ISSN 1000-9000(Print)
/1860-4749(Online)
CODEN JCTEEM

JCST

Vol.37 No.3 May 2022

Journal of Computer Science & Technology

SPONSORED BY INSTITUTE OF COMPUTING TECHNOLOGY
THE CHINESE ACADEMY OF SCIENCES &

CHINA COMPUTER FEDERATION

SUPPORTED BY NSFC

CO-PUBLISHED BY SCIENCE PRESS &

SPRINGER

Cao RY, Cao YX, Zhou GB *et al.* Extracting variable-depth logical document hierarchy from long documents: Method, evaluation, and application. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(3): 699–718 May 2022. DOI 10.1007/s11390-021-1076-7

Extracting Variable-Depth Logical Document Hierarchy from Long Documents: Method, Evaluation, and Application

Rong-Yu Cao[1,2] (曹荣禹), *Student Member*, *CCF*, Yi-Xuan Cao[1,2] (曹逸轩), *Member*, *CCF*, *IEEE*
Gan-Bin Zhou[3] (周干斌), and Ping Luo[1,2,4] (罗　平), *Senior Member*, *CCF*, *Member*, *IEEE*

[1] *Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences, Institute of Computing Technology Chinese Academy of Sciences, Beijing 100190, China*

[2] *University of Chinese Academy of Sciences, Beijing 100049, China*

[3] *WeChat Search Application Department, Tencent Holdings Ltd., Beijing 100080, China*

[4] *Peng Cheng Laboratory, Shenzhen 518066, China*

E-mail: {caorongyu19b, caoyixuan}@ict.ac.cn; ganbinzhou@tencent.com; luop@ict.ac.cn

Received October 16, 2020; accepted May 9, 2021.

Abstract In this paper, we study the problem of extracting variable-depth "logical document hierarchy" from long documents, namely organizing the recognized "physical document objects" into hierarchical structures. The discovery of logical document hierarchy is the vital step to support many downstream applications (e.g., passage-based retrieval and high-quality information extraction). However, long documents, containing hundreds or even thousands of pages and a variable-depth hierarchy, challenge the existing methods. To address these challenges, we develop a framework, namely Hierarchy Extraction from Long Document (HELD), where we "sequentially" insert each physical object at the proper position on the current tree. Determining whether each possible position is proper or not can be formulated as a binary classification problem. To further improve its effectiveness and efficiency, we study the design variants in HELD, including traversal orders of the insertion positions, heading extraction explicitly or implicitly, tolerance to insertion errors in predecessor steps, and so on. As for evaluations, we find that previous studies ignore the error that the depth of a node is correct while its path to the root is wrong. Since such mistakes may worsen the downstream applications seriously, a new measure is developed for a more careful evaluation. The empirical experiments based on thousands of long documents from Chinese financial market, English financial market and English scientific publication show that the HELD model with the "root-to-leaf" traversal order and explicit heading extraction is the best choice to achieve the tradeoff between effectiveness and efficiency with the accuracy of 0.9726, 0.7291 and 0.9578 in the Chinese financial, English financial and arXiv datasets, respectively. Finally, we show that the logical document hierarchy can be employed to significantly improve the performance of the downstream passage retrieval task. In summary, we conduct a systematic study on this task in terms of methods, evaluations, and applications.

Keywords logical document hierarchy, long document, passage retrieval

1 Introduction

Recently, the amount of electronic documents has increased rapidly along with the IT penetration into various vertical domains, such as financial, legal, government and education fields. To gain valuable insights from these unstructured documents, it is of the highest importance to obtain their underlying document structures so that these documents can be reedited, restyled, or reflowed to support many downstream natural language processing (NLP) and text mining applications. However, the transformation from the editing formats (e.g., WORD and LaTeX) of these documents to their display formats (e.g., PDF and JPG) only guarantees the appropriateness of document layout, while their underlying physical and logical structures are either par-

