NTCIR 17 Conference: Proceedings of the 17th NTCIR Conference on Evaluation of Information Access Technologies, December 12-15, 2023, Tokyo, Japan
DOI: https://doi.org/10.20736/0002001308

KSU at the NTCIR-17 UFO Task

Tomokazu Hayashi
i2386111@cc.kyoto-su.ac.jp
Kyoto Sangyo University
Japan

Hisashi Miyamori
miya@cc.kyoto-su.ac.jp
Kyoto Sangyo University
Japan

ABSTRACT

This paper describes the methods and results of Team KSU for the UFO task at NTCIR-17. In the TDE subtask, we designed methods for cell type classification using exhaustive tree structures based on the spanning sizes of the merged cells in the table. In the TTRE subtask, we designed methods for cell retrieval based on the cell class. Scores on the F-measure in the formal run were 95.37% for ID81 in TDE and 9.18%, 4.08%, and 6.63% for ID99 on the Name, Value, and Total, respectively, in TTRE. Scores on the F-measure including the formal run and late submission run were 95.37% for ID81 in TDE and 32.21%, 27.19%, and 29.70% for ID127 on the Name, Value, and Total, respectively, in TTRE.

KEYWORDS

Statistical data, Tabular data, Open data

TEAM NAME

KSU

SUBTASKS

TDE subtask
TTRE subtask

1 INTRODUCTION

Tabular data is a data format commonly used in various documents along with text, and is an important and indispensable element for many applications. However, tabular data is described in various structures, and understanding its content and mapping it to text is still not easy. The TDE subtask aims to classify each cell of a table in a given annual securities report into four classes, while the TTRE subtask aims to select the corresponding cells in the table associated with a given text in the securities report.

Figure 1 is an example of a table included in a securities report. As a characteristic of tables included in securities reports, there are many cases in which monetary units are listed in the upper right corner of the table, or the sum of monetary amounts, etc., are listed in a merged cell at the bottom of the table.

This paper describes the methods and results of Team KSU for the UFO task [3] in NTCIR-17. We focused on the fact that tables in securities reports have a complex structure created by merged cells and indentation in the cells and developed a method to incorporate this structure into the representation of tables in several ways.

Scores on the F-measure in the formal run were 95.37% for ID81 in TDE and 9.18%, 4.08%, and 6.63% for ID99 on the Name, Value, and Total, respectively, in TTRE. Scores on the F-measure including the formal run and late submission run were 95.37% for ID81 in TDE and 32.21%, 27.19%, and 29.70% for ID127 on the Name, Value, and Total, respectively, in TTRE.

(Unit : One million yen) (単位：百万円)

ヘッジ会計の方法 Hedge accounting method	取引の種類 Type of transaction	主なヘッジ対象 Main hedged subject	契約額等 Contract amount and so on	うち１年超 In over one years	時価 Current market price
原則的処理方法 Processing method in general	商品スワップ取引 Swap transactions of Merchandise 受取変動・支払固定 Receive-floating ・ Pay-fixed 原油 Crude oil	営業未払金 Non-operating trade payables	69,132	27,452	△24,304
	商品オプション取引 Option transactions of Merchandise 売建 Going short プット Put Option 原油 Crude oil	営業未払金 Non-operating trade payables	33,120	15,468	△7,229
	買建 Going long コール Call Option 原油 Crude oil	営業未払金 Non-operating trade payables	42,798	20,103	△1,717
合計 Sum			145,051	63,025	△33,250

Figure 1: Examples of tables included in the Annual Securities Report

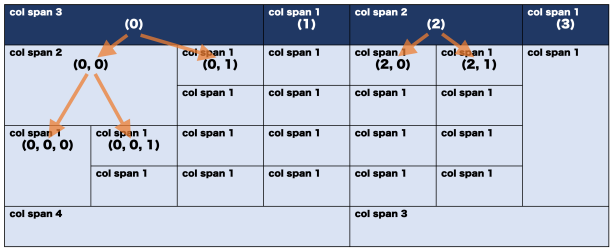

Figure 2: default vertical tree

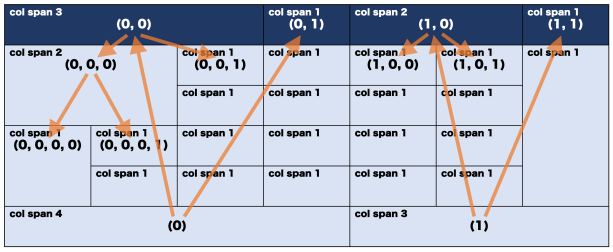

Figure 3: exhaustive vertical tree

2 DATA COLLECTION AND BASIC ANALYSIS

In order to clarify the guidelines for the construction of the system to be implemented, we first analyzed the data collection parameters of the data set of securities reports in HTML format provided by the UFO task.

