HTML-LSTM: Information Extraction from HTML Tables in Web Pages using Tree-Structured LSTM

Kazuki Kawamura* and Akihiro Yamamoto

Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan.
Kazuki.Kawamura@sony.com
akihiro@i.kyoto-u.ac.jp

Abstract. In this paper, we propose a novel method for extracting information from HTML tables with similar contents but with a different structure. We aim to integrate multiple HTML tables into a single table for retrieval of information containing in various Web pages. The method is designed by extending tree-structured LSTM, the neural network for tree-structured data, in order to extract information that is both linguistic and structural information of HTML data. We evaluate the proposed method through experiments using real data published on the WWW.

1 Introduction

Tables in Web pages are useful for displaying data representing relationships. We can find them on Web pages showing, for example, syllabus in universities, product information in companies, and flight information in airlines. Our research aims at integrating tables from various pages but representing the same type of relational data into a single table for retrieval of information. In this paper, we propose a novel method, called HTML-LSTM, for extracting information from tables in HTML with the same type of contents but with a different structure.

When we browse Web pages, tables representing the same type of relationships look to have similar visual structures but may not be matched completely. In some tables, every tuple is represented in a row, and in other pages, it is in a column. The ordering features (or attributes) may be different. Moreover, they are not always presented as similar HTML source code because different pages are usually designed by different organizations. The source code may contain noises such as codes for visual decorations and additional information. Therefore in order to extract and amalgamate relations from tables in different Web pages, we need to unify them in a common set of features (that is, relational schema), as well as in the structure in the level of HTML source codes. For this purpose, many methods have been proposed [2], but they work well only

* Now at Sony Group Corporation.

for HTML tables having almost the same structure in source codes or for the case where each feature to be extracted clearly differs from each other. Therefore, extracting and amalgamating relations from tables of similar content but of non-uniform structure is still a major challenge. We solve this problem by using neural networks developed recently and present our solution as HTML-LSTM.

Some neural networks for extracting effectively features from tree and graph structures have been proposed [5,7,13,25]. The Tree-LSTM [25] neural network is a generalization of LSTM to handle tree-structured data, and it is shown that the network effectively works mainly as a feature extractor for parse trees in the field of natural language processing. Since the source codes of HTML are also parsed into tree structures, we extend Tree-LSTM into HTML-LSTM for extracting features in relational data and tree structure from HTML data simultaneously.

We cannot apply Tree-LSTM to HTML data for the following reason: In parse trees of texts in natural language, linguistic information is given only in the leaves, while the other nodes are given information about the relationship between words. Therefore, Tree-LSTM transfers information in the direction from the leaves to the root and is often has been applied to tasks such as machine translation and sentiment analysis [6,25]. On the other hand, when an HTML source code is parsed into a tree, information is attached not only leaves but internal nodes and the root. In addition, when extracting the features of each element in HTML source codes for tables representing relational data, the path from the root tag `<table>` to the element is quite essential. This means that for extracting information from table data, manipulating parsing trees in the direction from the root to leaves as well as in the direction from leaves to the root. Therefore HTML-LSTM is designed so that information can be transferred in both directions.

In applying HTML-LSTM to information extraction from real HTML data, we first extract the substructure of a table in the data and convert it into a tree. Next, we extract features from the obtained tree structure using HTML-LSTM and classify the features to be extracted for each node. Finally, we integrate the extracted information into a new table. We also introduce a novel data augmentation method for HTML data in order to improve the generalization performance of information extraction.

We evaluate and confirm the effectiveness of the HTML-LSTM method for integrating HTML data formats by applying the method explained above to tables of preschools of local governments and tables of syllabuses published by universities, which are published on the Web. As the results, we succeeded in extracting the required information with an accuracy of an F_1-measure of 0.96 for the data of preschools and an F_1-measure of 0.86 for the syllabus data. Furthermore, our experimental results show that HTML-LSTM outperforms Tree-LSTM.

This paper is organized as follows. In Section 2, we describe previous methods for extracting information from Web pages. In Section 3, we introduce the architecture of our proposed method, HTML-LSTM, and how to use HTML-LSTM

for information extraction. In Section 4, we summarize the results of experiments using HTML data on the Web. Finally, in Section 5, we provide our conclusion.

2 Related Work

Extracting information from documents and websites and organizing it into a user-friendly form is called information extraction and is widely studied. The research field originates with the Message Understanding Conference (MUC) [8,24], which started in the 1980s. At this conference, every year, a competition is held to extract information from newspapers on, for example, terrorist activities, product development, personnel changes, corporate mergers, rocket launches, and participants competed for some scores evaluating their technique for information extraction.

