↓
w/o Header-Cell Rel.	77.37 (1.67 ↓)	-	70.21 (0.98 ↓)	1.33↓
w/o Subject-Attribute Rel.	78.52 (0.52 ↓)	-	70.58 (0.61 ↓)	0.57↓
w/o Cell-Cell Rel.	77.92 (1.12 ↓)	74.60 (3.01 ↓)	70.10 (1.09 ↓)	1.74↓
w/o LSTM	78.34 (0.79 ↓)	76.16 (1.55 ↓)	69.90 (1.29 ↓)	1.21↓
w/o GCN	**76.30 (2.74 ↓)**	**72.49 (5.12 ↓)**	**68.97 (2.22 ↓)**	**3.36↓**

Table 3: Ablation studies on the main components, where the last column shows the average of performance reduction. The numbers are in percentage and the signs % are omitted.

The results show that ablation causes performance degrade to different extents. We can observe that removing GCN, which conducts explicit graph reasoning, causes serious performance dropping 3.36% on average. It again confirms the effectiveness of explicitly modeling the inherit building components and their semantic correlations from Web semi-structured data, especially for lists (a decrease of 5.12%). The semantic relation ablations elaborate on the contribution of each relations in semantics fusion among components: Among them, the Cell-Cell relations is proven to contribute most in semantic modeling for its largest performance reduction(1.74% in average). The GraSSLM w/o Header-Cell/Subject-Attribute Relation dropped 1.33% and 0.57% in average, indicating the GCN successfully utilized these relations in table modeling. Additionally, the replacement of the Bi-LSTM component reduces the overall performance the least (0.79% on average), but is still better than simply using pooling method for token aggregation.

4.3 Pre-training Strategy Analysis

To evaluate the proposed pre-training techniques, we train the original GraSSLM model with different objectives. Specifically, we apply only one pre-training objective each time and evaluate their performance on the three datasets. The evaluation results are shown in Table 4.

Pre-training Strategy	**Table-QM**	**List-QM**	**DTable-QM**
None	77.41	76.79	69.22
WCM	77.66 (0.25 ↑)	77.29 (0.50 ↑)	70.98 (1.76 ↑)
NPO	78.54 (1.13 ↑)	77.42 (0.63 ↑)	71.06 (1.84 ↑)
WCM+NPO	**79.04 (1.63 ↑)**	**77.61 (0.82 ↑)**	**71.19 (1.97 ↑)**

Table 4: Performance comparison of pre-training objectives. The metrics are in percentage with signs % omitted.

Each objective contributes to the performance improvement. When we solely use WCM as the pre-training objective, the performance is increased up to 1.97% on all three datasets. WCM successfully guides the model to learn reasonable token-level embedding. Via pre-training with NPO only, the performance is increased up to 1.84%. NPO allows the inherent semantic correlations among neighbor nodes

to be propagated via the structural graph connections. The combination of WCM and NPO achieves the largest performance increase (1.97%), showing that this pre-training strategy exploits the structural information in tables and lists the best.

5 Related Work

Early studies on Query-Table Matching adopt IR approaches. For example, Chakrabarti et al. (2020) and Pimplikar and Sarawagi (2012) convert Web tables into multi-field documents and apply document retrieval pipelines proposed in (Jurafsky and Martin, 2006; Paşca, 2003). Zhang and Balog (2018) propose to create semantic features at text level, concept level and entity level. These methods mainly consider the textual information in tables, but largely ignore the inherent structural information in tables.

In recent years, learning representations for semi-structured data has received increasing interest. Nishida et al. (2017) propose to apply convolutional models to Web table. The rationale is to consider a Web table as a matrix of text, analogous to an image of pixels. However, their model does not show strong performance, partly because the semantic relationship among neighbor cells in tables may be far more complex than the simple adjacency relation among neighbor pixels in an image. Herzig et al. (2020) propose TAPAS, a weakly supervised table parsing method. TAPAS models the structure information of tables by explicitly encoding rows and columns. Similarly, Yin et al. (2020) propose TABERT, which focuses on pre-training methods for the table QA task. The authors design a pipeline for learning row-level and column-level representations. Müller et al. (2019) also builds a graph representation on tables cells, focusing on optimizing cell answer selection. However, These works only model the row/column relations among table cells, without considering other relations including caption-content relation, header-cell relation and subject-attribute relation. In this work, we give a thorough categorization of the relations among all components in semi-structured data, and propose a graph model to incorporate all these relations.

Our work is also generally related to the broad areas of graph neural networks and pre-training techniques, we refer interested readers to (Wu et al., 2020) and (Qiu et al., 2020) for comprehensive surveys.

6 Conclusion

Semi-structured data on the Web, including tables and lists, present a rich source for Web QA. Most of the previous methods do not take full advantage of structural information in semi-structured data. In this paper, we propose a novel approach to model both textual and structural information in semi-structured data. Extensive experimental results verify the effectiveness of our approach.

References

Sreeram Balakrishnan, Alon Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying webtables in practice.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Michael J Cafarella, Alon Halevy, and Nodira Khoussainova. 2009. Data integration for the relational web. *Proceedings of the VLDB Endowment*, 2(1):1090–1101.

Kaushik Chakrabarti, Zhimin Chen, Siamak Shakeri, and Guihong Cao. 2020. Open domain question answering using web tables. *arXiv preprint arXiv:2001.03272*.

Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2017. Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221–230.

World Wide Web Consortium et al. 1999. Html 4.01 specification.

Eric Crestan and Patrick Pantel. 2011. Web-scale table census and classification. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 545–554.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, and Wolfgang Lehner. 2015. Building the dresden web table corpus: A classification approach. In *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, pages 41–50. IEEE.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2010. Why does unsupervised pre-training help deep learning? In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 201–208.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2019. Hierarchical graph network for multi-hop question answering. *arXiv preprint arXiv:1911.03631*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

D Jurafsky and J Martin. 2006. Speech and language processing (question answering chapter).

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Larissa R Lautert, Marcelo M Scheidt, and Carina F Dorneles. 2013. Web table taxonomy and formalization. *ACM SIGMOD Record*, 42(3):28–33.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Thomas Müller, Francesco Piccinno, Massimo Nicosia, Peter Shaw, and Yasemin Altun. 2019. Answering conversational questions on structured data without logical forms. *arXiv preprint arXiv:1908.11787*.

Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, and Yoshihiro Matsuo. 2017. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Marius Paşca. 2003. Open-domain question answering from large text collections.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *arXiv preprint arXiv:1207.0132*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *arXiv preprint arXiv:2003.08271*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Hao Wang, Xiaodong Zhang, Shuming Ma, Xu Sun, Houfeng Wang, and Mengxiang Wang. 2018. A neural question answering model based on semi-structured tables. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1941–1951.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference*, pages 1553–1562.