Text-to-Table: A New Way of Information Extraction

Anonymous ACL submission

Abstract

We study a new problem setting of information extraction (IE), referred to as text-to-table. In text-to-table, given a text, one creates a table or several tables expressing the main content of the text, while the model is learned from text-table pair data. The problem setting differs from those of the existing methods for IE.

First, the extraction can be carried out from long texts to large tables with complex structures. Second, the extraction is entirely data-driven, and there is no need to explicitly define the schemas. As far as we know, there has been no previous work that studies the problem. In this work, we formalize text-to-table as a sequence-to-sequence (seq2seq) problem.

We first employ a seq2seq model fine-tuned from a pre-trained language model to perform the task. We also develop a new method within the seq2seq approach, exploiting two additional techniques in table generation: table constraint and table relation embeddings. We consider text-to-table as an inverse problem of the well-studied table-to-text, and make use of four existing table-to-text datasets in our experiments on text-to-table. Experimental results show that the vanilla seq2seq model can outperform the baseline methods of using relation extraction and named entity extraction. The results also show that our method can further boost the performances of the vanilla seq2seq model. We further discuss the main challenges of the proposed task. The code and data will be made publicly available.

1 Introduction

Information extraction (IE) is a task that aims to extract information of interest from text data and represent the extracted information in a structured form. Traditional IE tasks include named entity recognition which recognizes entities and their types (Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016; Devlin et al., 2019), relation extraction which identifies the relationships between entities (Zheng et al., 2017; Zeng et al., 2018; Luan et al., 2019; Zhong and Chen, 2020), etc. Since the results of IE are structured, they can be easily used by computer systems in different applications such as text mining.

In this work, we study IE in a new setting, referred to as text-to-table. First, the system receives a training dataset containing text-table pairs. Each text-table pair contains a text and a table (or tables) representing information extracted from the text. The system learns a model for information extraction. Next, the system employs the learned model to conduct information extraction from a new text and outputs the result in a table (or tables). Figure 1 gives an example of text-to-table, where the input (above) is a report of a basketball game, and the output (below) is two tables summarizing the scores of the teams and players.
long text. As in the example in Figure 1, extraction of information is performed from the entire document. The extracted information contains multiple types of scores of teams and players in a basketball game structured in table format. Second, the schemas for extraction are implicitly included in the training data, and there is no need to explicitly define the schemas. This reduces the need for manual efforts for schema design and annotations.

Our work is inspired by research on the so-called table-to-text (or data-to-text) problem, which is the task of generating a description for a given table. Table-to-text is useful in applications where the content of a table needs to be described in natural language. Thus, text-to-table can be regarded as an inverse problem of table-to-text. However, there are also differences. Most notably, their applications are different. Text-to-table can be applied to document summarization, text mining, etc.

In this work, we formalize text-to-table as a sequence-to-sequence (seq2seq) task. More specifically, we translate the text into a sequence representation of a table (or tables), where the schema of the table is implicitly contained in the representation. We also build the seq2seq model on top of a pre-trained language model, which is the state-of-the-art approach for seq2seq tasks (Lewis et al., 2019; Raffel et al., 2020). Although the approach is a natural application of existing technologies, as far as we know, there has been no previous study to investigate to what extent the approach works. We also develop a new method for text-to-table within the seq2seq approach with two additional techniques, table constraint and table relation embeddings. Table constraint controls the creation of rows in a table and table relation embeddings affect the alignments between cells and their row headers and column headers. Both are to make the generated table well-formulated.

The approach to IE based on seq2seq has already been proposed. Methods for conducting individual tasks of relation extraction (Zeng et al., 2018; Nayak and Ng, 2020), named entity recognition (Chen and Moschitti, 2018; Yan et al., 2021), and event extraction (Lu et al., 2021) have been developed. Methods for jointly performing multiple tasks of named entity recognition, relation extraction, and event extraction have also been devised (Paolini et al., 2021). Most of the methods exploit suitable pre-trained models such as BERT. However, all the existing methods rely on pre-defined schemas for extraction. Moreover, their models are designed to extract information from short texts, rather than long texts, and extract information with simple structures (such as an entity and its type), rather than information with complicated structures (such as a table).

We conduct extensive experiments on four datasets. Results show that the vanilla seq2seq model fine-tuned from BART (Lewis et al., 2019) can outperform the state-of-the-art IE models fine-tuned from BERT (Devlin et al., 2019; Zhong and Chen, 2020). Furthermore, results show that our proposed approach to text-to-table with the two techniques can further improve the extraction accuracies. We also summarize the challenging issues with the seq2seq approach to text-to-table for future research.

Our contributions are summarized as follows:
1. We propose the new task of text-to-table for IE. We derive four new datasets for the task from existing datasets.
2. We formalize the task as a seq2seq problem and propose a new method within the seq2seq approach using the techniques of table constraint and table relation embeddings.
3. We conduct extensive experiments to verify the effectiveness of the proposed approach.

2 Related Work

Information Extraction (IE) is a task of extracting information (structured data) from a text (unstructured data). For example, named entity recognition (NER) recognizes entities appearing in a text. Relation extraction (RE) identifies the relationships between entities. Event extraction (EE) discovers events occurring in a text.

Traditionally, researchers formalize the task as a language understanding problem. The state-of-the-art methods for NER perform the task on the basis of the pre-trained language model BERT (Devlin et al., 2019). The pipeline approach to RE divides the problem into NER and relation classification, and conducts the two sub-tasks in a sequential manner (Zhong and Chen, 2020), while the end-to-end approach jointly carries out the two sub-tasks (Zheng et al., 2017; Zeng et al., 2018; Luan et al., 2019). The state-of-the-art methods for EE also employ BERT and usually jointly train the models with other tasks such as NER and RE (Wadden et al., 2019; Zhang et al., 2019; Lin et al., 2020). All the methods assume the use of pre-defined
schemata (e.g., entity types for NER, entity and relation types for RE, and event templates for EE). Besides, most methods are designed for extraction from short texts. Therefore, existing methods for IE cannot be directly applied to text-to-table.