In the early years of the research area, rule-based methods were widely used [4,22], where rules are defined based on features such as the representation of the characters of the tokens in the sentence, the notation of the tokens (uppercase, lowercase, mixed case of uppercase and lowercase, spaces, punctuation, *etc.*), and the parts of speech of the tokens. Such rule-based methods require experts who manually create rules depending on the types of objects they want to extract. Since it is very time-consuming, algorithms have been developed to automatically create rules using labeled data [1,3,23]. In recent years, statistical methods have also been used in order to treat documents which may have many noises. Example of methods are Support Vector Machine (SVM) [26], Hidden Markov Model (HMM) [21], Maximum Entropy Markov Model (MEMM) [17], Conditional Markov Model (CMM) [16], and Conditional Random Fields (CRF) [19].

Our key idea is to introduce natural language processing methods for information extraction and simultaneously handle structural and linguistic information. Every Web page is written as a source code in HTML, with a clear tree structure after parsing it. The method to extract a specific part from a Web page using the structural information is called Web wrapper [14]. Some of the methods extract information by regarding a specific part in a Web page as data in a tree structure [11,18]. These methods work for Web pages of almost similar structure, and it is difficult to apply them to pages whose structure is completely different, but the meaning of them is the same. This situation often appears in tables representing relational data.

Other types of methods are treating linguistic features of Web pages based on natural language processing, in other words, treating the meaning of the texts on each page. These methods have the disadvantage that they cannot capture the structure of pages. However, natural processing has greatly advanced thanks to the introduction of greatly improved neural network techniques. Some researchers propose new types of neural networks which treat the parsing tree of texts in natural languages. This motivates us to apply such neural networks to extracting information taking into account structure and meaning simultaneously.

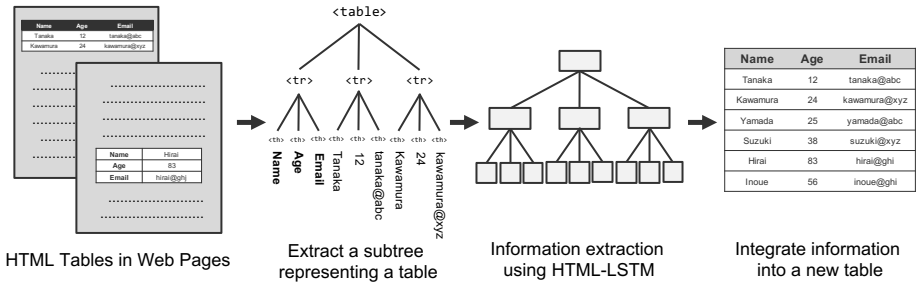


Fig. 1: The **HTML-LSTM** framework for information extraction and integration from HTML tables in web pages

3 HTML-LSTM

The overview of our proposed method is shown in Fig. 1. First, we extract the substructure of the table from the entire HTML data. Since Table tags (`<table> </table>`) are usually used to represent tables in HTML, we can extract the table by focusing on the region surrounded by the table. tags. Next, we convert the HTML data into a tree structure called DOM tree so that HTML-LSTM can take it as its input. Then, the obtained tree structure data is input to HTML-LSTM for feature extraction, and the obtained features of nodes are used to classify which attribute values each node belongs to. Finally, we pick up the nodes' information classified into the extraction target's attribute values and integrate the information in a new single table.

3.1 Extracting Information

In this subsection, we explain the details of HTML-TLSM and the extraction of information from HTML data using it. The workflow of information extraction is shown in Fig. 2. First, each element of the HTML data is encoded using Bi-LSTM (Bidirectional LSTM) [10,20] in order to obtain the language representation of each element. Next, HTML-LSTM is applied to obtain the features of the HTML data, considering the relationship between the positions of the elements in the parsed tree. In order to use the information of the whole tree structure of HTML data effectively, HTML-LSTM extends Tree-LSTM, in which the information flows only from leaf to root, to enable the flow of information from root to leaf as well as from leaf to root. Finally, the features of each node obtained by the HTML-LSTM are passed through the fully connected layer, and the softmax classifier is applied to determining which attribute value each node is classified as.

Encoding of HTML Data: The DOM tree that is fed into HTML-LSTM is obtained by parsing the HTML source code. In general, when parsing HTML data to a tree, the values of the nodes in the tree structure are HTML tags. In

