Numerical tables are widely used to present experimental results in scientific papers. For table understanding, a metric-type is essential to discriminate numbers in the tables. Herein, we introduce a new information extraction task, i.e., metric-type identification from multilevel header numerical tables, and provide a dataset extracted from scientific papers comprising header tables, captions, and metric-types. We propose joint-learning neural classification and generation schemes featuring pointer-generator-based and pretrained-based models. Our results show that the joint models can manage both in-header and out-of-header metric-type identification problems. Furthermore, transfer learning using fine-tuned pretrained-based models successfully improves the performance. The domain-specific of BERT-based model, SciBERT, achieves the best performance. Results achieved by a fine-tuned T5-based model are comparable to those obtained using our BERT-based model under a multitask setting.

Key Words: Metric-Type Identification, Table Understanding, Information Extraction

1 Introduction

Tables are an effective tool for presenting data efficiently in rows and columns. In scientific papers, numerical tables are typically used to present experimental results to facilitate data analysis. Examples of numerical tables presented in scientific papers are shown in Figure 1.

Multiple categories can be represented in table headers by incorporating several header sets in a hierarchical view; this is known as multilevel header tables. The presentation of tables in scientific papers must adhere to strict guidelines. For example, similar types of texts are to be written at the same level of header. Figure 1a shows a multilevel header example in the column section, with the task (Task 1 and Task 2) presented in the first header-level and the metric (Prec and Rec) in the second. The table contains a row header specifying the model (Model A, Model B, Model C, and Model D). The header information is typically limited owing to the unknown table schema. However, we assume that tables presented in scientific papers adhere to the rule whereby similar types of header names are categorized at the same header level.
To understand numbers in the tables, metric-types are important for discriminating the numbers. A comparison between numbers is applied for numbers in the same metric-type with different categories. For the table shown in Figure 1a, we cannot compare the number 60 for Model A in the first column with 60 in the second column because they represent different metric-types: Prec and Rec. Computing numbers of different metric-types will result in inaccurate analyses.

Header names may be written differently in different tables, e.g., using abbreviations such as p, pre, or prec to refer to precision. Due to the lexical diversity of header names, metric-type identification becomes more challenging. Using rule-based metric-type tagging or a limited set of metric-types in a dictionary is insufficient to encompass the diversity of metric-types. As tables presented in scientific papers are typically provided with logical captions and a logical categorization of the header level, we introduce a metric-type identification task that locates the metric-type in the headers using the caption and header names as inputs. For the example shown in Figure 1a, the metric-type is indicated at the second level of the column header.

Furthermore, we consider tables that do not include metric-types in their header (out-of-headers), as shown in Figure 1b. In these cases, the metric-types are provided in the caption. To consider tables with metric-types located and not located in the headers, we propose a joint framework of metric-type location prediction and metric-type token generation for the metric-type identification task in multilevel header tables. We adopted a pointer-generator (See et al. 2017) to generate metric-type tokens, combined with a softmax layer to predict the metric-type location.

We collected tables from scientific papers and hired workers to identify the metric-types. Because our annotated dataset is limited, we proposed transfer learning by fine-tuning available pretrained models trained on a large corpus in our task. We used BERT, a pretrained model with

| Models  | Task 1 | Task 2 |
|---------|--------|--------|
|         | Prec   | Rec    |
| Model A | 60     | 60     |
| Model B | 70     | 70     |
| Model C | 80     | 80     |
| Model D | 90     | 90     |

Table X. Model comparison in Task 1 and 2.

| Models  | Task 1 | Task 2 |
|---------|--------|--------|
|         | Prec   | Rec    |
| Model A | 60     | 65     |
| Model B | 70     | 75     |
| Model C | 80     | 85     |
| Model D | 90     | 95     |

Table Y. Model comparison in Task 1 and 2 (F-score).

(a) Metric-type in header
(b) No metric-type in header

Fig. 1 Examples of tables presented in scientific papers. Bold indicates metric-type.
bi-directional transformer encoders, to take advantage of its ability in producing more contexts from two directions. To perform prediction and generation tasks for solving our metric-type identification problem, we fine-tuned a pretrained encoder-decoder T5, which successfully solved multitask NLP problems with its unified framework.

Our contributions are as follows:

• We introduce a metric-type identification task for multilevel header tables and propose joint location prediction and generation models to solve the task.

• We provide a dataset comprising multilevel header numerical tables, captions, and metric-types extracted from scientific papers. Our dataset is publicly available.1

• We introduce a multilevel header table encoder mechanism to obtain table header representations and propose a pointer-generator-based model to cover out-of-headers in the metric-type identification task.

• We fine-tune a general pretrained encoder (BERT) and a domain-specific encoder (SciBERT) in our task and present the experimental results. We show that models incorporating the pretrained encoders yield significant performance gains, particularly domain-specific encoders.

• We fine-tune a pretrained encoder-decoder, T5, for our task in a multitask setting and present the experimental results.

2 Related Studies

Table information extraction is beneficial for covering unknown table schemes and understanding the table contents. Milosevic et al. (2019) proposed a framework for table information extraction in biomedical domains by defining rules for all possible variables. Specifically, for numerical variables, they retrieved metric-types by searching a set of possible tokens in their dictionary. Focusing on numerical tables, Nourbakhsh et al. (2020) extracted metric-types in earning reports by using similarity scores between stored metric-types and the corresponding non-numeric text for the leftmost cells. They investigated only header texts to identify metric-type from a limited set of tokens in a vocabulary. Dealing with out-of-vocabulary issues, we elaborate the caption information as additional inputs for our proposed framework.