2.1 TDE

Table 1 shows the distribution of datasets in the TDE subtask. The dataset used in the TDE subtask consists of 442 securities reports in HTML format, divided into 190 and 252 test and train data,

Table 1: Number of datasets in TDE subtask

	document	table	cell
test	190	1,660	45,499
train	252	2,530	66,369
total	442	4,190	111,868

respectively. There were 1,660 and 2,530 tables in the test and train data, respectively, and 45,499 and 66,369 cells, respectively.

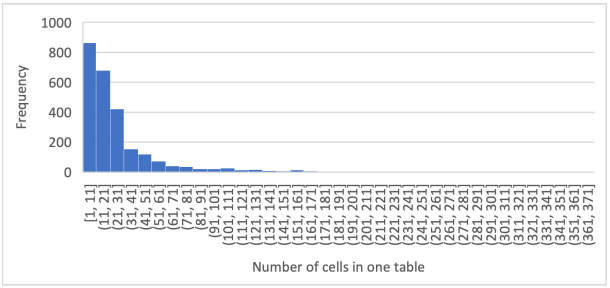

Figure 4: Histogram of cells contained in the table

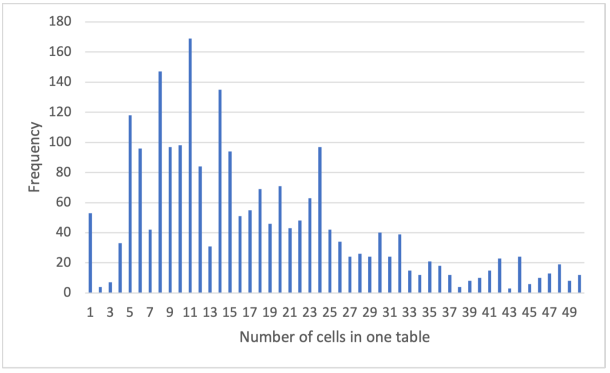

Figure 5: Bar chart of cells (less than 50 only) included in the table

Figure 4 shows a histogram of the number of cells in a table. The figure shows that the number of cells in a table is mostly less than 51. Figure 5 shows the frequency of tables with less than 50 cells. From the figure, it is found that the table consists of 11 cells most frequently.

2.2 TTRE

Table 2 shows the distribution of datasets in the TTRE subtask. The dataset used in the TTRE subtask consists of 67 securities reports in HTML format, which are divided into 25 and 42 datasets for test and train data, respectively. There were 1,125 and 1,726 tables in the test and train data, respectively, and 47,517 and 80,644 cells, respectively.

Table 3 shows the top 10 phrases with no retrieved cell in the TTRE subtask train data. It is found that there are many cases in

Table 2: Number of datasets in TTRE subtask

	document	table	cell
test	25	1,125	47,517
train	42	1,726	80,644
total	67	2,851	128,161

Table 3: Phrases for which there is no retrieved cell

phrase	frequency	phrase	frequency
(1)	50	※ 1	18
2	34	(注)	18
(2)	34	1	18
3	25	(3)	18
4	18	※ 2	15

"(注)" in the table represents a phrase for remarks in Japanese.

which no search cell corresponds to Name and Value when only numeric or symbolic phrases are entered. The number of phrases in the train data of the TTRE subtask was 3,402.