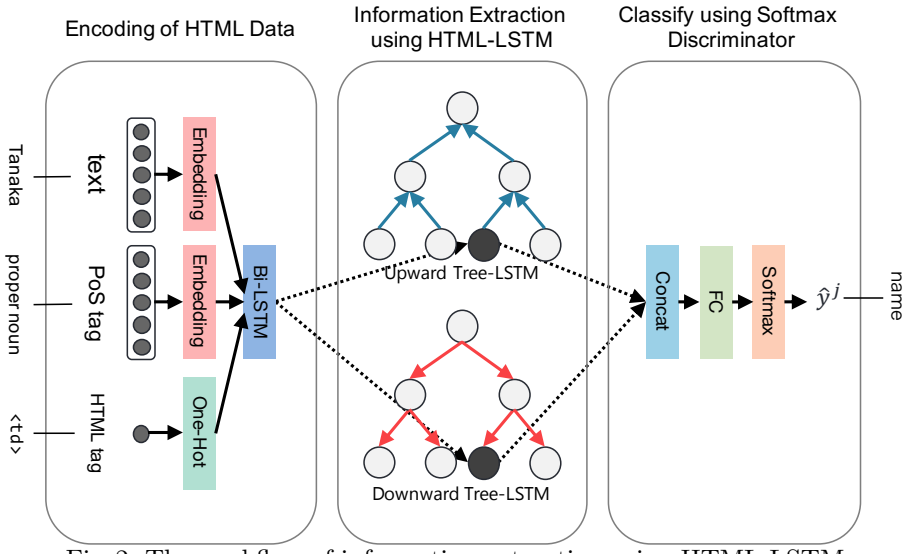


Fig. 2: The workflow of information extraction using HTML-LSTM

our method, in order to extract information by effective use of the linguistic and structural information of HTML, each node of the tree has three types of values: the HTML tag, the text between the start and end tags, and the PoS (part-of-speech) tags of the text, as shown in Fig. 3. The extracted text is treated as a sequence of words. The PoS tag is the sequence of parts-of-speech data corresponding to the sequence of words of the text. If the attribute names to be extracted differ from a Web page to a Web page, a dictionary is used to unify the attribute names. Finally, the obtained tree is converted into a binary tree because HTML-LSTM accepts only binary trees as input.

After converting the HTML data into a binary tree, the text, the sequence of PoS tags, and the HTML tag in each node are converted using a neural network to a representation for input to the HTML-LSTM. In particular, we combine the one-hot encoding of the tag t^j of a node j and the text w^j and PoS tags p^j converted by the embedding matrices E_{content} and E_{pos}, and feed them to Bi-LSTM:

$$
\begin{aligned}
e_t^j &= \left[\text{onehot}(t^j) \| E_{\text{content}}(w_t^j) \| E_{\text{pos}}(p_t^j)\right], \\
\overrightarrow{h_t^j} &= \overrightarrow{\text{LSTM}}(e_t^j, \overrightarrow{h_{t-1}^j}), \\
\overleftarrow{h_t^j} &= \overleftarrow{\text{LSTM}}(e_t^j, \overleftarrow{h_{t-1}^j}),
\end{aligned}
$$

where onehot is a function that converts a tensor to one-hot a representation and $\|$ is the concatenation of two tensors. The function $\overrightarrow{\text{LSTM}}$ is the forward LSTM and $\overleftarrow{\text{LSTM}}$ is the backward LSTM of the Bi-LSTM. The outputs at the last time T of the forward and backward LSTMs are combined to obtain the representation $x^j = \left[\overrightarrow{h_T} \| \overleftarrow{h_T}\right]$ for each node.

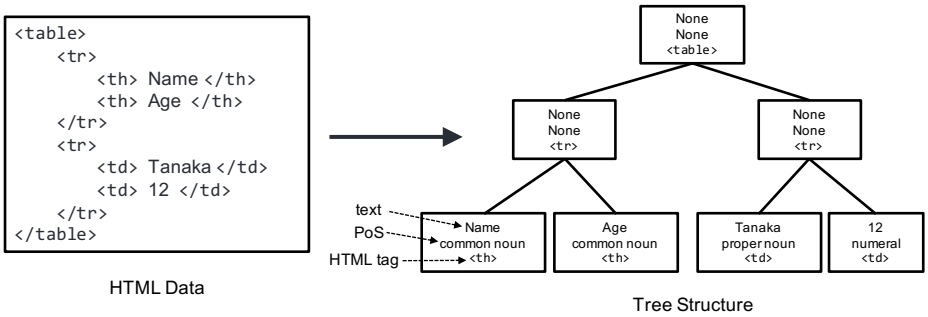


Fig. 3: Example of converting HTML data to a tree structure

HTML-LSTM: HTML-LSTM is composed of Upward Tree-LSTM, in which information flows from leaves to roots, and Downward Tree-LSTM, in which information is transmitted from roots to leaves. Upward Tree-LSTM uses Binary Tree-LSTM [25]. The model is expressed as follows:

$$
\begin{aligned}
i^j &= \sigma\Big(W^{(i)}x^j + \sum_{k\in\{\mathrm{L,R}\}} U_k^{(i)}h_k^j + b^{(i)}\Big),\\
f^j_{[\mathrm{L,R}]} &= \sigma\Big(W^{(f)}x^j + \sum_{k\in\{\mathrm{L,R}\}} U^{(f)}_{[\mathrm{L,R}]k}h_k^j + b^{(f)}\Big),\\
o^j &= \sigma\Big(W^{(o)}x^j + \sum_{k\in\{\mathrm{L,R}\}} U_k^{(o)}h_k^j + b^{(o)}\Big),\\
u^j &= \tanh\Big(W^{(u)}x^j + \sum_{k\in\{\mathrm{L,R}\}} U_k^{(u)}h_k^j + b^{(u)}\Big),\\
c^j &= i^j \odot u^j + \sum_{k\in\{\mathrm{L,R}\}} f_k^j \odot c_k^j,\\
h^j &= o^j \odot \tanh\left(c^j\right).
\end{aligned}
$$