Another series of related work is open information extraction (OpenIE), which aims to extract information from texts without relying on explicitly defined schemata (Banko et al., 2007; Wu and Weld, 2010; Mausam et al., 2012; Stanovsky et al., 2018; Zhan and Zhao, 2020). However, OpenIE aims to extract information with simple structures (i.e., relation tuples) from short texts, and the methods in OpenIE cannot be directly applied to text-to-table.

IE is also conducted at document level, referred to as doc-level IE. For example, some NER methods directly perform NER on a long document (Strubell et al., 2017; Luo et al., 2018), and others encode each sentence in a document, use attention to fuse document-level information, and perform NER on each sentence (Hu et al., 2020; Xu et al., 2018). There are also RE methods that predict the relationships between entities in a document (Yao et al., 2019; Nan et al., 2020a). However, existing doc-level IE approaches usually do not consider extraction of complex relations between many items.

**Sequence-to-sequence** (seq2seq) is the general problem of transforming one text into another text (Sutskever et al., 2014; Bahdanau et al., 2014), which includes machine translation, text summarization, etc. The use of the pre-trained language models of BART (Lewis et al., 2019) and T5 (Raffel et al., 2020) can significantly boost the performances of seq2seq, such as machine translation (Lewis et al., 2019; Raffel et al., 2020; Liu et al., 2020) and text summarization (Lewis et al., 2019; Raffel et al., 2020; Huang et al., 2020).

Recently, some researchers also formalize the IE problems as seq2seq, that is, transforming the input text into an internal representation. One advantage is that one can employ a single model to extract multiple types of information. Results show that this approach works better than or equally well as the traditional approach of language understanding, in RE (Zeng et al., 2018; Nayak and Ng, 2020), NER (Chen and Moschitti, 2018; Yan et al., 2021) and EE (Lu et al., 2021). Methods for jointly performing multiple tasks including NER, RE and EE have also been devised (Paolini et al., 2021).

**Data-to-text** aims to generate natural language descriptions from the input structured data such as sport commentaries (Wiseman et al., 2017). The structured data is usually represented as tables (Wiseman et al., 2017; Thomson et al., 2020; Chen et al., 2020), sets of table cells (Parikh et al., 2020; Bao et al., 2018), semantic representations (Novikova et al., 2017), or sets of relation tuples (Gardent et al., 2017; Nan et al., 2020b). The task requires the model to select the salient information from the data, organize it in a logical order, and generate an accurate and fluent natural language description (Wiseman et al., 2017). Data-to-text models usually adopt the encoder-decoder architecture. The encoders are specifically designed to model the input data, such as multi-layer perceptron (Puduppully et al., 2019a,b), recurrent neural network (Juraska et al., 2018; Liu et al., 2018; Shen et al., 2020), graph neural network (Marcheggiani and Perez-Beltrachini, 2018; Koncel-Kedziorski et al., 2019), or Transformer (Gong et al., 2019).

### 3 Problem Formulation

As shown in Figure 1, Text-to-table takes a text as input and produces a table or several tables to summarize the content of the text.

Formally, the input is a text denoted as \( x = x_1, x_2, \ldots, x_k \). The output is one table or multiple tables. For simplicity suppose that there is only one table denoted as \( T \). Further suppose that \( T \) has \( n_r \) rows and \( n_c \) columns. Thus, \( T \) contain \( n_r \times n_c \) cells, where the cell of row \( i \) and column \( j \) is a sequence of words \( t_{ij} = t_{ij,1}, t_{ij,2}, \ldots, t_{ij,t_{ij}} \).

There are three types of table: one that has both column headers and row headers, one that has only column headers, and one that only has row headers. For example, the player table in Figure 1 has both column headers (“Assists”, “Points”, etc) and row headers (“Al Horford”, “Isaiah Thomas”, etc). We let \( t_{1,j}, \quad j = 2, 3, \ldots, n_c \) to denote the column headers, let \( t_{i,1}, \quad i = 2, 3, \ldots, n_r \) to denote the row headers, and let \( t_{1,j}, \quad i = 2, 3, \ldots, n_r, j = 2, 3, \ldots, n_c \) to denote the non-header cells of the table. For example, in the player table in Figure 1, \( t_{1,2} = \text{Assists}, \quad t_{2,1} = \text{Al Horford}, \quad \text{and} \quad t_{2,2} = 5 \).

The information extracted via text-to-table can be leveraged in many different applications such as document summarization and text mining. For example, in Figure 1, one can quickly obtain the key information of the text by simply looking at the tables summarized from the text.

There are differences between text-to-table and...
We represent the entire table with a sequence of tokens, a separation token denoted as "\(\langle s\rangle\)" and a new-line token denoted as "\(\langle n\rangle\)". For a table \(t\), we represent each row \(t_i\) with a sequence of cells delimited by separation tokens:

\[
t_i = \langle s\rangle, t_{i,1}, \langle s\rangle, \ldots, \langle s\rangle, t_{i,n_c}, \langle s\rangle.
\]

We represent the entire table with a sequence of rows delimited by new-line tokens:

\[
t = \langle s\rangle, t_{1,1}, \langle s\rangle, \ldots, \langle s\rangle, t_{1,n_c}, \langle s\rangle, \langle n\rangle, \langle s\rangle, t_{2,1}, \langle s\rangle, \ldots, \langle s\rangle, t_{2,n_c}, \langle s\rangle, \langle n\rangle, \ldots, \langle s\rangle, t_{n,1}, \langle s\rangle, \ldots, \langle s\rangle, t_{n,n_c}, \langle s\rangle.
\]

Figure 2 shows the sequence of the player table in Figure 1. When there are multiple tables, we create a sequence of tables using the captions of the tables as delimiters.