3 METHODS

TUTA [9] achieves state-of-the-art results on five datasets by using a tree-based structure called a bi-dimensional coordinate tree to represent the hierarchical information contained in the table, and by using embedding in conjunction with row and column indexes. TUTA assumes that when there is a hierarchical structure in a table, the size of the merged cells decreases gradually from the top (or leftmost) to the bottom (or rightmost) of the table. However, with this method, as shown in Figure 2, coordinates can only be assigned to cells in accordance with this assumption, and coordinates cannot be assigned to large joint cells that appear in the middle of the table. Therefore, this paper proposes a method to construct a bi-dimensional coordinate tree in descending order of the size of the joint cells in the table, obtained by exhaustively checking the sizes of the merged cells in a table.

In the following, the bi-dimensional coordinate tree determined by TUTA is called the “default tree”, of which the vertical (column) and horizontal (row) trees are called the “default vertical tree” and “default horizontal tree”, respectively, and the bi-dimensional coordinate tree determined by the proposed method is called the “exhaustive tree”, of which the vertical (column) and horizontal (row) trees are called “exhaustive vertical tree” and “exhaustive horizontal tree”, respectively.

3.1 exhaustive tree

The proposed method is explained using Figure 3. First, for the exhaustive vertical tree, the parent node is set to the column containing the cell with the largest col span. However, if the cells have the same size, the cell at the top is adapted. In the example table, the two cells marked (0) and (1) correspond. Next, for each cell corresponding to the parent node, the column containing the cell

with the next largest column span is made a child node of the parent node. However, if the cells have the same size, the cell at the top is adapted. In the example table, the parent node (0) has a col span of 4, and the cell with the next largest col span among the four columns on the left side of the table is the cell labeled (0, 0), so the cell in that column that encompasses the column span is made a child node of the parent node. The cell with the range of columns is made a child node of the parent node. Here, the two cells marked (0, 0) and (0, 1) are applicable. The same procedure is repeated recursively, ending when the size of the column span (col span) reaches 1. Similarly, for the exhaustive horizontal tree, the tree structure is constructed based on the size of the row span.

3.2 Data conversion

The securities reports in HTML format were converted to a format suitable for input into the TUTA model.

3.2.1 *Table extraction.*

The ranges enclosed by <table> tags were extracted from the HTML document. However, tables without a single cell-id assigned by the annotator were excluded.

3.2.2 *Table normalization.*

HTML tables were normalized using the NFKC (Normalization Form Compatibility Composition) method. All △ symbols were replaced with - symbols.

3.2.3 *Convert to json format.*

The table in HTML format was converted to json format suitable for input into the TUTA model. The cell features used are shown in Table 4. Examples of cells in HTML format are shown in Figure 6. V represents the cell value, "812.6" in the example. DT represents the data type of the cell value: 0 for text, 1 for numeric, and 5 for blank. Numeric values are determined based on whether the majority of the cell values are numeric. HF indicates whether the cell value contains a mathematical expression, and is set to 0. LB, TB, BB, and RB were set to 1 if there were ruled lines on the left, top, bottom, and right parts, respectively, and 0 otherwise. BC is the background color, and FC is the font color. FB indicates whether the font is bold or not, and is set to 1 if the font is bold and 0 otherwise. The coordinates of each cell, the coordinates of the joined cells, and the tree structure of the table were also used as features of the table.

Table 4: Feature set of the cell used

Feature name	Description	e.g.
V	cell value	812.6
DT	data type	1
HF	if has formula	0
LB	if has left border	1
TB	if has top border	1
BB	if has bottom border	1
RB	if has right border	1
BC	background color	#ffffff
FC	font color	#000000
FB	if has font bold	0

```
<td data-ufo-tde-cell-id="S100LO2J-0102010-tab2-r5c6"
    data-ufo-tde-cell-type="data"
    style="border-left: 1px solid #000000;
        border-top: 1px solid #000000;
        border-right: 1px solid #000000;
        border-bottom: 1px solid #000000;
        vertical-align: middle">
    <p style="margin-right: 42px; text-align: right">
        <span style="font-size: 12px">812.6</span>
    </p>
</td>
```

Figure 6: Example of cell in HTML format

3.2.4 *Translate into English.*

The Google Translate API was used to translate the cell values. In some cases, the Japanese currency unit "yen" was mistranslated as "circle," so all "circle" was replaced with "yen".