Upward Tree-LSTM has a forget gate f^j, an input gate i^j, an output gate o^j, a memory cell c^j, and a hidden state h^j, just like a simple LSTM. In the expressions, σ denotes the sigmoid function and $\odot$ denotes the element-wise product. Both of the parameters W and U are weights, and b is the bias. All of these parameters are learnable. As shown in Fig. 4a, Upward Tree-LSTM is a mechanism that takes two inputs and gives one output. In this case, the forget gate uses its own parameters U_{L} and U_{R} to select the left and right children's information $c^j_{\mathrm{L}}, c^j_{\mathrm{R}}$.

On the other hand, Downward Tree-LSTM, in which information flows from roots to leaves, is as follows:

$$
\begin{aligned}
i^j &= \sigma\left(W^{(i)}x^j + U^{(i)}h^j + b^{(i)}\right), \\
f^j_{[\mathrm{L,R}]} &= \sigma\left(W^{(f)}x^j + U^{(f)}_{[\mathrm{L,R}]}h^j + b^{(f)}\right), \\
o^j_{[\mathrm{L,R}]} &= \sigma\left(W^{(o)}x^j + U^{(o)}_{[\mathrm{L,R}]}h^j + b^{(o)}\right), \\
u^j &= \tanh\left(W^{(u)}x^j + U^{(u)}h^j + b^{(u)}\right), \\
c^j_{[\mathrm{L,R}]} &= i^j \odot u^j + \sum_{k\in\{\mathrm{L,R}\}} f^j_k \odot c^j, \\
h^j_{[\mathrm{L,R}]} &= o^j_{[\mathrm{L,R}]} \odot \tanh(c^j_{[\mathrm{L,R}]}).
\end{aligned}
$$

As shown in Fig. 4b, this mechanism takes one input and gives two outputs. In this case, the forgetting gate generates two outputs by operating on the input c^j with different parameters U_L and U_R, so that the model can choose information to transmit to the left and right children.

Finally, we combine the Upward Tree-LSTM hidden state $h^j_{\uparrow}$ and the Downward Tree-LSTM hidden states $h^j_{l\downarrow}$, $h^j_{jr\downarrow}$ to obtain $h^j = [h_{j\uparrow}|h_{jl\downarrow}\|h_{jr\downarrow}]$ as the feature of each node. A softmax classifier predicts the label $\hat{y}^j$ from among the N classes,

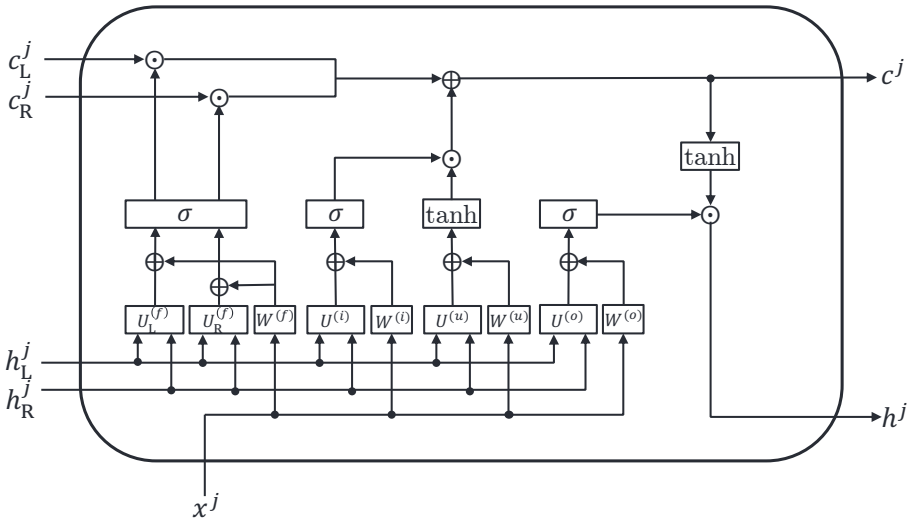
$$
\begin{aligned}
p &= \mathrm{softmax}\left(W^{(s)}h^j + b^{(s)}\right), \\
\hat{y}^j &= \underset{i\in\{1,\ldots,N\}}{\arg\max}\, p_i,
\end{aligned}
$$

where $W^{(s)}, b^{(s)}$ are the learnable parameters. The set of labels consists of the set of attributes to be extracted and a special label *Other* that does not belong to any of the attributes to be extracted. For example, when extracting the attributes *Name* and *Age* from the table shown in Fig. 1 (left), there are three types of classes: *Name*, *Age*, and *Other*. The attribute values "Hirai" and "8" in the table belong to the class of *Name* and *Age*, respectively, while the attribute value "hirai@ghi" and the attribute names "Name", "Age", and "Email" belong to the class of *Other*. We classify all the nodes in the HTML tree to determine what attribute each node is (or is not included in any of the attributes to be extracted).