The study that is the most similar to ours is that of Hou et al. (2019), who used tables from the experimental result section, combined with the title and abstract as document representations.

1 Dataset is available on https://github.com/titech-nlp/metrictable
to extract triples comprising tasks, datasets, and metrics for leaderboard construction. In our study, we represent tables more generically to preserve the original table structure in a multilevel header form that encompasses different structures in a real table. Therefore, our framework can be easily implemented in all types of tables. Milosevic et al. (2016) investigated multilevel tables to automatically detect table structures from XML tables.

Our pointer-generator-based model in the metric-type generation scheme is inspired by the promising results of the pointer-generator network (See et al. 2017) for the summarization task. The network deals with the out-of-vocabulary issue by jointly copying from source texts and generating from vocabularies.

Recent studies demonstrate that pretrained models can be successfully fine-tuned for downstream NLP tasks, thereby obviating the necessity to train a new model from scratch. Due to the effectiveness of transfer learning for a limited dataset, we propose fine-tuning pretrained models and utilizing universal language representations trained on a large corpus. A pretrained model that appropriately facilitates our framework for understanding the context of our table representation is BERT. We used the original BERT model (Devlin et al. 2019) trained on the BooksCorpus (800M words) and Wikipedia (2,500M words).

To improve contextualized representations in the scientific domain, Beltagy et al. (2019) introduced a domain-specific BERT model, i.e., SciBERT, which was trained on 1.14M papers from Semantic Scholar. Friedrich et al. (2020) used SciBERT on their models to solve the information extraction task in the same domain and achieved significant performances. Similarly, we fine-tuned SciBERT in our proposed BERT-based model as we used scientific papers as our dataset sources.

To encompass the prediction and generation tasks of our proposed framework, we fine-tuned a pretrained encoder-decoder, T5, which can be easily adapted to multitask settings. Raffel et al. (2020) introduced T5 as a unified framework that converts NLP problems into text-to-text tasks using the same loss function and decoding procedure. They demonstrated that their approach can be successfully applied to various tasks such as summarization, question-answering, and natural language inference.
3 Metric-Type Identification for Numerical Tables

3.1 Dataset

We automatically extracted tables from PDF files of scientific papers in the computational linguistics domain using PDFMiner\(^2\) and Tabula\(^3\) as extraction tools and used only numerical tables associated with experimental results using the keywords *evaluation, result, comparison,* and *performance.* We used papers from the ACL and EMNLP conferences (2016 to 2019) on the ACL Anthology\(^4\) website as data sources.

In tables presented in scientific papers, information regarding table semantics is rarely provided. Based on the manner by which information is “read” from a table, Hurst (2000) separated functional table areas into access cells and data cells. Access cells comprise column headers and/or row headers. We define the data structure based on their functional areas: table caption (*capt*), row headers (*rh*), column headers (*ch*), and cells. The headers in the row and column have several levels, and we assume that header names at the same level are of the same type. Figure 2 shows the structure of the table.

We asked several qualified workers in the computer science field to manually verify the extracted table structure to ensure the separation of row headers, column headers, and cells, as shown in Figure 3. Based on the table structure, a header-level was defined as the location order of a group of header names in the same column of row headers, or in the same row of column headers. Subsequently, the workers annotated the metric-types, which are a unit of measurement for numbers in the table, by prioritizing the location of the metric-type in a specific header-level. Detailed annotation instructions are provided in Figure 7 in the Appendix.

\[ \begin{array}{c}
  \text{ch}_j, \text{level 1} \\
  \text{ch}_j, \text{level 2} \\
  \vdots \\
  \text{ch}_j, \text{level v} \\
  \text{rh}_i, \text{level 1} \quad \ldots \quad \text{rh}_i, \text{level u} \\
  \text{cell}_i, j
\end{array} \]

Table X. Caption.

Fig. 2 Table structure.
The annotators successfully identified the metric-types of approximately 70% of the tables in their headers, and they determined the metric-types of the remainder based on information provided in the table captions. When no metric-type was provided in the headers, we assumed the metric-type was the same for all values in the table. The structure from the example shown in Figure 3 is `capt: “model comparison in task 1 and 2”; rh level 1: (models, models, models, models); rh level 2: (model a, model b, model c, model d); ch level 1: (task 1, task 1, task 2, task 2); ch level 2: (prec, rec, prec, rec); and metric-type: (prec, rec, prec, rec) (identified in ch level 2).

We double annotated 10% of our corpus and obtained near-perfect inter-rater agreement (0.813) using Krippendorf’s alpha (Krippendorff and Craggs 2016) in identifying whether metric-type was located in the row header, column header, or neither. A substantial agreement (0.762) was achieved in determining metric-type tokens using the caption information for not-in-header metric-type cases.

The statistics of our dataset are provided in Table 1.