Let \(x = x_1, \ldots, x_{|x|}\) and \(y = y_1, \ldots, y_{|y|}\) denote the input and output sequences respectively. In inference, the model generates the output sequence based on the input sequence. The model conducts generation in an auto-regressive way, which generates one token at each step based on the tokens it has generated so far. In training, we learn the model based on the text-table pairs \(\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}\). The objective of learning is to minimize the cross-entropy loss.

We refer to the method described above as "vanilla seq2seq". There is no guarantee, however, that the output sequence of vanilla seq2seq represents a well-formulated table. We add a post-processing step to ensure that the output sequence is a table. The post-processing method takes the first row generated as well-defined, deletes extra cells at the end of the other rows and inserts empty cells at the end of the other rows.

### 4.2 Techniques

We develop two techniques to improve table generation, called table constraint and table relation embeddings. We use "our method" to denote the seq2seq approach with these two techniques.¹

#### Table Constraint

Our method exploits a constraint in the decoding process to ensure that the output sequence represents a well-formulated table. Specifically, our method calculates the number of cells in the first row it generates, and then forces the following rows to contain the same number of cells.

#### Table Relation Embeddings

Our method also incorporates table relation embeddings including row relation embeddings and column relation embeddings into the self-attention of the Transformer decoder. Given a token in a non-header cell, the row relation embeddings \(\tau_r^K\) and \(\tau_r^V\) indicate which row header the token is aligned to, and the column relation embeddings \(\tau_c^K\) and \(\tau_c^V\) indicate which column header the token is aligned to.

¹Our method is able to generate the output containing multiple tables. This is discussed in Appendix C.
Let us consider the self-attention function in one block of Transformer decoder: at each position, self-attention only attends to the previous positions. For simplicity, let us only consider one head in the self-attention. At the \( t \)-th position, the input of self-attention is the sequence of representations \\
\( z = (z_1, \cdots, z_t) \) and the output is the sequence of representations \\
\( h = (h_1, \cdots, h_t) \), where \( z_i \in \mathbb{R}^d \) and \( h_i \in \mathbb{R}^d \) are the representations at the \( i \)-th position \( (i = 1, \cdots, t) \).

In a conventional Transformer decoder, self-attention is defined as follows,

\[
h_i = \left( \sum_{j=1}^{i} \alpha_{ij}(z_j W^V) \right) W^O,
\]

(3)

\[
\alpha_{ij} = \frac{e^{e_{ij}}}{\sum_{j=1}^{i} e^{e_{ij}}}, e_{ij} = \frac{(z_i W^Q)(z_j W^K)^T}{\sqrt{d_k}},
\]

(4)

\[ i = 1, \cdots, t, \quad j = 1, \cdots, i \]

where \( W^Q, W^K, W^V \in \mathbb{R}^{d \times d_k} \) are the query, key, and value weight matrices respectively, and \( W^O \in \mathbb{R}^{d_k \times d} \) is the output weight matrix.

In our method, self-attention is defined as:

\[
h_i = \left( \sum_{j=1}^{i} \alpha_{ij}(z_j W^V + r_{ij}^V) \right) W^O,
\]

(5)

\[
\alpha_{ij} = \frac{e^{e_{ij}}}{\sum_{j=1}^{i} e^{e_{ij}}}, e_{ij} = \frac{(z_i W^Q)(z_j W^K + r_{ij}^K)^T}{\sqrt{d_k}},
\]

(6)

\[ i = 1, \cdots, t, \quad j = 1, \cdots, i \]

where \( r_{ij}^K \) and \( r_{ij}^V \) are relation vectors representing the relationship between the \( i \)-th position and the \( j \)-th position.

The relation vectors \( r_{ij}^K \) and \( r_{ij}^V \) are defined as follows. For the token at the \( i \)-th position, if the token at the \( j \)-th position is a part of its row header, then \( r_{ij}^K \) and \( r_{ij}^V \) are set to the row relation embeddings \( \tau_{ij}^K \) and \( \tau_{ij}^V \). Similarly, for the token at the \( i \)-th position, if the token at the \( j \)-th position is a part of its column header, then \( r_{ij}^K \) and \( r_{ij}^V \) are set to the column relation embeddings \( \tau_{ij}^K \) and \( \tau_{ij}^V \). Otherwise, \( r_{ij}^K \) and \( r_{ij}^V \) are set to 0. In inference, to identify the row header or the column header of a token, we parse the sequence generated so far to create a partial table using the new-line tokens and separation tokens in the sequence. Figure 3 illustrates how relation vectors are constructed.

5 Experiments

5.1 Datasets

We make use of four existing datasets which are traditionally utilized for data-to-text: Rotowire (Wiesman et al., 2017), E2E (Novikova et al., 2017), WikiTableText (Bao et al., 2018), and WikiBio (Lebret et al., 2016). In each dataset, we filter out the content in the tables that does not appear in the texts. We plan to make the processed datasets publicly available for future research. Table 2 gives the statistics of the Rotowire dataset and Table 1 gives the statistics of the other three datasets.