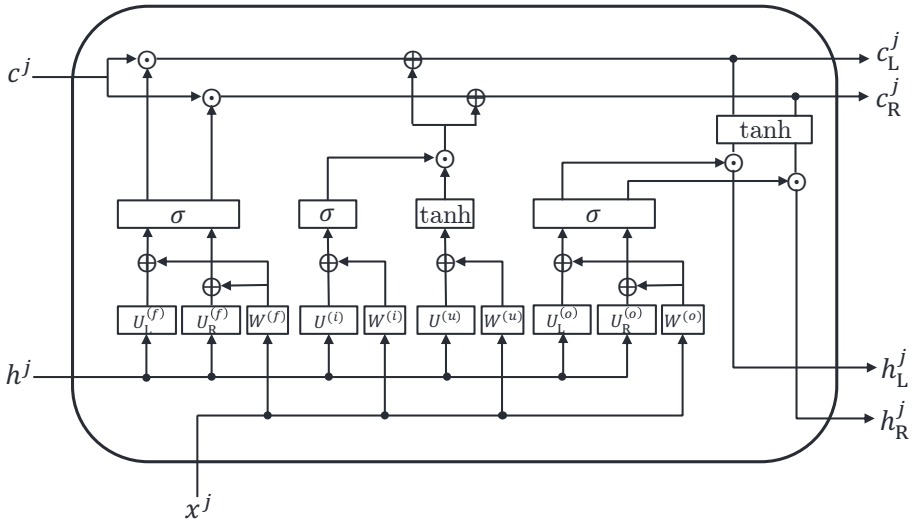
Every HTML source code may contain much information that is not required to be extracted and noise for decoration. In treating real data, most nodes are not the target of extraction, and therefore the trees as the inputs of HTML-LSTM tend to be imbalanced. In order to treat such trees, we use Focal Loss [15] which is an extension of Cross-entropy Loss to deal with class imbalance. Focal Loss is defined as follows with the correct label N and one-hot vector t:

$$
\mathcal{L}_{\mathrm{focal}} = -\alpha_i \sum_{i=1}^{N} (1-p_i)^{\gamma}\, t_i \log\left(p_i\right),
$$

where α_i is the frequency inverse of each class and γ is the hyperparameter.



(a) Upward Tree-LSTM



(b) Downward Tree-LSTM

Fig. 4: HTML-LSTM architecture

Also, in order to improve scores of the model's recall and precision in a well-balanced way, F1 loss is used jointly. The F1 Loss is given by $1 - F_1$, where F_1 is the average of the F_1 measures of each class. This is denoted as $\mathcal{L}_{\mathrm{f1}}$, and the final loss function $\mathcal{L}$ is given as

$$\mathcal{L} = \mathcal{L}_{\mathrm{focal}} + \mathcal{L}_{\mathrm{f1}}.$$

Furthermore, the table data on the Web is equivalent to the original data even if the order of the rows and columns of the table is changed. Therefore, we introduce a data augmentation technique that randomly changes the order of rows and columns, thereby increasing the number of HTML data used for training.

3.2 Integrating Information

After classifying the class of each node in the HTML tree using HTML-LSTM, we extract the required information from the tree and integrate it into a new table. For each class (attribute), the node with the highest classification score in the HTML tree is selected, and the text of that node is extracted and put into the table. Here, the classification score is the maximum value of the output of the softmax classifier for each class, *i.e.*, $\max_i p_i$. The left side of Fig. 5 shows an example of the tree after classifying the class of each node. Each node contains the text of the original element (top row in the box), the class with the highest classification score (bottom row left in the box), and the classification score (bottom row right in the box). For example, to extract the information of *Name* class from this tree, extract the text of the node elements classified as *Name* class. In this case, the only node classified in the *Name* class is "Tanaka", so we extract "Tanaka" as the *Name* information of this HTML tree and put it in the table. In the same way, when we extract the *Age* class information, we find that there are two nodes classified as *Age* class: "12" and "None". Comparing the classification scores, the classification score of "12" is 0.87, and "None" is 0.23. Since the score of "12" is higher, we extract "12" as the *Age* information of this HTML tree and put it in the table. However, if multiple values should be classified into a certain class, there will be omissions in the extraction. Therefore, when it is expected that there are multiple values classified into a certain class in the web table, a threshold value is set, and all the texts of the nodes that exceed the threshold value are extracted. For example, suppose that in a given HTML tree, there are three nodes classified into the *Name* class, "Tanaka", "Suzuki", and "Apple" with classification scores of 0.97, 0.89, and 0.23, respectively. If the threshold is set to 0.5, then "Tanaka" and "Suzuki" will be extracted as the *Name* class information for this tree.