![Table X: Model comparison in Task 1 and 2.](image)

| Models | Task 1 | Task 2 |
|--------|--------|--------|
|        | Prec   | Rec    | Prec   | Rec    |
| Model A| 60     | 60     | 60     | 60     |
| Model B| 70     | 70     | 70     | 70     |
| Model C| 80     | 80     | 80     | 80     |
| Model D| 90     | 90     | 90     | 90     |

Table 1 Dataset statistics.

| Metric Type       | Task 1 | Task 1 | Task 2 | Task 2 |
|-------------------|--------|--------|--------|--------|
| Prec              | 60     | 60     | 60     | 60     |
| Rec               | 70     | 70     | 70     | 70     |
| Prec              | 80     | 80     | 80     | 80     |
| Rec               | 90     | 90     | 90     | 90     |

Fig. 3 Illustration of table preprocessing.
3.2 Problem Definition

Let Table = (capt, rh_i^k, ch_i^l, cell_{ij}), where 1 \leq i \leq n_r, 1 \leq j \leq n_c, 1 \leq k \leq u, 1 \leq l \leq v, denote an n_r \times n_c table with the u levels of rh and v levels of ch. The task is to identify a tuple of metric-type tokens, denoted by \( \hat{m} \). When the tuple is in the k-th level of the row header, we extract the rh_k as \( \hat{m} \). When the tuple is in the l-th level of the column header, we extract ch_l as \( \hat{m} \). If both the row and column headers do not contain a tuple of metric-type tokens, we generate \( \hat{m} \) using information from the table caption (capt). The formulation for the metric-type identification is as follows:

\[
\hat{m} = \begin{cases} 
(rh_1^k, rh_2^k, ..., rh_n^k), k \in \{1, ..., u\} & \text{if the tuple is in the row header} \\
(ch_1^l, ch_2^l, ..., ch_n^l), l \in \{1, ..., v\} & \text{if the tuple is in the column header} \\
(w_m^1, w_m^2, ..., w_m^n), w_m \in W_m \text{ or } w_m \in \text{capt} & \text{otherwise},
\end{cases}
\]

where \( W_m \) is the set of metric-types in the vocabulary. If the table headers do not contain any metric-types, then the table is likely to exhibit the values of a single metric-type. In other words, when \( \hat{m} \) is not located in header rh or ch, we will find a metric-type token \( w_m \) from \( W_m \) or \( \text{capt} \), and copy \( w_m \) to all the columns\(^5\) from 1 to \( n_c \), resulting in \( w_1^m = w_2^m = ... = w_n^m = w_m \), where \( w_i^m \) is a metric-type token in position \( i \).

4 Models

We propose neural models to identify the metric-type for multilevel header tables using a joint model that enables metric-type location prediction and metric-type token generation.

4.1 Pointer-Generator Supervised Attention Model

We obtained the representations of captions and header-levels using a bidirectional long short-term memory (BiLSTM) encoder and then captured the header-level weights using supervised attention between the header-level encoder and the metric-type header-location outputs. In the generation scheme, we adopted the pointer-generator network to consider captions as source texts and metric-type vocabulary in the metric-type generation gate. The architecture of the proposed model is shown in Figure 4.

**Header encoder** We obtained the vector representation of each header-level by averaging the vectors of all header name tokens at the same level. Let \( E_{rh_k} \) denote the average vector of the

\(^5\) We chose columns instead of rows simply because majority metrics are mentioned in their column headers.
Fig. 4 Architecture of proposed pointer-generator-based model for identifying metric-types in tables.

Initial vector representations of the row header tokens at level $k$. Similarly, let $E_{ch_l}$ denote the average vector of the initial vector representations of the column header tokens at level $l$. We used BiLSTM as our encoder, which learns bidirectional long-term dependencies between time steps of sequence data. We input a sequence of row header vectors $E_{rh_1:u}$ to the BiLSTM encoder to obtain the representation of the $k$-th level $h_{rh_k}$, while considering both the entire history $E_{rh_1:k}$ and the entire future $E_{rh:k:u}$ as follows:

$$h_{rh_k} = BiLSTM(E_{rh_1:u}, k).$$  \hspace{1cm} (2)

Similar to the row headers, for a sequence of column header vectors $E_{ch_1:v}$, we used the BiLSTM encoder to obtain the representation of the $l$-th level $h_{ch_l}$, while considering both the entire history $E_{ch_1:l}$ and entire future $E_{ch_1:v}$ as follows:

$$h_{ch_l} = BiLSTM(E_{ch_1:v}, l).$$  \hspace{1cm} (3)

To obtain the contexts of a sequence of row header-level $C_{rh}$ and column header-level $C_{ch}$, we incorporated the dot attention mechanism proposed by Luong et al. (2015). We selected the hidden state of the last level and combined it with the weighted hidden states as follows:
Suadaa et al. Metric-Type Identification for Multilevel Header Numerical Tables

\[ C_{rh} = [h_{rh_u}; \sum_{k=1}^{u} a_{rh_k} h_{rh_k}] \]

\[ C_{ch} = [h_{ch_v}; \sum_{l=1}^{v} a_{ch_l} h_{ch_l}] \]

where attention \( a_{rh_k} \) and \( a_{ch_l} \) are derived by comparing the final encoder outputs \( h_{rh_u} \) and \( h_{ch_v} \) with each source hidden state \( E_{rh_k} \) and \( E_{ch_l} \), respectively, as follows:

\[ a_{rh_k} = \frac{\exp(h_{rh_u}^T E_{rh_k})}{\sum_{k'} \exp(h_{rh_u}^T E_{rh_{k'}})} \]

\[ a_{ch_l} = \frac{\exp(h_{ch_v}^T E_{ch_l})}{\sum_{l'} \exp(h_{ch_v}^T E_{ch_{l'}})} \]

Note that \([x; y]\) denotes the concatenation of vectors \( x \) and \( y \).