Regular Paper

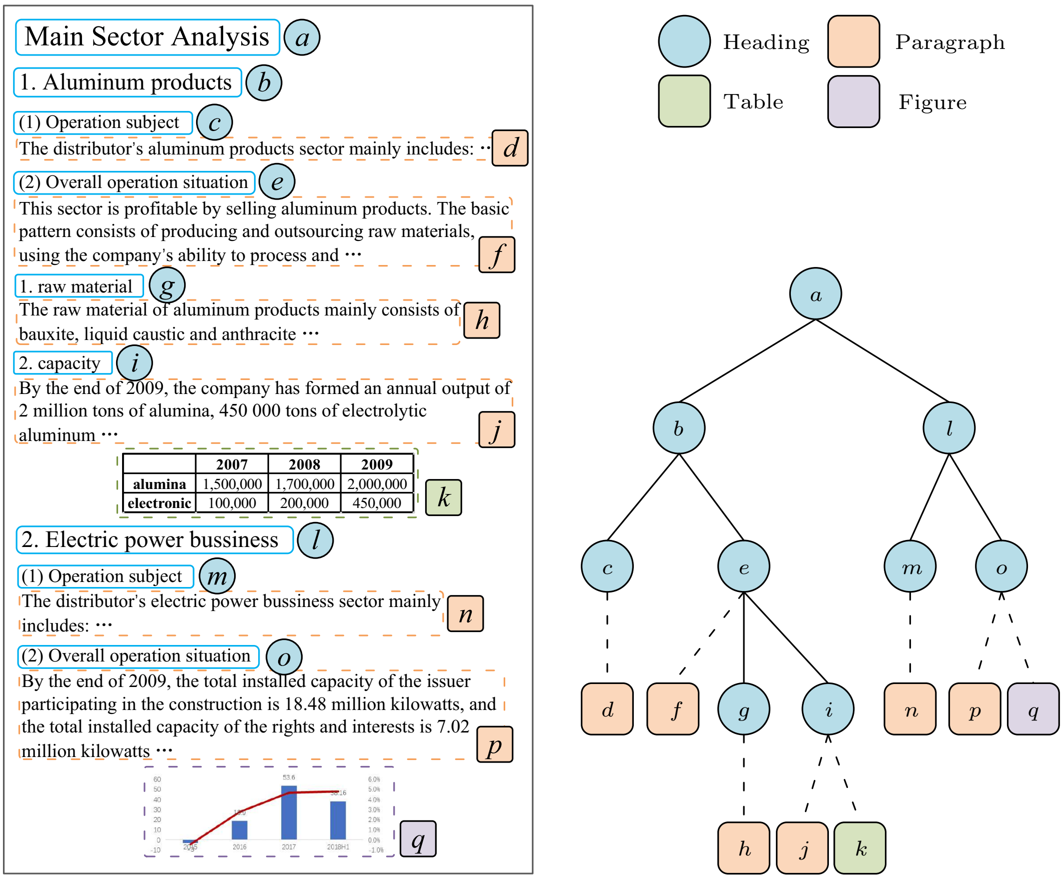
This work was supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002104, and the National Natural Science Foundation of China under Grant Nos. 62076231 and U1811461.

tially or even completely lost[1]. Hence, it is still an open issue to make this transformation reversible generally.

To this end, in this paper, we study the problem of extracting the variable-depth logical document hierarchy from long documents, which aims to organize the recognized "physical document objects" into hierarchical structures. A typical example of this task with a one-page document and its logical document hierarchy is shown in Fig.1. Here, physical objects refer to paragraphs, tables, charts and figures in a document[2]. We assume that a predecessor step detects these objects and ranks them by the reading order already. The goal of this study is to transform the flat structure of these physical objects into a hierarchical structure, which reflects the parallel and containment relationship between these physical objects. The discovery of logical document hierarchy helps to support many downstream applications such as hierarchical browsing, passage-based retrieval, high-quality information extraction and reading comprehension[3–6].