3.3 Implementation Details

The dimensions of the hidden layer of the model are 128 and 5 for the embedding layer of the text and part-of-speech tags, respectively, in the information

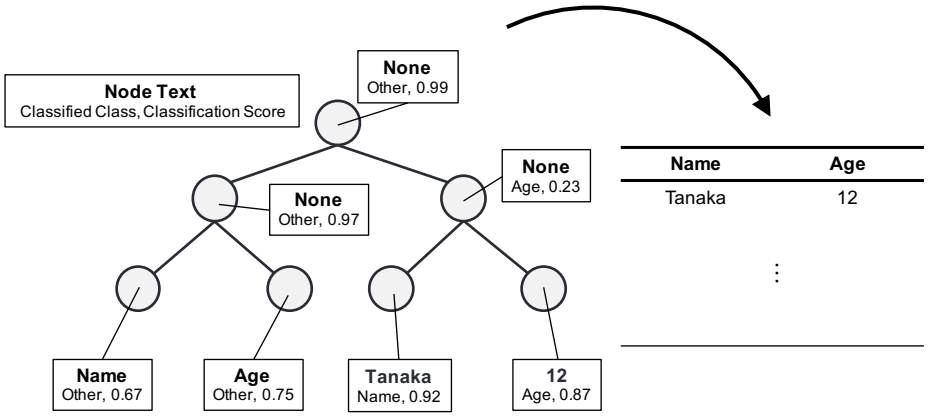


Fig. 5: Example of information integration

embedding part of the HTML data. The dimensions of the hidden layer of the HTML-LSTM are 64, and the dimensions of the linear layer used for classification are 64.

We use Adam [12] as the optimization algorithm, with a minibach size of 128. The learning rate starts from 10^{-2} and is divided 2 every 15 epochs, and the model is trained for 50 epochs. The parameters of Adam are α of 10^{-2}, β_1 of 0.9 and β_2 of 0.999. We use Dropout [9] with a probability of 0.5 is used to prevent overfitting.

4 Experiments

We evaluate our method on tables of preschools published by local governments and tables of syllabus published by universities. These data are published on the Web, and the information is presented using table structures represented by HTML. For evaluation, we use Recall, Precision, and their harmonic mean, F_1 measure. F_1 measure and Precision and Recall are defined by

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$
$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN},$$

where TP, TN, FP, and FN are the true positives, true negatives, false positives, false negatives, respectively.

4.1 Experiments on Preschool Data

Many local governments publish a list of preschools in their localities in a table on the Web. Those pages have common information, such as the preschools' name,

Table 1: Information extraction result for the preschool data

Attribute	Precision	Recall	F_1 measure
name	0.94	1	0.97
address	0.92	1	0.96
telephone number	0.87	1	0.92
other	1	0.98	0.99
mean	0.93	0.99	0.96

Table 2: Information integration result for the preschool data

Name	Address	Telephone number
Tourin Kindergarten	43 Shichiku Takanawacho Kita-ku	492-4717
Kamo Nursery Room	59 Kamigamo Ikedonocho Kita-ku	585-5958
Nonohana Preschool	37 Koyama Nishionocho Kita-ku	354-6927
Nisihuji Preschool	1584-1 Nishifujicho	0848-55-6920
Mitsugi Chuo Preschool	94 Mitsugicho Hanajiri	0848-76-0044

address, and phone number, but each page has a different HTML structure. In this experiment, we extracted and integrated the information of *name*, *address*, and *phone number* from these pages. We collected a total of 47 HTML data from 47 local governments that contain information on preschools for the experiment. The HTML data were converted into a tree with a text, PoS tags, and HTML tags at each node. The obtained ordered trees had 22–249 nodes (107 nodes on average) and contained 16 types of PoS tags. Since the data collected for the experiment was in Japanese, the word boundaries are not obvious. Therefore, the series of text and PoS tags were obtained by morphological analysis using Janome[1]. We also unified attribute names that have the same meaning but different notation, and labeled each node manually. The classes were four types of labels: *name, address, phone number*, and *other*.

Table 1 shows the results of information extraction for each attribute in the case of 10 fold cross-validation of preschool data. Table 2 shows the integrated table of the information extracted from the preschool data. We can see that our model does good work on information extraction and integration from these results.