Caption encoder Let \( E_{\text{capt}_x} \) denote the initial vector representation of the caption token at position \( x \). Similar to the headers, we used the BiLSTM encoder with attention \( a_{\text{capt}_x} \) to compute the context vector of a caption sequence \( C_{\text{capt}} \) with length \( t \), while considering both the entire history \( E_{\text{capt}_1:x} \) and entire future \( E_{\text{capt}_{x+1:t}} \) as follows:

\[ h_{\text{capt}_x} = \text{BiLSTM}(E_{\text{capt}_{1:x}}, x), \]

\[ a_{\text{capt}_x} = \frac{\exp(h_{\text{capt}_x}^T E_{\text{capt}_x})}{\sum_{t'} \exp(h_{\text{capt}_t}^T E_{\text{capt}_{t'}})} \]

\[ C_{\text{capt}} = [h_{\text{capt}_t}; \sum_{x=1}^{t} a_{\text{capt}_x} h_{\text{capt}_x}] \]

Metric-type header-location gates We input the concatenation of the row and column header contexts to the softmax layer with linear transformation to obtain the metric-type header-location probability, as follows:

\[ p_{hloc} = \text{softmax}(W([C_{rh}; C_{ch}] + b)), \]

which includes the probabilities of the metric-types located in the row headers \( p_{rh} \) and column headers \( p_{ch} \), or not located in the headers \( p_{\text{capt}} \), where \( p_{rh} + p_{ch} + p_{\text{capt}} = 1 \).

Metric-type header-level gates Since the attention scores \( a_{rh_k} \) and \( a_{ch_l} \) capture the relevant header-level information in rows and columns, these attention scores are used as header-level weights as follows:

\[ w_{hlvl_i} = \begin{cases} a_{rh_i} p_{rh} & \text{if } i \leq u \\ a_{ch_{i-u}} p_{ch} & \text{if } i > u \end{cases} \]
where $i \in \{1, \ldots, u, (u + 1), \ldots, (u + v)\}$ is a header-level index.

**Metric-type generation gates** In our pointer-generator network, we used the sigmoid layer with linear transformation to obtain a switch copy probability as follows:

$$p_{\text{copy}} = \text{sigmoid}(W(C_{\text{capt}}) + b),$$  \hfill (13)

which allows us to select between copying a word $w_{\text{capt}}$ from a table caption and generating a word $w_m$ from the metric-type vocabulary, where $p_{\text{copy}} \in [0, 1]$. We used a softmax function with linear transformation to compute the probability distribution over the metric-type vocabulary:

$$P_{\text{vocab}}(w_m) = \text{softmax}(W(C_{\text{capt}}) + b).$$  \hfill (14)

Subsequently, we obtained the probability distribution over the extended vocabulary, as follows:

$$P(w_m) = p_{\text{copy}} \sum_{i : w_i = (w_m)} a_{\text{capt}i} + (1 - p_{\text{copy}})P_{\text{vocab}}(w_m),$$  \hfill (15)

where $i$ is the index of metric-type tokens in the vocabulary.

**Learning objective** For training, we used the negative log-likelihood objective as the loss function. In addition, we adopted supervised attention (Liu et al. 2016) to jointly supervise the row and column header-level attention to obtain the metric-type header-level. We combined all loss functions in the location classification and token generation model, and defined $\alpha$ as the weight, as follows:

$$\mathcal{L} = -(1 - \alpha)(\sum_c z_{hloc_c} \log p_{hloc_c} + \sum_{i=1}^{u+v} \log w_{hvl,i}) + \alpha(\log p_{\text{copy}} + \log P_{\text{vocab}}(w_m)), \hfill (16)$$

where $c \in \{\text{capt}, rh, ch\}$ is the metric-type header-location class and $z_{hloc_c}$ is the binary indicator (0 or 1) of each corresponding class.

### 4.2 Fine-tuning BERT-based Model

**Input representation** The input text in a fine-tuned BERT-based model is preprocessed by inserting two special tokens, i.e., [CLS] and [SEP]. In the original BERT architecture, [CLS] is appended to the beginning of the input as the representation of the entire input sequence, whereas [SEP] is inserted after each input type as a sign for a segment boundary. For example, in question-answering tasks with two types of input text, i.e., pairs of questions and answers, a [CLS] token is appended before the question tokens, and [SEP] tokens are placed after question and answer tokens to separate the question and answer segments. Based on the study by Liu
and Lapata (2019), we customized these preprocessing schemes by inserting [CLS] before each segment and inserting [SEP] after each segment. We partitioned our inputs into several segments: caption, row header level 1 to $u$, and column header level 1 to $v$.

After preprocessing, the input text is denoted as a sequence of tokens $X = (x_1, x_2, \cdots, x_n)$. Three types of embedding are assigned to each $x_i$: token embeddings representing the meaning of each token, segmentation embeddings indicating the segment boundaries of a sequence of tokens, and position embeddings indicating the token position within the sequences. Because BERT only includes two segments in its input, we named the odd segment as segment A and the even segment as segment B. The sum of these three embeddings was input to the bidirectional transformer layer of the BERT.