Rotowire is from the sports domain. Each instance is composed of a text and two tables, where the text is a report of a basketball game and the two tables represent the scores of teams and players respectively (cf., Figure 1). Each table has column headers describing the types of scores, and row headers describing the names of teams or players. The texts are long and may contain irrelevant information such as the performance of players in other games. Therefore, this is a challenging dataset.

E2E is from the restaurant domain. Each instance is a pair of short text and automatically constructed table, where the text is a description of a restaurant, and the table has two columns with column headers summarizing the characteristics of the restaurant. The tables are automatically constructed, where the texts in the tables are from a limited set and thus are lack of diversity.

WikiTableText is an open-domain dataset. Each instance includes a text and a table, where the text is a description and the table has a row and two columns with column headers, collected from a Wikipedia infobox. The texts are short and contain information similar to that in the tables.

WikiBio is extracted from the Wikipedia biography pages. Each instance consists of a text and a table, where the text is the introduction of Wikipedia page\(^2\) and the table is from the infobox of Wikipedia page and has two columns with column headers. The input texts are usually long and contain more information than the tables.

5.2 Procedure

Methods: We conduct experiments with vanilla seq2seq and our method, as well as baselines.

We know of no existing method that can be directly employed in text-to-table. For each dataset,
we adopt Transformer (Vaswani et al., 2017) as we first define the schemas based on the training data, then use an existing method of relation extraction (RE) or named entity extraction (NER) to extract information, and finally create tables based on the schemas and extracted information. We take it as the baseline for the dataset. No baseline can be applied to all four datasets. For RE, we use PURE, a state-of-the-art method (Zhong and Chen, 2020). For NER, we use BERT (Devlin et al., 2019).

**Training:** For vanilla seq2seq and our method, we adopt Transformer (Vaswani et al., 2017) as the model and fine-tune the models from BART-base. We also experiment with BART-large. For RE and NER, we fine-tune the models from BERT-base-uncased. All models are trained with Adam optimizer until convergence. Hyper-parameters are shown in Appendix A. For the small datasets of Rotowire and WikiTableText, we run experiments five times with different random seeds and take average of results to reduce variance.

**Evaluation:** We evaluate the performance of a method based on the number of correct non-header cells in the tables. To judge whether a cell is correctly generated in the table, we use not only its content but also its row header and column header to ensure that the cell is on the right row and right column. Exact match is used to compare the content of the generated cell and the ground truth. We adopt precision, recall, and F1 score as evaluation measures. We calculate the measures on each generated table and then take the average on all tables. This evaluation assumes that the ordering of rows and columns is not important. We find that this assumption is applicable to the four datasets and many real-world scenarios. We also evaluate the percentage of output sequences that cannot represent well-formulated tables, referred to as error rate.

### 5.3 Results on Rotowire

Table 3 shows the results on the Rotowire dataset. One can see that in terms of F1 score, our method performs the best followed by vanilla seq2seq, and both outperform the baselines of doc-level RE and sent-level RE. The RE baselines perform quite well, but they heavily rely on rules and cannot beat the seq2seq approach. Among them the doc-level RE performs better than sent-level RE, because some information in Rotowire can only be extracted when cross-sentence context is provided. We implement two baselines of RE, namely doc-level RE and sent-level RE. We take team names, player names, and numbers of scores as entities and take types of scores as relations. Sent-level RE predicts the relations between entities within each sentence. Doc-level RE predicts the relations between entities within a window (the window size is 12 entities) and uses the approximation model proposed by Zhong and Chen (2020) to speed up inference.

### 5.4 Results on E2E, WikiTableText and WikiBio

Table 4 shows the results of our method, vanilla seq2seq, and the baseline of NER on E2E, WikiTableText, and WikiBio. Again, the seq2seq approach outperforms the baseline. The NER baseline has slightly higher precision, but the seq2seq approach has significantly higher recall and F1. Our method and vanilla seq2seq are comparable, because the table structures in the three datasets are very simple (there are only two columns in the tables), and the use of the two techniques does not further improve the performances. The NER baseline has high precision but low recall, mainly because NER can only make the right decision when it is clear.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Valid</th>
<th>Test</th>
<th># of tokens</th>
<th># of rows</th>
<th># of columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2E</td>
<td>10.0k</td>
<td>1.3k</td>
<td>2.0k</td>
<td>4.58</td>
<td>2.00</td>
<td>4.70</td>
</tr>
<tr>
<td>WikiTableText</td>
<td>122.30</td>
<td>351.05</td>
<td>42.1k</td>
<td>4.20</td>
<td>2.00</td>
<td>7.26</td>
</tr>
<tr>
<td>WikiBio</td>
<td>582.7k</td>
<td>72.8k</td>
<td>72.8k</td>
<td>24.90</td>
<td>4.20</td>
<td>2.00</td>
</tr>
</tbody>
</table>

Table 1: Statistics of E2E, WikiTableText, and WikiBio datasets, including number of instances in training, validation, and test sets, number of BPE tokens per instance and number of rows per instance.
Table 3: Results of our method, vanilla seq2seq, and the baselines of doc-level RE and sent-level RE, on Rotowire.

<table>
<thead>
<tr>
<th></th>
<th>Team</th>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent-level RE</td>
<td>81.05</td>
<td>75.29</td>
</tr>
<tr>
<td>Doc-level RE</td>
<td>78.57</td>
<td>74.74</td>
</tr>
<tr>
<td>Vanilla seq2seq</td>
<td>84.05</td>
<td>83.59</td>
</tr>
<tr>
<td>Our method</td>
<td>84.40</td>
<td>84.01</td>
</tr>
</tbody>
</table>

Table 4: Results of our method, vanilla seq2seq, and the baseline of NER, on E2E, WikiTableText and WikiBio.