3.3 TDE

The pre-trained TUTA-implicit model was fine-tuned to be classified into four classes: Metadata, Header, Attribute, and Data. There are three types of TUTA models: TUTA-base model, TUTA-implicit model, and TUTA-explicit model. The TUTA-implicit model is a model that uses position embedding during training. The TUTA-implicit model was used in this study because previous studies have shown that the TUTA-implicit model has the best performance. The architecture of the model was used without modification. The batch size was set to 2 or 4, and the learning rate was set to 8e-6, up to 200 epochs.

3.4 TTRE

We propose a cell retrieval method considering the cell class. Figure 7 shows an overview of the proposed method in the TTRE subtask. First, the text of a given phrase and the text of each cell of a table in the same document are input to Text Encoder to obtain their respective embedded representations. Then, the similarity between the phrase and the cell text is calculated, and the Name is determined. Next, the features of the table are input into the model for Cell Type Classifier to obtain the class of each cell in the table. Finally, the Value is determined based on the information in Name and the class of each cell in the table.

3.4.1 *Name.*

First, an embedded representation of a given phrase and the text of each cell of a table in the same document was obtained by the multilingual-e5 [8] model. Then, we calculated the cosine similarity between the phrase and the embedded representation of the cell text. Next, the phrases were ranked in order of similarity, and the top K phrases that exceeded the threshold were designated as names. The threshold values for the similarity and the number of cases to be retrieved were set to the values that would result in the highest performance of Name according to the training data. Phrases consisting only of numbers or symbols were excluded. Japanese phrases meaning "note" that occur frequently were also excluded.

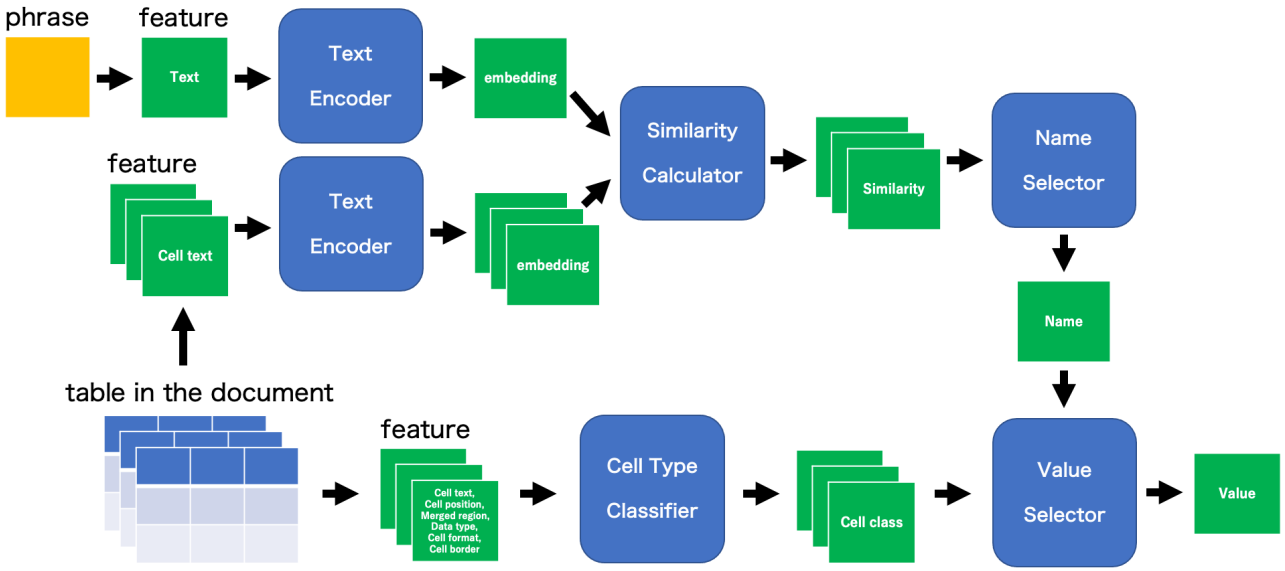

Figure 7: Overview of the processing of the proposed method in the TTRE subtask

3.4.2 Value.

Table features were input into the TUTA model, and each cell was classified into one of four classes: Metadata, Header, Attribute, and Data. Next, cells that belonged to the same row or column as the cell whose name was presumed to be Name were extracted, and cells that were classified into the Data class were designated as Value.