We used the token representations from the top hidden layers of the pretrained transformer as context embeddings. We assumed that the context vectors of each CLS token can represent the segment sequences more effectively. As shown in Figure 5, we labeled the input embedding as $E$, the final hidden vector of the [CLS] token for the $i^{th}$ input segment as $C_i \in \mathbb{R}_H$, and the final hidden vector for the $j^{th}$ input token as $T_j \in \mathbb{R}_H$.

We used a metric-type header-location gate and a metric-type header-level gate for metric-type location classification, and a metric-type generation gate to generate metric-type tokens from vocabulary encompassing out-of-header metric-types. The proposed BERT-based model

![Architecture of proposed BERT-based model for identifying metric-types in tables.](image)
architecture is shown in Figure 5.

**Metric-type header-location gates** We input the first segment context $C_1$ to the softmax layer via linear transformation to obtain the metric-type header-location probability, as follows:

$$p_{hlloc} = \text{softmax}(W(C_1) + b).$$

(17)

**Metric-type header-level gates** In our task, segments were used to represent the table section that is related the most closely with the metric-type. We incorporated the segment context $C_i$ into the sigmoid layer via linear transformation to obtain the probability that the metric-type is located at a specific header-level, as follows:

$$p_{hlvl_i} = \text{sigmoid}(W(C_i) + b).$$

(18)

Subsequently, the probabilities were normalized to all segments as the weight score of the header level, as follows:

$$w_{hlvl_i} = \frac{p_{hlvl_i}}{\sum_i p_{hlvl_i}}.$$  

(19)

**Metric-type generation gates** We used a softmax function with linear transformation based on the first segment context $C_1$ to compute the probability distribution over the metric vocabulary, as follows:

$$P_{vocab}(w_m) = \text{softmax}(W(C_1) + b).$$

(20)

**Learning objective** We combined all loss functions in the metric-type header-location, metric-type header-level, and metric-type generation gates as follows:

$$\mathcal{L} = -((1 - \alpha)(\sum_c z_{hlloc_c} \log p_{hlloc_c} + \sum_{i=1}^n \log w_{hlvl_i}) + \alpha \log P_{vocab}(w_m)),$$

(21)

where $\alpha$ is the weight of the metric-type generation functions.

4.3 Fine-tuning T5-based Model

The input text in the fine-tuned T5-based model was preprocessed by inserting several specific tokens to discriminate tokens in different locations. Tokens [CAPT], [ROW], and [COL] were placed before the caption, row-name, and column-name tokens, respectively. [ROW$i$] and [COL$j$] were inserted before the row-name tokens at the $i$ level and column-name tokens at the $j$ level, respectively; each name was separated by the [SEP] token. We fine-tuned the T5 model to perform multiple tasks, metric-type header-location classification, and metric-type token generation.
Fig. 6 Architecture of proposed T5-based model for identifying metric-types in tables.

Each task, we appended a prefix string to the input of the model, i.e., “identify location:” for the first task and “identify metric:” for the second task. The architecture of the proposed T5-based model is shown in Figure 6.

5 Experimental Settings

5.1 Baseline Model

Because our task pertain primarily to location classification and token prediction from metric-type vocabularies, we selected SVM as a baseline owing to its effectiveness in high-dimensional spaces, particularly in text classification. We used two SVM classification models as baselines: a metric-type location prediction model and a metric-type token prediction model from the vocabulary of metric-types. We used “tf.idf” from the concatenation header name tokens for all levels as an input representation in the first model and “tf.idf” of the caption tokens in the second one. We performed a grid search to tune the hyperparameters of the SVM model on the development set for search space \( c \in \{0.1, 1, 10, 100, 1000\} \) and \( gamma \in \{0.001, 0.0001\} \); subsequently, we selected the \( c \) and \( gamma \) parameters of the SVM that indicated the best accuracy.

5.2 Evaluation Metrics

The “accuracy” metric was used to evaluate the metric-type location and generated metric-type tokens.

**Metric-type location accuracy** The target of the metric-type location prediction model is the metric-type located in the row headers, in the column headers, or neither. The accuracy of the header-location \( (acc_{hloc}) \) is the rate of correct header-location predictions.

Details regarding the metric-type location at the header-level are required to identify metric-type token lists. We computed the accuracy of the metric-type header-level \( (acc_{hlvl}) \) using the ratio of correct header-level predictions to the total number of predictions.
**Metric-type token accuracy** Let \( \hat{m} = (\hat{w}_1, \ldots, \hat{w}_n) \) denote the sequence of predicted metric-type tokens for \( n_r \) rows or \( n_c \) columns (depending on the header-location prediction), and \( m = (w_1, \ldots, w_n) \) denote the target ones, e.g., \( \hat{m} = (f_1, f_1, f_1) \) and \( m = (f-1, f-1, f-1) \). We calculate the metric-type token accuracy using the string matching of all token lists in \( \hat{m} \) and \( m \), i.e.,

\[
acc_{sm}^m = \frac{\text{# correct } \hat{m}}{\text{# } \hat{m}},
\]

and string matching of each token pair \( \hat{w}_m \) and \( w_m \) in the token lists, i.e.,

\[
acc_{sm, \text{token}}^m = \frac{\text{# correct } \hat{w}_m}{\text{# } w_m}.
\]

To account for token predictions involving an abbreviation, we compute the metric-type token accuracy based on ordered character matching, as follows:

\[
acc_{ocm}^m = \frac{d}{\text{# } \hat{w}_m},
\]

where \( d \) is the number of \( \hat{w}_m \), whose characters appear in \( w_m \) in the same order. For example, the predicted token \( RG1 \) is regarded as correct when the reference token is \( ROUGE-1 \).