<table>
<thead>
<tr>
<th></th>
<th>E2E</th>
<th>WikiTableText</th>
<th>WikiBio</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER</td>
<td>99.39</td>
<td>84.99</td>
<td>90.80</td>
</tr>
<tr>
<td>Vanilla seq2seq</td>
<td>97.95</td>
<td>97.87</td>
<td>97.87</td>
</tr>
<tr>
<td>Our method</td>
<td>97.97</td>
<td>97.89</td>
<td>97.88</td>
</tr>
</tbody>
</table>

Table 5: Results of ablation study on our method by excluding pre-trained language model (Pre), table constraint (TC) and table relation embeddings (TRE). We conduct a significance test to check whether the performance is significantly better than vanilla seq2seq with pre-trained language models (i.e., with Pre but without TC or TRE). † and ‡ represent p < 0.05 and p < 0.01 respectively.

<table>
<thead>
<tr>
<th></th>
<th>Rotowire/Team</th>
<th>Rotowire/Player</th>
<th>E2E</th>
<th>WikiTableText</th>
<th>WikiBio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre TC TRE</td>
<td>Rotowire/Team</td>
<td>Rotowire/Player</td>
<td>E2E</td>
<td>WikiTableText</td>
<td>WikiBio</td>
</tr>
<tr>
<td>¬ × × ¬</td>
<td>28.05</td>
<td>7.75</td>
<td>94.45</td>
<td>46.37</td>
<td>67.51</td>
</tr>
<tr>
<td>× × × ×</td>
<td>30.61</td>
<td>10.67</td>
<td>95.53</td>
<td>47.13</td>
<td>67.43</td>
</tr>
<tr>
<td>× × × × †</td>
<td>82.97</td>
<td>81.96</td>
<td>97.87</td>
<td>59.26</td>
<td>68.98</td>
</tr>
<tr>
<td>× × × × ‡</td>
<td>83.09 †</td>
<td>82.24 †</td>
<td>97.88</td>
<td>59.29 †</td>
<td>68.98</td>
</tr>
<tr>
<td>× × × × †</td>
<td>83.30 †</td>
<td>82.50 †</td>
<td>97.87</td>
<td>59.12</td>
<td>69.02</td>
</tr>
<tr>
<td>× × × × † ‡</td>
<td>83.36 †</td>
<td>82.53 †</td>
<td>97.88</td>
<td>59.14</td>
<td>69.02</td>
</tr>
</tbody>
</table>

We implement the baseline of NER in the following way. We view the non-head cells in the tables as entities and their row headers as entity types. In training, we match the non-head cells into the texts and take them as “entities” in the texts. Only a proportion of the non-header cells can be matched into the texts (85% for E2E, 74% for WikiTableText, and 69% for WikiBio).

5.5 Additional Study

We carry out ablation study on our method. Specifically, we exclude pre-trained language model, table constraint (TC) and table relation embeddings (TRE) from our method. Note that our method without TC and TRE is equivalent to vanilla seq2seq. Table 5 gives the results on the four datasets.

It can be seen that the use of both TC and TRE can significantly improve the performance on Rotowire, which indicates that our method is particularly effective when the tables are large with many rows and columns. There are not significant improvements on E2E, WikiTableText, and WikiTableText, apparently because formulation of tables is easy for the three datasets. Therefore, we conclude that the two techniques of TC and TRE are helpful when the task is difficult.

The use of pre-trained language model can boost the performance on all datasets, especially on Rotowire and WikiTableText. This indicates that pre-trained language model is particularly helpful when the task is difficult and the size of training data is small.

We observe that vanilla seq2seq makes more formatting errors than our method, especially on player tables in Rotowire that have a large number of columns. It indicates that for vanilla seq2seq, it is difficult to keep track of the columns in each row and make alignments with the column headers. In contrast, the two techniques of our method can help effectively cope with the problem. Figure 4 shows a bad case of vanilla seq2seq, where the model correctly infers the column of “assists” but fails to infer the columns of “personal fouls”, “points”, and “total rebounds” for the row of “Rajon Rondo”. In contrast, our method can successfully handle the case, because TC can eliminate the incorrectly formatted output, and TRE can make correct align-
Table 6: Results of our method and vanilla seq2seq with base and large BART models on all four datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Rotowire/Team</th>
<th>Rotowire/Player</th>
<th>E2E</th>
<th>WikiTableText</th>
<th>WikiBio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla seq2seq (BART base)</td>
<td>82.97</td>
<td>81.96</td>
<td>97.87</td>
<td>59.26</td>
<td>68.98</td>
</tr>
<tr>
<td>Our method (BART base)</td>
<td>83.36</td>
<td>82.53</td>
<td>97.88</td>
<td>59.14</td>
<td>69.02</td>
</tr>
<tr>
<td>Vanilla seq2seq (BART large)</td>
<td>86.31</td>
<td>86.59</td>
<td>97.94</td>
<td>62.71</td>
<td>69.66</td>
</tr>
<tr>
<td>Our method (BART large)</td>
<td>86.31</td>
<td>86.83</td>
<td>97.90</td>
<td>62.41</td>
<td>69.71</td>
</tr>
</tbody>
</table>

Table 6: Results of our method and vanilla seq2seq with base and large BART models on all four datasets.

Figure 4: A bad case generated by vanilla seq2seq. The assists, points and total rebounds of Rajon Rondo should be 18, 7 and 8 respectively. The model generates one less column between “Assists” and “Personal fouls”.

ments with the column headers.