4 EXPERIMENTS

In this section, we examine the effectiveness of the exhaustive tree method for understanding table structure through the TDE and TTRE subtasks.

4.1 METHODS TO BE EVALUATED

4.1.1 TDE.

We compare the usefulness of three methods: the conventional default tree method, the proposed exhaustive tree method, and a method that does not consider the tree structure of the table (hereafter referred to as "no tree").

4.1.2 TTRE.

We compare four methods: a conventional method with a default tree, a proposed method with an exhaustive tree, a method that does not consider the tree structure of the table (hereinafter called "no tree"), and a method that does not consider the class of cells. A method that does not consider the class of cells is one that assumes that all cells belonging to the same row or column as the cell estimated to be Name are Value. We also examine the effect of different Text Encoders. The Text Encoder is a multilingual-e5 model, a sentence-luke model [10], a sentence-bert model [6], a deberta-v2 model [2], mdeberta-v3 model [1].

One general-purpose text embedding model with state-of-the-art performance is E5 (EmbEddings from bidirEctional Encoder rEpresentations). E5 is pre-trained by weakly supervised contrastive learning on a large unlabeled dataset, CCPairs. Fine tuning of this pre-trained model on a small labeled dataset yields even higher quality text embeddings. Using the pre-trained parameters of xlm-roberta-base as initial values, multilingual-e5 uses a wide variety of multilingual datasets (Filtered mC4, CC News, NLLB, Wikipedia, Filtered Reddit, S2ORC, Stackexchange, xP3, and Miscellaneous unsupervised SBERT data). For the models pre-trained on this multilingual dataset, we used labeled datasets (MS MARCO, NQ, Trivia QA, NLI from SimCSE, ELI5, DuReader Retrieval, KILT Fever, KILT HotpotQA, SQuAD, Quora Mr. TyDi, MIRACL) for fine tuning.

In addition to the pre-training task in BERT, LUKE is a new model that learns contextualized representations of words and entities, reaching state-of-the-art performance on a variety of entity-related tasks. The model treats words and entities in a given text as independent tokens and outputs their contextualized representations. In order to focus on entities here, we extend the self-attention mechanism to enable the calculation of word- and entity-aware scores.

SBERT is a model with a Siamese network structure that adds a pooling layer to the output of each of the two BERTs to obtain better sentence embedding than conventional BERT. This SBERT

Table 5: Scores of TDE subtask in formal run and late submission

ID	Method	Batch size	Epoch	F-measure
81	TUTA $_{\text{no tree}}$	2	44	**0.9537**
139	TUTA $_{\text{default tree}}$	2	44	0.9460
140	TUTA $_{\text{default tree}}$	2	130	0.9487
141	TUTA $_{\text{default tree}}$	4	64	0.9459
142	TUTA $_{\text{default tree}}$	4	130	0.9422
148	TUTA $_{\text{exhaustive tree}}$	2	54	0.9438
149	TUTA $_{\text{exhaustive tree}}$	2	116	0.9480
150	TUTA $_{\text{exhaustive tree}}$	4	82	0.9455
151	TUTA $_{\text{exhaustive tree}}$	4	127	**0.9503**