Although the recovery of logical document hierarchy attracts extensive researches[7–12], most studies focus on scientific papers or web pages, where only tens of pages are contained and the logical hierarchy is often fixed and shallow (four levels at most). Recently, millions of disclosure documents in the financial area from different countries have been published every year. However, these documents, such as annual reports, prospectuses, usually have hundreds of pages, and their hierarchies are much deeper with variable depth. Based on the thousands of benchmark documents with their annotated hierarchies, Fig.2 shows the distribution of physical object numbers and the distribution of headings on each depth. We observe that all the documents have at least 500 physical objects, 90% of headings locate on the 3rd–7th level of the trees, and the maximal depth is 11.

Such a variable-depth logical hierarchy from a long document challenges the existing methods on logical



Main Sector Analysis

1. Aluminum products

(1) Operation subject

The distributor's aluminum products sector mainly includes: ⋯

(2) Overall operation situation

This sector is profitable by selling aluminum products. The basic pattern consists of producing and outsourcing raw materials, using the company's ability to process and ⋯

1. raw material

The raw material of aluminum products mainly consists of bauxite, liquid caustic and anthracite ⋯

2. capacity

By the end of 2009, the company has formed an annual output of 2 million tons of alumina, 450 000 tons of electrolytic aluminum ⋯

	2007	2008	2009
alumina	1,500,000	1,700,000	2,000,000
electronic	100,000	200,000	450,000

2. Electric power bussiness

(1) Operation subject

The distributor's electric power bussiness sector mainly includes: ⋯

(2) Overall operation situation

By the end of 2009, the total installed capacity of the issuer participating in the construction is 18.48 million kilowatts, and the total installed capacity of the rights and interests is 7.02 million kilowatts ⋯

(a) (b)

Fig.1. Example about logical document hierarchy discovery. (a) Example document page and physical objects on this page. (b) Logical document hierarchy of this document page.

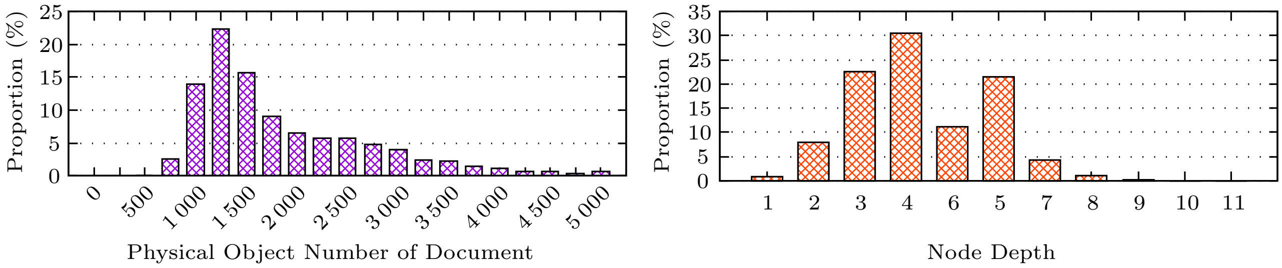


Fig.2. Distribution on the benchmark documents. (a) Distribution of physical object numbers. (b) Distribution of headings on each depth.

hierarchy recovery[8, 10–12]. Previous solutions can be grouped into three types. The first type[11, 12] formulates this task as a sequence labeling task, which employs Long Short Term Memory (LSTM) or Conditional Random Field (CRF) to extract contextual features of surrounding physical objects and classifies each heading into the absolute hierarchical depth. However, this type of methods fixes the space of depth labels and assigns an "absolute" depth to each physical object. In this study, we argue that since the hierarchical depth of physical objects depends on the containment and parallel relationship between contextual physical objects, the hierarchical depth should be considered as a "relative" concept rather than an "absolute" one. Additionally, due to extremely long distances among physical objects in the documents with hundreds of pages, sequence labeling based methods might not work well in capturing such long-distance context. The experimental results in this study also show that these methods obtain a lower accuracy on our benchmark documents. The second type of methods in [10, 13] is the rule-based one. They mostly propose some assumptions on logical hierarchy. For example, the study in [10] assumes that the headings with the same visual and textual style always locate at the same hierarchical depth. However, as illustrated by our benchmark documents, these assumptions are not always true. The third type is the hierarchy generation based method in [8]. It dynamically generates the logical hierarchy by considering the containment and parallel relationship between headings. However, this work lacks systematic studies on the possible variants of the generation process.