4.2 Experiments on Syllabus Data

The syllabus is the data which shows the contents of lectures in universities and is published on the Web by many universities, mainly using a table format. We collected the syllabus from 22 different universities on the Web and used the HTML data of 20,257 pages for the experiment. The syllabus data was converted

[1] https://mocobeta.github.io/janome/

Table 3: Information extraction result for the syllabus data

Attribute	Precision	Recall	F_1 measure
course title	0.76	0.82	0.77
instructor name	0.81	0.79	0.80
target student	0.90	0.76	0.82
target year	0.94	0.75	0.83
year/term	0.97	0.83	0.89
day/period	0.89	0.87	0.88
number of credits	0.83	0.94	0.88
other	0.99	0.99	0.99
mean	0.89	0.84	0.86

into a tree structure in the same way as the data of preschools, and the attribute names with different notations were unified and labeled. The obtained ordered tree has 19 to 1,591 nodes (109 on average) and contains 25 kinds of tags. Since some of the obtained trees contain much noise other than necessary information, we clipped the nodes after the 100th node in the post-order in the syllabus data. The extracted attributes are *course title, instructor name, target student, target Year, year/term, day/period*, and *number of credits*. Therefore, there are eight types of labels in the syllabus data, including *other* in addition to these seven classes.

Table 3 shows the results of information extraction for each attribute in the case of 5 fold cross-validation of syllabus data. In each split, 18 (or 17) of the 22 universities were used for training, and 4 (or 5) were used for testing. Table 4 shows the integrated table of the information extracted from the syllabus data. The F_1 measure of the information extraction in the syllabus data is less than that in the preschool data. We believe this is because more attributes are extracted than in the preschool data, and more noise is included in the syllabus data. The result of the information integration shows that information can be extracted even in the blank areas (*i.e.*, areas that originally had no information). This shows that our model can use not only linguistic information but also structural information.

Table 4: Information integration result for the syllabus data

Course Title	Instructor Name	Target Student	Target Year	Year/Term	Day/Period	Number of Credits
Instrumental Analysis, Adv. II	OOE KOUICHI Graduate School of Engineering Professor	Graduate Student	Master's student Doctoral student	Second semester	Thu. 4, 5	1
Advanced Studies: Educational of Clinical Psychologist II	NISHI MINAKO Graduate School of Education Associate Professor	Graduate student	Doctoral student	Second semester	Tue. 5	1
Tax Law II	OKAMURA TADAO	Graduate student	2, 3	Second semester	Fri. 5	
Advanced Materials Science & Engineering in Industries II	TSUJI NOBUHIRO Graduate School of Engineering Professor	Graduate student	Master's student Doctoral student	Second semester	Tue. 4	2
Advanced Reading on Clinical Psychology I	NATORI TAKUJI Part-time Lecturer	Graduate student	Master's student	First semester	Tue. 1	
Organic Chemistry in Food Science III	IRIE KAZUHIRO Graduate School of Agriculture Professor	Undergraduate student	2nd year students	Second semester	Tue. 3	2
Advanced Environmental Biophysics	SAKABE AYAKA Hakubi Center Assistant Professor KOSUGI YOSHIKO Graduate School of Agriculture Professor	Graduate student		Intensive First semester	Intensive, First semester Scheduled for July 3, 27, 28, and 29	2
Innovative Humano-habitability	YANAGAWA AYA Research Institute for Sustainable Humanosphere Assistant Professor HATA TOSHIMITSU Research Institute for Sustainable Humanosphere Lecturer YOSHIMURA TSUYOSHI Research Institute for Sustainable Humanosphere Professor	Graduate student		Intensive, First semester	Intensive, 5/8 (Fri.), 5/15 (Fri.), 5/22 (Fri.)	2
Exercises in Calligraphy and Copying B	HASEGAWA CHIHIRO Graduate School of Human and Environmental Studies Associate Professor	Undergraduate student	2nd–4th year students	Second semester	Wed. 2	2
Energy and Information, Adv.	OOHAYASHI FUMIAKI Part-time Lecturer	Graduate student	Doctoral students	Intensive, First semester	Intensive	2
Primary German A	KOMODA NAMI Part-time Lecturer	Undergraduate student	All students	First semester	Tue. 3	2

Table 5: Ablation study result

Method	F_1 measure
Tree-LSTM [25] (Upward Tree-LSTM)	0.8285
HTML-LSTM (Upward Tree-LSTM + Downward Tree-LSTM)	0.8414
HTML-LSTM w/ HTML data augmentaion	0.8575

4.3 Ablation Experiments

We conducted ablation studies to investigate the effect of adding root-to-leaf information transfer, which is the opposite direction of the traditional Tree-LSTM, and the effect of HTML data augmentation introduced in this study. In the HTML data augmentation, the order of any pair of rows and any pair of columns in the table was switched with a probability of 0.5. The setting of the experiment is the same as the previous experiment on syllabus data, and we compare the average values of all classes of F_1 measure of the traditional Tree-LSTM, our HTML-LSTM, and the HTML-LSTM with data augmentation.