We also investigate the effect of the scale of pre-trained language model BART. We use both BART-base and BART-large and conduct fine-tuning on top of them for vanilla seq2seq and our method. Table 6 gives the results on the four datasets. The results show that the use of BART-large can further boost the performances on all four datasets, indicating that it is better to use larger pre-trained models when computation cost is not an issue.

5.6 Discussions

We analyze the experimental results on the four datasets and identify five challenging issues.

(1) Text Diversity: Extraction of the same content from different expressions is one challenge. For example, the use of synonyms is very common in Rotowire. The team of “Knicks” is often referred to as “New York”, its home city. Identification of the same entities from different expressions is needed in the task.

(2) Text Redundancy: There are cases such as those in WikiBio, in which the texts contain much redundant information. This poses a challenge to the text-to-table model to have a strong ability in summarization. It seems that the seq2seq approach works well to some extent but further improvement is undoubtedly necessary.

(3) Large Table: The tables in Rotowire have large numbers of columns, and the extraction from them is challenging even for our method of using TC and TRE.

(4) Background Knowledge: WikiTableText and WikiBio are from open domain. Thus, performing text-to-table on such kind of datasets require the use of much background knowledge. A possible way to address this challenge is to use more powerful pre-trained language models or external knowledge bases.

(5) Reasoning: Sometimes the information is not explicitly presented in the text, and reasoning is required to conduct correct extraction. For example, an article in Rotowire reports a game between the two teams “Nets” and “Wizards”. From the sentence: “The Nets seized control of this game from the very start, opening up a 31 - 14 lead after the first quarter”, humans can infer that the point of “Wizards” is 14, which is still difficult for machines.

6 Conclusion

We propose employing text-to-table as a new way of information extraction (IE), which extracts information of interest from the input text and summarizes the extracted information in tables. The advantage of the approach is that one can easily conduct information extraction from either short texts or long texts to create simple tables or complex tables without explicitly defining the schemas. Text-to-table can be viewed as an inverse problem of table-to-text. We formalize text-to-table as a sequence-to-sequence problem on top of a pre-trained model. We further propose an improved method using a seq2seq model and table constraint and table relation embeddings techniques. We conduct experiments on four datasets derived from existing table-to-text datasets. The results demonstrate that our proposed approach outperforms existing methods using conventional IE techniques. We further analyze the challenges of text-to-table for future study. The issues include diversity of text, redundancy of text, large-table, background knowledge, and reasoning.
References


Guohai Xu, Chengyu Wang, and Xiaofeng He. 2018. Improving clinical named entity recognition with global neural attention. In Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data, pages 264–279. Springer.


A Hyper-parameters

We list the hyper-parameters of the pre-trained models in Table 7. The training hyper-parameters for BART-base model in vanilla seq2seq and our method are listed in Table 8.

B Table Constraint Algorithm

The pseudo-codes for table constraint are in Algorithm 1.

C Our Method with Multiple Tables

Our method is able to generate the output containing multiple tables. For example, in Rotowire dataset, the output data contains two tables representing the scores of teams and players respectively. In this section, we illustrate how our method works for Rotowire dataset as a special case.

To represent the tables with a sequence, we use captions as delimiters. For Rotowire, as shown in Figure 1, the first table is the team table, and its caption is “Team:”. The second table is the player table, and its caption is “Player:”. Let $t_{team}$ and $t_{player}$ denote the table and player tables respectively. Therefore, the sequence representation is “Team: (n) $t_{team}$ (n) Player: (n) $t_{player}$.”

For table constraint (TC), we only use TC when the seq2seq model is generating a table. When generating a caption, we do not pose any constraints to the decoding process. Since the captions do not start with the separation token $\langle s \rangle$, if the current line starts with the separation token $\langle s \rangle$, then the model is generating a table. Otherwise, it is generating a caption.

For table relation embeddings (TRE), we calculate the relation vectors separately for each table. However, the parameters including the row relation embeddings (i.e., $r^{t}$ and $r^{c}$) and the column
Table 7: The hyper-parameters of the pre-trained models in our experiments. We list the number of layers, hidden dimensions (hidden dim.), heads, and parameters. BART-base and BART-large are used for vanilla seq2seq and our method, while BERT-base-uncased is used for the baselines of RE and NER.

<table>
<thead>
<tr>
<th>Pre-trained model</th>
<th>Methods</th>
<th>layers</th>
<th>hidden dim.</th>
<th>heads</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>BART-base</td>
<td>Vanilla seq2seq and ours</td>
<td>12</td>
<td>768</td>
<td>16</td>
<td>139M</td>
</tr>
<tr>
<td>BART-large</td>
<td>Vanilla seq2seq and ours</td>
<td>24</td>
<td>1024</td>
<td>16</td>
<td>406M</td>
</tr>
<tr>
<td>BERT-base-uncased</td>
<td>NER and RE</td>
<td>12</td>
<td>768</td>
<td>12</td>
<td>110M</td>
</tr>
</tbody>
</table>

Table 8: The training hyper-parameters for BART-base model on all four datasets. We list the warmup updates (warmup), total updates (total upd.), learning rate (lr), and batch size (bsz, in terms of how many tokens per batch).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>warmup</th>
<th>total upd.</th>
<th>lr</th>
<th>bsz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotowire</td>
<td>400</td>
<td>8000</td>
<td>3e-05</td>
<td>4096</td>
</tr>
<tr>
<td>E2E</td>
<td>400</td>
<td>8000</td>
<td>1e-05</td>
<td>4096</td>
</tr>
<tr>
<td>WikiTableText</td>
<td>2000</td>
<td>8000</td>
<td>1e-04</td>
<td>4096</td>
</tr>
<tr>
<td>WikiBio</td>
<td>4000</td>
<td>40000</td>
<td>1e-04</td>
<td>4096</td>
</tr>
</tbody>
</table>