Inspired by how humans construct hierarchical trees in reading, we propose a novel model, namely Hierarchy Extraction from Long Document (HELD). Specifically, we sequentially insert each physical object at the proper position of the tree. By a certain traversal order, we inquire about all the possible insertion positions in the current tree until we find the proper one. Determining whether each possible position is proper or not can be formulated as a binary classification problem, namely the "put-or-skip" module. The hierarchical tree is generated until all the physical objects have been inserted. In this framework, the put-or-skip module is the key step. We propose an LSTM-based sub-model to detect the relative containment and parallel relationships between physical objects. The combination of both visual and textual features is adopted to capture the relationship between the local context of each insertion position and the physical object to be inserted. Furthermore, we study the design variants in HELD, including traversal orders of the insertion positions, heading extraction explicitly or implicitly, tolerance to insertion errors in predecessor steps, and so on.

As for evaluations, we find that previous studies ignore the error that the depth of a node is correct while its path to the root is wrong. Since such mistakes might seriously worsen the downstream applications, we propose a new measure, where an inserted node is correct if and only if the path from the root to itself is completely the same as the ground-truth path. We argue that this measure should be adopted in future studies of logical document hierarchy discovery.

Based on 1 030 Chinese documents from the financial domain (namely the Chinese dataset), 1 203 English documents from the financial domain (namely the English dataset) and 1 732 arXiv documents from the scientific domain (namely the arXiv dataset), we compare the proposed HELD model with the rule-based, sequence-tagging and existing generation-based methods. In the Chinese dataset, the HELD model achieves the best accuracy of 0.973 1, while the rule-based, sequence-tagging and existing generation-based methods obtain the accuracy of 0.376 4, 0.940 3 and 0.933 9 respectively. In the English dataset, the HELD model achieves the best accuracy of 0.730 1, while the three baseline methods obtain the accuracy of 0.477 9, 0.643 6 and 0.656 3, respectively. In the arXiv dataset, the HELD model achieves the best accuracy of 0.957 8, while the three baseline methods obtain the accuracy of 0.837 5, 0.890 8 and 0.903 4, respectively. To achieve the tradeoff between effectiveness and efficiency, the HELD model with the "root-to-leaf" traversal order and explicit heading extraction is the best choice with the accuracy of 0.972 6, 0.729 1 and 0.956 7 on Chinese, English and arXiv datasets, respectively, and 8.3x speedup in inference efficiency.

Finally, we demonstrate that the logical document hierarchy can be employed to significantly improve the performance of a downstream application, namely passage retrieval in a long document.

In summary, we conduct a systematic study on extracting variable-depth logical document hierarchy from long documents in terms of methods, evaluations, and applications.

This paper has the following organization. In Section 2, we review some related work. Details of the proposed HELD model are described in Section 3. Section 4 and Section 5 present the configurations and results of experiments respectively. Section 6 introduces a downstream application — passage-based retrieval. This paper ends with a summary and a brief discussion of future work in Section 7.

2 Related Work

2.1 Logical Document Hierarchy Extraction

The discovery of logical document hierarchy is a conventional task, and Summers[6] gave a proper definition: the logical structure consists of a hierarchy of segments of the document, each of which corresponds to a visually distinguished semantic component of the document. Generally, previous studies can be grouped into the rule-based method and the learning-based method.

For rule-based methods, Tsujimoto and Asada[14] aimed to discover the logical structure in multi-article newspapers, by using some generic transformation rules and a virtual field separator technique. Conway[13] used a set of grammar rules, each of which is a string of components specified by neighbor relation, and page parsing techniques to recognize document logical structure. Manabe and Tajima[10] proposed some assumptions, for instance, two headings with the same visual style should locate at the same significant level. Then, they sorted these headings by some visual styles (e.g., font size, bold or italic) and then generated a heading hierarchy.