Table 6: Scores of TTRE subtask in formal run and late submission

ID	Method	Name	Value	Total
120	multilingual-e5-small	0.3198	0.1154	0.2176
125	multilingual-e5-small + TUTA $_{\text{no tree}}$	0.3198	0.2688	0.2943
144	multilingual-e5-small + TUTA $_{\text{default tree}}$	0.3198	0.2669	0.2934
158	multilingual-e5-small + TUTA $_{\text{exhaustive tree}}$	0.3198	0.2625	0.2912
122	multilingual-e5-base	**0.3221**	0.1186	0.2204
127	multilingual-e5-base + TUTA $_{\text{no tree}}$	**0.3221**	**0.2719**	**0.2970**
146	multilingual-e5-base + TUTA $_{\text{default tree}}$	**0.3221**	**0.2704**	**0.2962**
156	multilingual-e5-base + TUTA $_{\text{exhaustive tree}}$	**0.3221**	0.2659	0.2940
123	multilingual-e5-large	0.3212	0.1153	0.2182
128	multilingual-e5-large + TUTA $_{\text{no tree}}$	0.3212	0.2679	0.2945
147	multilingual-e5-large + TUTA $_{\text{default tree}}$	0.3212	0.2662	0.2937
154	multilingual-e5-large + TUTA $_{\text{exhaustive tree}}$	0.3212	0.2629	0.2920
116	sentence-luke-japanese-base-lite	0.3000	0.1120	0.2060
136	sentence-luke-japanese-base-lite + TUTA $_{\text{no tree}}$	0.3000	0.2563	0.2781
152	sentence-luke-japanese-base-lite + TUTA $_{\text{default tree}}$	0.3000	0.2545	0.2773
153	sentence-luke-japanese-base-lite + TUTA $_{\text{exhaustive tree}}$	0.3000	0.2537	0.2768
118	sentence-bert-base-ja-mean-tokens-v2	0.2805	0.1067	0.1936
138	sentence-bert-base-ja-mean-tokens-v2 + TUTA $_{\text{no tree}}$	0.2805	0.2435	0.2620
112	deberta-v2-tiny-japanese	0.2753	0.1029	0.1891
132	deberta-v2-tiny-japanese + TUTA $_{\text{no tree}}$	0.2753	0.2419	0.2586
113	deberta-v2-base-japanese	0.2810	0.1051	0.1931
133	deberta-v2-base-japanese + TUTA $_{\text{no tree}}$	0.2810	0.2453	0.2632
114	deberta-v2-large-japanese	0.2739	0.1015	0.1877
134	deberta-v2-large-japanese + TUTA $_{\text{no tree}}$	0.2739	0.2367	0.2553
110	mdeberta-v3-base	0.1983	0.0700	0.1342

not only acquires sentence embeddings better than conventional BERT, but also significantly reduces inference time.

DeBERTa is a modified version of BERT and RoBERTa that uses a disentangling attention mechanism to represent each word and its relative position as two independent vectors. During pre-training, the decoder is extended to mask absolute word positions. In addition, virtual adversarial learning is used during fine tuning. DeBERTa V2 augments the tokenizer's vocabulary, adds a convolutional layer to learn the local dependence of input tokens, shares the position and content projection matrices in the attention mechanism, and uses buckets to encode relative positions. DeBERTa V3 is an improved version of DeBERTa that uses Gradient Disentangled Embedding Sharing. mDeBERTa is a model trained on DeBERTa using CC100, a multilingual dataset.

The sentence-luke, sentence-bert, and deberta-v2 models used Japanese models, while the multilingual-e5 and mdeberta-v3 models used multilingual models. For the deberta-v2 model, Juman++ 2.0.0-rc3 [5, 7] was used during tokenization. The thresholds for the similarity and the number of cases to be retrieved were set to the values for which the performance of Name was highest depending on the training data.

Table 7: Detailed scores of TTRE subtask

Method	Name			Value			Total
	Precision	Recall	F1	Precision	Recall	F1	F1
multilingual-e5-base	**0.3556**	**0.4574**	**0.3221**	0.0857	**0.5069**	0.1186	0.2204
multilingual-e5-base + TUTA $_{\text{no tree}}$	**0.3556**	**0.4574**	**0.3221**	**0.2682**	0.5062	**0.2719**	**0.2970**
multilingual-e5-base + TUTA $_{\text{default tree}}$	**0.3556**	**0.4574**	**0.3221**	**0.2658**	0.5062	**0.2704**	**0.2962**
multilingual-e5-base + TUTA $_{\text{exhaustive tree}}$	**0.3556**	**0.4574**	**0.3221**	0.2646	**0.5065**	0.2659	0.2940
sentence-luke-japanese-base-lite	0.3401	0.4213	0.3000	0.0827	0.4643	0.1120	0.2060
sentence-luke-japanese-base-lite + TUTA $_{\text{no tree}}$	0.3401	0.4213	0.3000	0.2600	0.4639	0.2563	0.2781
sentence-luke-japanese-base-lite + TUTA $_{\text{default tree}}$	0.3401	0.4213	0.3000	0.2571	0.4639	0.2545	0.2773
sentence-luke-japanese-base-lite + TUTA $_{\text{exhaustive tree}}$	0.3401	0.4213	0.3000	0.2594	0.4641	0.2537	0.2768

4.2 RESULTS AND DISCUSSION

4.2.1 TDE.

Table 5 shows the experimental results of the TDE subtask. Compared to the proposed method (ID:148-151), the method that does not consider the tree structure of the table (ID:81) shows higher performance. The performance of the proposed method (ID:148-151) and the method with default tree (ID:139-142) did not differ significantly. There are three possible reasons for the low performance of the methods considering the tree structure of the table.