### 5.3 Implementation Details

We implemented our models using the AllenNLP library (Gardner et al. 2018). In our pointer-generator-based model, we used pretrained word embeddings for initialization and two-layer BiLSTMs with 256 hidden sizes in both the caption and header-level encoders. We added a dropout (Srivastava and Hovy 2014) with a probability of \( p = 0.1 \) to our header and caption encoder. We evaluated our models using \( k \)-fold cross-validation. Because we had imbalanced classes of metric-type location with row-header metric-type instances below 2%, we used \( k = 5 \) to increase the probability of minority class instances being evaluated.

To perform optimization in the training phase, we used Adam as the optimizer with a batch size of 10 and learning rates of \( 3 \times 10^{-3} \) and \( 3 \times 10^{-5} \) in the pointer-generator-based and BERT-based models, respectively, with a slanted triangular schedule (Howard and Ruder 2018). We trained the model for a maximum of 20 epochs and implemented early stopping on the validation set (patience of 10), and we set \( \alpha \) to 0.5. We used the original BERT and the domain-specific SciBERT uncased model to fine-tune our BERT-based model. For our T5-based model, we fine-tuned the model using Adafactor optimizer with a constant learning rate of 0.001 (Raffel et al. 2020).
6 Results

6.1 Experimental Results

Model comparison The performances of the proposed and baseline models are shown in Table 2. As shown, the pointer-generator supervised attention model initialized by Glove embeddings outperformed the baseline (SVM) in predicting the metric-type location. The accuracy of this model for metric-type generation was better than that of the baseline. A slight difference in the pointer-generator supervised-attention performance of the proposed model as compared with the SVM implies that the deep neural network architecture afforded minimal improvement in our limited dataset. Adding more datasets is suggested when training a deep learning model from scratch. Furthermore, we demonstrated that using only pretrained embeddings of a large pretrained model decreased the accuracy. The performance of our pointer-generator-based model deteriorated significantly when the input was initialized using BERT and SciBERT.

We fine-tuned the large pretrained BERT, SciBERT, and T5 to exploit deep neural models trained on a larger corpus. For our fine-tuned BERT-based models, we used the context representation of their transformer encoders and added gates to solve our tasks. The accuracy of our fine-tuned BERT-based models was significantly better than that of pointer-generator-based models trained on our limited corpus, where header-location and header-level prediction accuracies exceeding 88% and generation accuracy improvement exceeding 3 points (%) were achieved. The fine-tuned BERT-based model using the domain-specific SciBERT led to significant improvements in all metrics because its corpus is similar to ours.

The performance of our fine-tuned T5 with an encoder-decoder architecture is comparable to that of the fine-tuned encoder-only BERT model. The unified framework of T5 was easily

| Model                                | acc\textsubscript{hloc} | acc\textsubscript{hlev} | acc\textsubscript{m} | acc\textsubscript{m,token} | acc\textsubscript{ocm,token} |
|--------------------------------------|--------------------------|--------------------------|-----------------------|-----------------------------|-----------------------------|
| SVM                                  | 85.04                    | 82.37                    | 73.48                 | 70.22                       | 70.22                       |
| Pointer-Generator Supervised-Att (Glove) | 85.78                    | 84.89                    | 74.22                 | 73.06                       | 75.76                       |
| Pointer-Generator Supervised-Att (BERT) | 65.78                    | 53.78                    | 37.33                 | 25.29                       | 27.38                       |
| Pointer-Generator Supervised-Att (SciBERT) | 65.63                    | 52.74                    | 34.37                 | 23.10                       | 25.20                       |
| Fine-tuned BERT                      | 89.04                    | 88.15                    | 78.07                 | 76.91                       | 78.19                       |
| Fine-tuned SciBERT                   | 91.85                    | 91.56                    | 80.74                 | 79.88                       | 81.75                       |
| Fine-tuned T5                        | 89.63                    | 85.22                    | 77.78                 | 76.03                       | 76.91                       |

Table 2 Accuracy scores (%) of different metric-type identification models using five-fold cross-validation. Scores differed significantly those of fine-tuned SciBERT with paired-bootstrap-resampling test \((p < 0.05)\). **Bold** indicates the best score.
adapted to our tasks by inserting different prefixes in the input. Unlike fine-tuned BERT-based models, no added layer was used in the fine-tuning of T5.

**Effect of copy mechanism**  We evaluated our pointer-generator-based model using an ablation test, as presented in Table 3. As shown, the performance of our generation model without a copy mechanism decreased. This shows that incorporating the copy mechanism is beneficial to metric-type token generation. Our model demonstrated the worst accuracy when it was executed without a pointer-generator network because the location prediction model alone failed to manage out-of-header metric-types.

**Effect of segment embeddings**  Table 4 shows the effect of segment embeddings in our BERT-based model. The accuracies of the fine-tuned BERT and the SciBERT models without segment embeddings both decreased. This implies that segment embeddings successfully discriminated the header-level boundaries in the input representation of the BERT-based models.

### 6.2 Qualitative Analysis

As introduced in the model formulation (Eq. 1), our models include row-header, column-header, and not-in-header metrics based on metric-type location. We evaluated the performances of our models for each location category based on the precision ($P$), recall ($R$), and F1-score ($F$).
Suadaa et al.  