Learning-based methods can be further separated into two classes, sequence labeling based and tree generation based methods. Some of sequence labeling based methods first recognize physical objects and determine the reading order of them, and then use different models to classify the absolute hierarchical depth of each physical object. For example, Luong *et al.*[7] used conditional random field (CRF), Rahman and Finin[11] used RNNs and Bentabet *et al.*[12] used LSTMs to classify physical into four categories: main-text, section-header, subsection-header and subsubsection-header. Other studies[15, 16] combined the rule-based and model-based method to extract the logical structure. For the tree generation based method, Pembe and Güngör[8] proposed a tree-based learning approach to generate the logical hierarchy node by node, by considering the containment and parallel relationship between nodes.

2.2 Physical Structure Recognition

Physical structure recognition is a basic step for extracting logical document hierarchy. It focuses on dividing the document into flat segmentations, rather than a hierarchy[17]. Here, flat segmentation represents an ordered list of physical objects (e.g., tables, paragraphs, figures[2]). Physical structure recognition can be categorized into top-down and bottom-up approaches.

The top-down approach[14, 18, 19] starts from the whole document and splits it into smaller components iteratively. Tsujimoto and Asada[14] divided the document page into some rectangle blocks and used simple rules to classify each block. Nagy *et al.*[19] proposed a vertical and horizontal cut-off-lines based method and Baird *et al.*[18] proposed a shape-directed-covers based method to recursively split the document page into smaller regions.

The bottom-up approach gathers pixels or characters into text lines and then combines them into physical objects. Early studies[13, 20] are grammar-based methods, which design different layout grammars to analyze the physical structure. Later studies consider the task as the semantic segmentation or sequence tagging problem. By regarding as the semantic segmentation problem, some studies[21, 22] detect the contour of each physical object (by length algorithm[23]) and classify pixels in it (by FCN, VGG[24, 25]) to determine the type of physical objects. By regarding as the sequence tagging problem, some studies[7, 26] split the document into an ordered list of text lines and determine the reading order of these text lines. Then, different methods (e.g., CRF, RNNs) are used to classify the type of each text line. Neighboring text lines with the same type will be grouped into a physical object.

3 Hierarchy Extraction from Long Document Model

Here, we introduce the Hierarchy Extraction from Long Document (HELD) model to convert an ordered list of physical objects (e.g., paragraphs, tables, charts and figures), namely $C = \{c_i\}_{i=1}^N$ where N is the total number of physical objects, into a hierarchical tree T. We assume that the list C is obtained in a predecessor step—physical structure recognition. Hence, after a careful evaluation, we adopt a commercial product, PDFLux[1] for this step. It can obtain physical structures and determine a reading order on various financial documents with a high accuracy, especially for disclosed financial documents.

3.1 Framework of Hierarchical Tree Generation Sub-Model

When a human reads a document, in his/her mind he/she actually constructs the logical document hierarchy gradually along the sequential reading process. When encountering a physical object in the document, he/she makes the decision on inserting this node into one of the possible positions in the current tree. Inspired by this human process, we propose the framework of Hierarchy Extraction from Long Document (HELD). Specifically, HELD sequentially checks each physical object in C one by one and inserts it into a proper position of the current tree. The key to this process is to clearly define all the possible insertion positions of the current tree. Before that, we first define the rightmost-branch of the current tree as follows.

Definition 1. *The rightmost-branch of the tree is an ordered list of nodes, where the first node is root ϕ, and each next node is the rightmost child of the previous node.*

For example, as shown in Fig.3, for the current tree its rightmost-branch is "ϕ, a, c, f", where each node is highlighted with a red circle.

With this rightmost-branch of the tree, we further define the possible insertion positions of the current tree as follows.

Definition 2. *For a new node to insert, its possible insertion positions are all the last children of the nodes in the rightmost-branch of the current tree.*

As shown in Fig. 3, there are four nodes in the rightmost-branch; thus there are four possible insertion positions: p_1, p_2, p_3, p_4 are the last children of ϕ, a, c, f, respectively.