The results are shown in the Table 5. This result shows that the ability of information extraction can be improved by using not only the root-to-leaf direction but also the leaf-to-root direction. We can also see that the data augmentation of HTML can further improve the accuracy of information extraction.

5 Conclusion

In this paper, we proposed HTML-LSTM, a method for extracting and integrating required information from tables contained in multiple Web pages. The method is an extension of Tree-LSTM, which is mainly used in the field of natural language processing and extracts words in texts attached to the leaves of DOM trees of HTML data in a bottom-up manner. Our method treats DOM trees in a bottom-up manner and then a top-down manner to extract sequences of part-of-speeches and tags attached to nodes in the DOM trees. We applied HTML-LSTM to a list of childcare facilities and syllabus data that are opened on the Web and confirmed that HTML-LSTM could extract information with F_1 measures of 0.96 and 0.86, respectively.

In the future, we would improve HTML-LSTM to extract information from fragments of HTML data other than tables. Such fragments are also transformed into DOM trees. For tables or lists, some special tags are prepared in HTML, but other fragments may not have such tags. In order to overcome the problem, choosing good positive and negative examples would be important. Also, modifying the HTML-LSTM algorithm would be needed.

References

1. Aitken, J.S.: Learning Information Extraction Rules: An Inductive Logic Programming Approach. ECAI pp. 355–359 (2002)
2. Chang, C.H., Kayed, M., Girgis, M., Shaalan, K.: A survey of web information extraction systems. IEEE Transactions on Knowledge and Data Engineering 18(10), 1411–1428 (2006)
3. Ciravegna, F.: Adaptive Information Extraction from Text by Rule Induction and Generalisation. IJCAI 2, 1251–1256 (2001)
4. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Enviroment for Robust NLP Tools and Applications. ACL pp. 168–175 (2002)
5. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. NeurIPS pp. 3844–3852 (2016)
6. Eriguchi, A., Hashimoto, K., Tsuruoka, Y.: Tree-to-Sequence Attentional Neural Machine Translation. ACL 2, 823–833 (2016)
7. Goller, C., Kuechler, A.: Learning Task-Dependent Distributed Representations by Backpropagation through Structure. Neural Networks 1, 347–352 (1996)
8. Grishman, R.: Message Understanding Conference-6: A Brief History. COLING pp. 466–471 (1996)
9. Hinton, G.: Dropout : A Simple Way to Prevent Neural Networks from Overfitting. JMLR pp. 1929–1958 (2014)
10. Hochreiter, S., Urgen Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9(8), 1735–1780 (1997)
11. Kashima, H., Koyanagi, T.: Kernels for Semi-Structured Data. ICML pp. 291–298 (2002)
12. Kingma, D.P., Ba, J.L.: Adam: A Method for Stochastic Optimization. ICLR (2015)

13. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. ICLR (2017)
14. Kushmerick, N.: Wrapper Induction: Efficiency and Expressiveness. Artificial Intelligence 118(1-2), 15–68 (2000)
15. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal Loss for Dense Object Detection. ICCV pp. 3844–3852 (2017)
16. Malouf, R.: Markov Models for Language-Independent Named Entity Recognition. CoNLL pp. 187–190 (2002)
17. Michael, A.: Maximum Entropy Markov Models for Information Extraction and Segmentation Andrew. ICML pp. 591–598 (2000)
18. Muslea, I., Minton, S., Knoblock, C.: Active Learning for Hierarchical Wrapper Induction. AAAI p. 975 (1999)
19. Peng, F., McCallum, A.: Information Extraction from Research Papers using Conditional Random Fields. Information Processing and Management 42(4), 963–979 (2006)
20. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing 45(11), 2673–2681 (1997)
21. Seymore, K., Mccallum, A., Rosenfeld, R.: Learning Hidden Markov Model Structure. AAAI Workshop pp. 37–42 (1999)
22. Shaalan, K., Raza, H.: Arabic Named Entity Recognition from Diverse text Types. GoTAL 5221, 440–451 (2008)
23. Soderland, S.: Learning Information Extraction Rules for Semi-Structured and Free Text. Machine Learning 34(1), 233–272 (1999)
24. Sundheim, B.M.: Overview of the Fourth Message Understanding Evaluation and Conference. Fourth Message Understanding Conference pp. 3–22 (1992)
25. Tai, K.S., Socher, R., Manning, C.D.: Improved Semantic Representations from Tree-Structured Long Short-Term Memory Networks. ACL-IJCNLP 1, 1556–1566 (2015)
26. Takeuchi, K., Collier, N.: Use of Support Vector Machines in Extended Named Entity Recognition. COLING pp. 1–7 (2002)