(1) The tree structure proposed by TUTA was built with information on joined cells, indentation, and formulas, but in this experiment, the tree structure was built only with information on joined cells.
(2) The cell unit assumed by TUTA was different from that of the TDE subtasks.
(3) The fact that there were few complex tables in the TDE subtask data set that required a tree structure.

As a specific example of the third case, in the case of a table like Figure 1, the TDE subtask mainly defined a single cell as an area surrounded by a ruled line, but TUTA defined a single cell as a text unit, which we considered to represent a more complex hierarchy.

4.2.2 TTRE.

Table 6 shows the experimental results of the TTRE subtask. Compared to the proposed method (ID:154, 156, 158), the method without considering the tree structure of the table (ID:127) shows higher performance on Value and Total. In all the results, the method without considering the tree structure of the table, the method with the default tree, the proposed method, and the method without considering the cell classes performed better in Value and Total, in that order. As for the effect of different Text Encoders, the method with multilingual-e5-base (ID:122, 127, 146, 156) showed the highest performance in Name, Value, and Total.

Table 7 shows the detailed experimental results of the method using multilingual-e5-base and the method using sentence-luke-japanese-base-lite. For all methods, recall was higher than precision. multilingual-e5-base was the best performing multilingual model, and sentence-luke-japanese-base-lite was the best performing Japanese model. Comparing Value between the method using TUTA and the method without TUTA, there was no significant difference in recall and a significant difference in precision. This indicates that it is very effective to determine the cell type and exclude cells other than those of the data class when determining Value. In all methods, recall was higher than precision. This may be due to the fact that Name is determined only by the similarity between a given phrase and cell values, resulting in the acquisition of tables that are different from those that should actually be referenced.

Table 8: Degree of agreement between multilingual model and Japanese model prediction results

degree of agreement	frequency (percentage)
both match correctly	441 (0.235)
both match incorrectly	677 (0.361)

Table 8 shows the degree of agreement between the prediction results using multilingual-e5-base and sentence-luke-japanese-base-lite methods for Name. In Name, there were 1,118 cases in which the predictions for each phrase matched perfectly for the two methods, accounting for 59.63% of the total number of cases. The number of cases correctly identified by both methods was 441, accounting for 23.52% of the total. The number of cases where both methods made exactly the same error was 677, accounting for 36.11% of the total.

5 ADDITIONAL EXPERIMENTS

In this section, we examine the effectiveness of exhaustive tree methods for understanding table structure using the deexcelerator (DeEx)[4] dataset. DeEx is a dataset for cell type classification. There are 6 classes of classification: metadata, notes, data, attributes, header, and derived, and their distributions are shown in Table 10. The DeEx dataset is characterized by the fact that most of the cells are classified into the data class. On the other hand, only a few cells are classified into the notes class.

5.1 METHODS TO BE EVALUATED

We verify the usefulness of the proposed method by comparing four methods: the conventional default tree method, the proposed exhaustive tree method, the no tree method that does not consider the tree structure of the table, and the raw tree method provided by the authors of TUTA. The pre-trained TUTA-implicit model was

Table 9: Scores of experiments with DeEx dataset

Method	metadata	notes	data	attributes	header	derived	macro-F1
TUTA $_{\text{no tree}}$	0.8574	**0.4825**	**0.9937**	**0.8114**	**0.8701**	**0.8092**	**0.8041**
TUTA $_{\text{default tree}}$	**0.8585**	0.4419	0.9930	**0.8382**	0.8547	0.7391	0.7876
TUTA $_{\text{exhaustive tree}}$	**0.8671**	**0.5589**	**0.9931**	0.8045	**0.8638**	0.7491	**0.8061**
TUTA $_{\text{raw tree}}$	0.8516	0.4271	0.9911	0.7888	0.8070	**0.7767**	0.7737

Table 10: Distribution by cell type

cell type	quantity	percentage
metadata	20,020	1.54%
notes	5,314	0.41%
data	1,232,762	94.58%
attributes	7,024	0.54%
header	21,810	1.67%
derived	16,538	1.27%

fine-tuned to be classified into 6 classes: metadata, notes, data, attributes, header, and derived. The architecture of the model was used without modification. The dataset was the DeEx dataset randomly divided into five parts. Cross-validation was used to calculate the average macro-F1. The batch size was set to 2 or 4, and the learning rate was set to 8e-6, up to 200 epochs.