Metric-Type Identification for Multilevel Header Numerical Tables

| Model        | Row-header |  |  | Col-header |  |  |  | Not-in-header |  |  |  |
|--------------|------------|---|---|------------|---|---|---|---------------|---|---|---|
|              | \(P\) | \(R\) | \(F\) | \(P\) | \(R\) | \(F\) | \(P\) | \(R\) | \(F\) | \(P\) | \(R\) | \(F\) |
| SVM          | 6.67 | 6.67 | 6.67 | 87.28 | 87.06 | 86.41 | 76.23 | 76.98 | 76.04 |   |
| PGen (Glo.)  | 36.67 | 31.67 | 33.33 | 90.44 | 90.49 | 90.40 | 53.90 | 71.86 | 60.99 | 91.82 | 70.91 | 79.43 |
| F-BERT       | 50.00 | 46.67 | 48.00 | 92.00 | 92.07 | 92.00 | 80.20 | 81.68 | 80.74 |
| F-SciBERT    | 50.00 | 53.33 | 51.43 | 96.18 | 92.71 | 94.36 | 84.16 | 91.85 | 87.58 |
| F-T5         | 10.00 | 13.34 | 11.43 | 92.47 | 92.25 | 92.04 | 81.68 | 85.87 | 82.40 |

Table 5  Performance scores (%) of metric-type location prediction for each header-location class using five-fold cross-validation. **Bold** indicates the best score.

As shown in Table 5, F-SciBERT demonstrated the best performance for all metric-type location cases. Due to the small number of instances in the dataset, the overall performance of the row-header metric-type category was worse than that of the others. Applying a specific procedure to handle the imbalanced class problems is left for future work.

The pointer-generator-based and fine-tuned BERT-based model outperformed the baseline in predicting row-header and column-header metric-type cases. Specifically, a significant margin was observed in predicting the row-header metric-type as the minority class. This indicates that combining the metric-type header-location and the header-level gate improved the model’s ability to determine the metric-type location in all cases. The text-to-text mechanism of the fine-tuned T5 failed to perform the metric-type location prediction of the minority cases.

For the not-in-header metric-type cases, the models successfully generated metric-type tokens from the vocabulary. Because we added the copying ability to our pointer-generated-based models, we present both the generating and copying performances in Table 5. As shown, our proposed model performed better than the generating model in terms of copying.

Additionally, we investigated the errors in the predicted metric-type tokens. We discovered that the models generated more generic metric-types; for example, they extracted score as a prediction of the target accuracy. By contrast, our models generated terms similar to the ground truth metric, such as the metric-type pearson’s for the target \(r\). Examples of table captions, headers, and metric-type for each metric-type location are shown in Table 6.

7 Conclusion

In this study, we extracted multilevel header numerical table datasets comprising header ta-
Table 6 Example of table caption, headers, and predicted metric-type.
Acknowledgement

This paper is a revised version of a previous paper by Suadaa et al. (2021), which was published in the proceedings of the 16th European Chapter of the Association for Computational Linguistics (EACL-2021). This study was partially supported by JST PRESTO (Grant Number JPMJPR1655). We thank the anonymous reviewers for their discussions pertaining to this study and their comments on previous drafts of the paper.

References

Beltagy, I., Lo, K., and Cohan, A. (2019). “SciBERT: A Pretrained Language Model for Scientific Text.” In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Friedrich, A., Adel, H., Tomazic, F., Hingerl, J., Benteau, R., Marusczyk, A., and Lange, L. (2020). “The SOFC-Exp Corpus and Neural Approaches to Information Extraction in the Materials Science Domain.” In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 1255–1268, Online. Association for Computational Linguistics.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. (2018). “AllenNLP: A Deep Semantic Natural Language Processing Platform.” In Proceedings of Workshop for NLP Open Source Software (NLP-OSS), pp. 1–6, Melbourne, Australia. Association for Computational Linguistics.

Hou, Y., Jochim, C., Gleize, M., Bonin, F., and Ganguly, D. (2019). “Identification of Tasks, Datasets, Evaluation Metrics, and Numeric Scores for Scientific Leaderboards Construction.” In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 5203–5213, Florence, Italy. Association for Computational Linguistics.

Howard, J. and Ruder, S. (2018). “Universal Language Model Fine-tuning for Text Classifi-
cation.” In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 328–339, Melbourne, Australia. Association for Computational Linguistics.

Hurst, M. P. (2000). The Interpretation of Tables in Texts. Ph.D. thesis, The University of Edinburgh.

Krippendorff, K. and Craggs, R. (2016). “The Reliability of Multi-Valued Coding of Data.” Communication Methods and Measures, 10(4), pp. 181–198.

Liu, L., Utiyama, M., Finch, A., and Sumita, E. (2016). “Neural Machine Translation with Supervised Attention.” In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 3093–3102, Osaka, Japan. The COLING 2016 Organizing Committee.

Liu, Y. and Lapata, M. (2019). “Text Summarization with Pretrained Encoders.” In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Luong, T., Pham, H., and Manning, C. D. (2015). “Effective Approaches to Attention-based Neural Machine Translation.” In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Milosevic, N., Gregson, C., Hernandez, R., and Nenadic, G. (2016). “Disentangling the Structure of Tables in Scientific Literature.” In Métais, E., Meziane, F., Saraee, M., Sugumaran, V., and Vadera, S. (Eds.), Natural Language Processing and Information Systems, pp. 162–174, Cham. Springer International Publishing.