Algorithm 1: Decoding using table constraint. \(\langle\text{eos}\rangle, \langle s\rangle, \langle n\rangle\) denote the end of sentence, separation token, and new-line token respectively. Seq2seq denotes the seq2seq model. Decode denotes the decoding algorithm such as beam search and greedy search.

```
Input: \(x = [x_1, x_2, \cdots, x_{|x|}]\)
Output: \(y = [y_1, y_2, \cdots, y_{|y|}]\)

\(1\) \(y \leftarrow []\)
\(2\)
\(3\) repeat
  \(/*\) generates the first row: only allows generation of \(\langle n\rangle\) or \(\langle \text{eos}\rangle\) after \(\langle s\rangle\)
  \(4\) \(p() \leftarrow \text{seq2seq}(x, y)\)
  \(5\) if \(y_{|y|} \neq \langle s\rangle\) then
    \(6\) \(p(\langle n\rangle) \leftarrow 0, p(\langle \text{eos}\rangle) \leftarrow 0\)
    \(7\) \(y.append(\text{decode}(p))\)
  until \(y_{|y|} = \langle n\rangle\) or \(y_{|y|} = \langle \text{eos}\rangle\)
  \(9\) if \(y_{|y|} = \langle \text{eos}\rangle\) then
    \(10\) return \(y\)
  \(11\) \(n_c \leftarrow \text{number of cells of the first row}\)
  \(12\) repeat
    \(/*\) generates the next rows: each row contains exactly \(n_c\) cells
    \(13\) repeat
      \(/*\) generates a row
      \(14\) \(p() \leftarrow \text{seq2seq}(x, y)\)
      if current row has \(n_c\) columns then
        \(15\) \(p(t) \leftarrow 0, \forall t \neq \langle \text{eos}\rangle\) and \(t \neq \langle n\rangle\)
      else
        \(16\) \(p(\langle n\rangle) \leftarrow 0, p(\langle \text{eos}\rangle) \leftarrow 0\)
        \(y.append(\text{decode}(p))\)
      until \(y_{|y|} = \langle n\rangle\) or \(y_{|y|} = \langle \text{eos}\rangle\)
      if \(y_{|y|} = \langle \text{eos}\rangle\) then
        \(22\) return \(y\)
      \(23\)
```

D Evaluation Details

To evaluate a text-to-table system, we adopt precision, recall, and F1 score as evaluation measures. We calculate the measures on each generated table and then take the average on all tables.

Specifically, we represent each non-header cell as a tuple containing the row header, column header, and cell content. Then, we take the collection of cell tuples in the ground truth table as reference, and calculate the precision, recall, and F1 score of the predicted cell tuples. We use exact match to check whether a predicted non-header cell is correct. A predicted non-header cell will be considered as a true positive only when it is exactly the same as the ground truth, that is, when they have the same row header, cell header, and content. In other words, we use not only its content but also its row header and column header to ensure that the cell is on the right row and right column. A limitation of exact match is that it will fail to consider the cases where the prediction is a synonym of the reference. However, we observe that the row/column names and cell contents in datasets are quite consistent, so such mistakes are uncommon. Therefore, we use exact match for simplicity.

As shown in Figure 1, the tables in the dataset contain some empty cells, that is, cells with word sequences of zero length. These cells do not contain actual information and are only used as place-holders. Therefore, for both the golden reference and the prediction, we ignore the empty cells and...
Figure 5: An example of NER data on WikiBio dataset. Each row header is an entity type, and each non-header cell is an entity.

Figure 6: An example of RE data from Rotowire dataset. “Al Horford” and “15” are entities, and “Points” is the relation type.

E Information Extraction Baselines

E.1 Relation Extraction

To use relation extraction (RE) as our baseline for Rotowire dataset, we take team names, player names, and numbers of scores as entities and take types of scores as relations. An example relation is shown in Figure 6, which can be represented as a relation tuple (Al Horford, Points, 15). “Al Horford” is the subject entity, “15” is the object entity, and “Points” is one of the pre-defined relation types. There are 38 relations in total.

To create synthetic training data, we match the player names, team names and score numbers to the texts. We adapt the rules provided by Wiseman et al. (2017) which is able to conduct fuzzy match.

E.2 Named Entity Recognition

We use named entity recognition (NER) as our baseline for E2E, WikiTableText, and WikiBio datasets. Specifically, since each table is a two-column table with a header column, we consider the row header as entity type and the non-header cells as entity mentions. An example is shown in Figure 5. For the row with a header “name” and a non-header cell “majda vrhnovnik”, we take “majda vrhnovnik” as an entity with the type “name”. Here, “name” is one of the pre-defined entity types. We collect all headers in the training set to collect the entity types. We have 7 entity types for E2E, 2262 entity types for WikiTableText, and 2272 entity types for WikiBio.

To create synthetic training data, we match the contents of non-header cells to the texts. However, the data is usually paraphrased or even abstracted from the text, so not all non-header cells can be matched to the text. We match 85% non-header cells for E2E, 74% for WikiTable, and 69% for WikiBio.

F Detailed Cases for Challenges

In this section, we provide cases for the challenges discussed in Section 5.6.

(1) Text Diversity: Extraction of the same content from different expressions is one challenge. For example, the use of synonyms is very common in Rotowire. Figure 7 illustrates the use of synonyms for the example in Figure 1. The red color denotes the team of “Knicks”, which is often referred to as “New York”, its home city. The blue color denotes the team of “Celtics”, which is often referred to as “Boston”, its home city.
**Figure 8:** An example from WikiBio dataset to illustrate the challenges of text redundancy and background knowledge. Only the highlighted information is captured in the output table. Other information such as the experience of Philippe Adnot is redundant. Moreover, the system should have background knowledge about the French political system to extract information about the constituency of Philippe Adnot.