5.2 RESULTS AND DISCUSSION

Table 9 shows the experimental results on the DeEx dataset. Of all the methods, TUTA $_{\text{exhaustive tree}}$ gave the best performance in macro-F1. TUTA $_{\text{exhaustive tree}}$ also showed 3.24% higher performance than TUTA $_{\text{raw tree}}$ in macro-F1. However, there was not a significant difference between TUTA $_{\text{no tree}}$ and TUTA $_{\text{exhaustive tree}}$. For all methods, the F-measure for the data class was the highest and the F-measure for the notes class was the lowest. The number of data classes is the largest and the number of notes classes is the smallest, suggesting that the distribution of cell types has a significant impact. TUTA $_{\text{exhaustive tree}}$ has the best performance in the notes class, indicating that the proposed method may have a high ability to learn effectively from a small number of data.

6 CONCLUSIONS

This paper describes the methods and results of Team KSU for the UFO task at NTCIR-17. In the TDE subtask, we designed methods for cell type classification using exhaustive tree structures based on the spanning sizes of the merged cells in the table. In the TTRE subtask, we designed methods for cell retrieval based on the cell class. Scores on the F-measure in the formal run were 95.37% for ID81 in TDE and 9.18%, 4.08%, and 6.63% for ID99 on the Name, Value, and Total, respectively, in TTRE. Scores on the F-measure including the formal run and late submission run were 95.37% for ID81 in TDE and 32.21%, 27.19%, and 29.70% for ID127 on the Name, Value, and Total, respectively, in TTRE. The experimental results show that the method without considering the tree structure of the table (ID:81) performs better than the proposed method (ID:148-151) in the TDE subtask. In the TTRE subtask, the method without considering the tree structure of the table (ID:127) performed better on Value and Total than the proposed method (ID:154, 156, 158). There are three expected reasons for the low performance of the methods considering the tree structure of the table: "The tree structure proposed by TUTA was built with information on joined cells, indentation, and formulas, but in this experiment, the tree structure was built only with information on joined cells.", "The cell unit assumed by TUTA was different from that of the TDE subtasks.", and "The fact that there were few complex tables in the TDE subtask data set that required a tree structure.". Additional experiments on the DeEx dataset show that the proposed method performs best on macro-F1. In addition, the proposed method outperforms the previous study by 3.24% on macro-F1.

REFERENCES

[1] Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. arXiv:2111.09543 [cs.CL]

[2] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION. In *International Conference on Learning Representations*. https://openreview.net/forum?id=XPZIaotutsD

[3] Yasutomo Kimura, Hokuto Ototake, Kazuma Kadowaki, Takahito Kondo, and Makoto P. Kato. 2023. Overview of the NTCIR-17 UFO Task. *Proceedings of The 17th NTCIR Conference* (12 2023). https://doi.org/10.20736/0002001321

[4] Elvis Koci, Maik Thiele, Josephine Rehak, Oscar Romero, and Wolfgang Lehner. 2019. DECO: A Dataset of Annotated Spreadsheets for Layout and Table Recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 1280–1285. https://doi.org/10.1109/ICDAR.2019.00207

[5] Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological Analysis for Unsegmented Languages using Recurrent Neural Network Language Model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 2292–2297. https://doi.org/10.18653/v1/D15-1276

[6] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[7] Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2018. Juman++: A Morphological Analysis Toolkit for Scriptio Continua. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Brussels, Belgium, 54–59. https://doi.org/10.18653/v1/D18-2010

[8] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *arXiv preprint arXiv:2212.03533* (2022).

[9] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1780–1790.

[10] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In *EMNLP*.