Milosevic, N., Gregson, C., Hernandez, R., and Nenadic, G. (2019). “A Framework for Information Extraction from Tables in Biomedical Literature.” International Journal on Document Analysis and Recognition (IJDAR), 22(1), pp. 55–78.

Nourbakhsh, A., Ghassemi, M. M., and Pomerville, S. (2020). “SPread: Automated Financial Metric Extraction and Spreading Tool from Earnings Reports.” In Caverlee, J., Hu, X. B., Lalmas, M., and Wang, W. (Eds.), WSDM ’20: The 13th ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3–7, 2020, pp. 853–856. ACM.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.” Journal of Machine Learning Research, 21(140), pp. 1–67.

See, A., Liu, P. J., and Manning, C. D. (2017). “Get To The Point: Summarization with Pointer-
Suadaa et al. Metric-Type Identification for Multilevel Header Numerical Tables

Generator Networks.” *CoRR*, abs/1704.04368.

Srivastava, S. and Hovy, E. (2014). “Vector Space Semantics with Frequency-driven Motifs.” In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 634–643, Baltimore, Maryland. Association for Computational Linguistics.

Suadaa, L. H., Kamigaito, H., Okumura, M., and Takamura, H. (2021). “Metric-Type Identification for Multi-Level Header Numerical Tables in Scientific Papers.” In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 3062–3071, Online. Association for Computational Linguistics.
Appendix

A Annotation Guideline

Figure 7 shows annotation instructions for metric-type identification task.

### Instructions

We will show you a numerical table extracted from a scientific paper. Your job is to identify metric-type of numbers in the table, whether the metric-type location is in row header (case 1) or column header (case 2). Fill the metric-type token with a set of header names representing the metric-type, separated by a comma. For example:

**Case 1: metric-type in row header**

Table 1. Automatic semantic evaluation (higher is better for all but SER).

|        | BASE  | +ADJ  | +SENT | +STYLE |
|--------|-------|-------|-------|--------|
| BLEU   | 0.126 | 0.164 | 0.166 | 0.173  |
| METEOR | 0.206 | 0.233 | 0.234 | 0.235  |
| CIDEr  | 1.300 | 1.686 | 1.692 | 1.838  |
| NIST   | 3.840 | 4.547 | 4.477 | 5.537  |
| Avg SER| 0.053 | 0.063 | 0.064 | 0.090  |

**Metric-type location:** row header  column header  not in header

**Metric-type token:** BLEU, METEOR, CIDEr, NIST, Avg SER

**Case 2: metric-type in column header**

Table 2. Experimental results in exploring the shared syntactic order event detector.

|        | Pre. | Rec. | F1  |
|--------|------|------|-----|
| Model  | CL_Trans_MLP | 20.3 | 16.3 | 18.1 |
| Model  | CL_Trans_CNN  | 32.5 | 16.3 | 21.7 |
| Model  | CL_Trans_Hybrid | 30.4 | 17.6 | 22.3 |

**Metric-type location:** row header  column header  not in header

**Metric-type token:** Pre., Rec., F1

If you cannot identify the metric-type in the header name (case 3), please fill the metric-type token by deciding a single metric-type after reading carefully the description mentioning table number in the paper source, including table caption. For example:

**Case 3: metric-type not in header**

Table 3. Comparison with Transformer on French-English translation task. The evaluation metric is case insensitive BLEU score.

|        | Dev  | Test |
|--------|------|------|
| Method | Transformer | 29.42 | 35.15 |
| Method | This work   | 30.40 | 36.04 |

**Description mentioning Table 3 in the paper**

Table 3 shows that our model also outperforms Transformer by 0.89 BLEU points on French-English translation task.

**Metric-type location:** row header  column header  not in header

**Metric-type token:** BLEU

---

Fig. 7 Metric-type annotation instructions.
Lya Hulliyatus Suadaa: is a Ph.D. student at the Department of Information and Communications Engineering, School of Engineering, the Tokyo Institute of Technology. She is supported by the Indonesian Endowment Fund for Education (LPDP). Her current research interests include natural language generation and information extraction.

Hidetaka Kamigaito: received his Ph.D. from the Tokyo Institute of Technology, and is currently an assistant professor at the Institute of Innovative Research, the Tokyo Institute of Technology. His current research interests include natural language processing with emphasis on document-level text processing.

Manabu Okumura: received his Ph.D. from the Tokyo Institute of Technology in 1989. He was an assistant at the Department of Computer Science, the Tokyo Institute of Technology, from 1989 to 1992; and an associate professor at the School of Information Science, the Japan Advanced Institute of Science and Technology, from 1992 to 2000. He is currently a professor at the Institute of Innovative Research, the Tokyo Institute of Technology. His current research interests include natural language processing, particularly text summarization, computer-assisted language learning, sentiment analysis, and text data mining.

Hiroya Takamura: received his Ph.D. from the Nara Institute of Science and Technology. He worked as a professor at the Tokyo Institute of Technology and is currently a research team leader at the AI Research Center of Advanced Industrial Science and Technology. His current research interests include natural language processing.

(Received July 31, 2021)
(Accepted September 6, 2021)