<table>
<thead>
<tr>
<th>Name</th>
<th>Philippe Adnot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birth Place</td>
<td>France</td>
</tr>
<tr>
<td>Occupation</td>
<td>Farmer</td>
</tr>
<tr>
<td>Residence</td>
<td>France</td>
</tr>
<tr>
<td>Birth Date</td>
<td>25 August 1945</td>
</tr>
<tr>
<td>Constituency</td>
<td>Aube Canton de Méry-sur-Seine</td>
</tr>
</tbody>
</table>

The Denver Nuggets defeated the Milwaukee Bucks, 121 - 117, at Pepsi Center on Friday. It was n't the ideal win for Denver considering they nearly blew a monster lead, but they were able to hold on for a much needed victory. In fact, the Nuggets led by 22 in the third quarter, before allowing the Bucks to get it within two points late in the fourth. Denver were able to hold on though, despite missing their last four free - throws. Rebounding was the biggest factor in the game, with Denver winning that battle, 48 - 38. The Nuggets also dominated the assist - to - turnover ratio, recording five more assists and committing four less turnovers. The Bucks (21 - 28) put in a valiant effort to pull out a comeback win, but fell just shy to mark their 10th loss in 11 games. Jabari Parker was Milwaukee's best player, as he accrued 27 points, 11 rebounds, four assists and two steals. Giannis Antetokounmpo was n't far behind, collecting 23 points, eight rebounds, five assists, two steals and two blocks. John Henson was the only other Buck in double figures, as he amassed 16 points, five rebounds and four blocks. The Nuggets (22 - 27) have now won eight of their last 12 games, as they could n't be happier to see Nikola Jokic back and healthy. Jokic recorded his first career triple - double in the win. As he accumulated 21 points, 13 rebounds and 11 assists. Wilson Chandler started for Danilo Gallinari (groin) and tallied 23 points, as they rebound, three assists, two steals and four blocks. Kenneth Faried recorded a double - double, totaling 19 points and 11 rebounds. Jamal Murray was the leading scorer off the bench, dropping 18 points on 5 - of - 9 from the field.

<table>
<thead>
<tr>
<th>Team</th>
<th>Losses</th>
<th>Total Points</th>
<th>Points in 2nd Quarter</th>
<th>Wins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nuggets</td>
<td>27</td>
<td>121</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>Bucks</td>
<td>28</td>
<td>117</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Player</th>
<th>Assists</th>
<th>Blocks</th>
<th>Field Goals Attempted</th>
<th>Field Goals Made</th>
<th>Points</th>
<th>Total Rebounds</th>
<th>Steals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Giannis Antetokounmpo</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
<td>23</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Jabari Parker</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td>27</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>John Henson</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td>16</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Wilson Chandler</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td>23</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Kenneth Faried</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td>Nikola Jokic</td>
<td>11</td>
<td>2</td>
<td></td>
<td></td>
<td>9</td>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>Jamal Murray</td>
<td>11</td>
<td>2</td>
<td></td>
<td></td>
<td>9</td>
<td>5</td>
<td>18</td>
</tr>
</tbody>
</table>

Figure 9: The team table has 3 rows and 4 columns, and the player table has 8 rows and 8 columns.

“Knicks” is often referred to as “New York”, its home city. Similarly, “Celtics” is often referred to as “Boston”, its home city. Identification of the same entities from different expressions is needed in the task.

(2) Text Redundancy: There are cases such as those in WikiBio, in which the texts contain much redundant information. An example is shown in Figure 8, where only the highlighted information is captured in the output table. Other information such as the experience of Philippe Adnot is redundant. This poses a challenge to the text-to-table model to have a strong ability in summarization. It seems that the seq2seq approach works well to some extent but further improvement is undoubtedly necessary.

(3) Large Table: The tables in Rotowire have large numbers of columns, so extraction from them is challenging even for our method of using TC and TRE. As presented in Table 2, team tables have 2.71 rows and 4.84 columns on average, and player tables have 7.26 rows and 8.75 columns on average. An example is shown in Figure 9, where the team table has 3 rows and 4 columns, and the player table has 8 rows and 8 columns.

(4) Background Knowledge: WikiTableText and WikiBio are from open domain. Thus, performing text-to-table on such kind of datasets require...
the use of much background knowledge. Also in Figure 8, the extraction system should have background knowledge about the French political system in order to extract information about the constituency of Philippe Adnot. A possible way to address this challenge is to use more powerful pre-trained language models or external knowledge bases.

(5) Reasoning: Sometimes the information is not explicitly presented in the text, and reasoning is required to conduct correct extraction. For example, as shown in Figure 10, an article in Rotowire reports a game between the two teams “Nets” and “Wizards”. From the sentence: “The Nets seized control of this game from the very start, opening up a 31 - 14 lead after the first quarter”, humans can infer that the point of “Wizards” is 14, which is still difficult for machines.

<table>
<thead>
<tr>
<th>Team</th>
<th>Points in 1st quarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nets</td>
<td>31</td>
</tr>
<tr>
<td>Wizards</td>
<td>14</td>
</tr>
</tbody>
</table>

Figure 10: An example from Rotowire which requires reasoning to perform information extraction. The article in Rotowire reports a game between the two teams “Nets” and “Wizards”. From the sentence: “The Nets seized control of this game from the very start, opening up a 31 - 14 lead after the first quarter”, humans can infer that the point of “Wizards” is 14, which is still difficult for machines.