The correctness of this definition of possible insertion positions is guaranteed by the following theoretical analysis. It is clear that the pre-order traverse of the document hierarchical tree generates the list of physi-

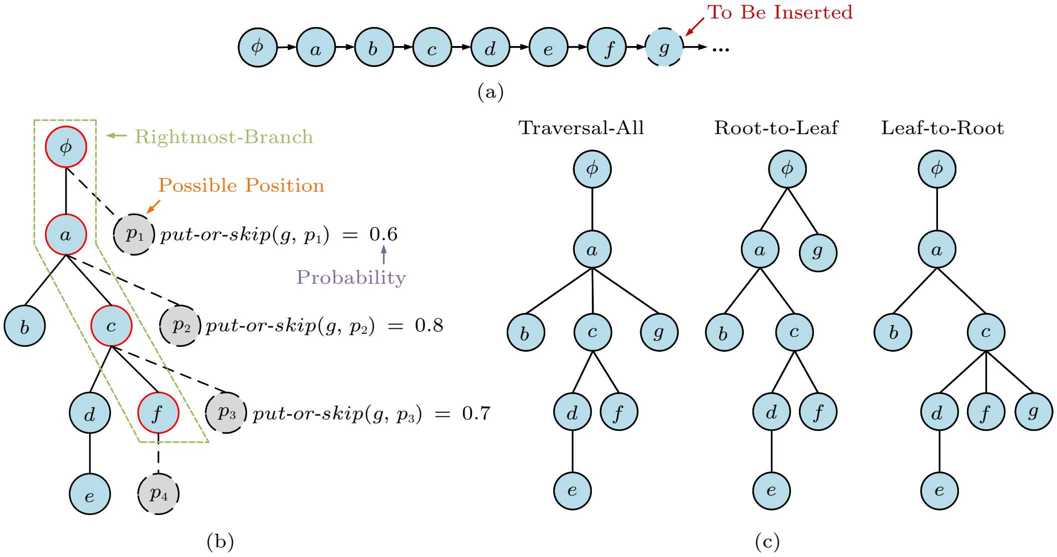


Fig.3. Example about tree generation. (a) Inserting physical object g, the rightmost-branch, is ϕ, a, c and f. (b) Hierarchical tree when inserting g. p_1, p_2, p_3 and p_4 are possible positions for g to be inserted and each has a probability. (c) Results after inserting g under three traversal methods.

[1] https://pdflux.com/, Apr. 2021.

cal objects in the reading order of the document. This definition can theoretically guarantee that the node to insert is always ranked at the last position of the preorder traverse of the tree after insertion only if it is inserted into any one of these possible positions.

With all these possible positions, we need a module to decide which position is proper for the current node. Next, Subsection 3.2 gives the training objective function of the whole framework, and Subsection 3.3 details the module of selecting the right insertion position.

3.2 Objective Function

For each document with the physical nodes of $\{c_i\}_{i=1}^{N}$, the joint probability (P) of the tree can be decomposed into the probability of each physical object c_i, in the condition that its previous physical objects have been already inserted. Specifically, it can be represented as follows,

$$\log P(\mathcal{T}) = \sum_{i=1}^{N} \log P(c_i|\mathcal{T}_{c_1,c_2,\cdots,c_{i-1}}),$$

where $\mathcal{T}_{c_1,c_2,\cdots,c_{i-1}}$ is the sub-tree constructed by the nodes of $c_1, c_2, \cdots, c_{i-1}$.

In the sub-tree $\mathcal{T}_{c_1,c_2,\cdots,c_{i-1}}$, we only consider all the possible insertion positions, denoted as $S_i = \{s_i^j\}_{j=1}^{M_i}$ where s_i^j is the j-th possible insertion position, and M_i represents the number of possible insertion positions. Among these positions, we denote s_i^* as the correct insertion position.

Then, $\log P(c_i|\mathcal{T}_{c_1,c_2,\cdots,c_{i-1}})$ can be further expanded as

$$\sum_{j=1}^{M_i} \left(\mathbb{1}\left(s_i^j = s_i^*\right) \times \log P\left(c_i \; |ctx(s_i^j)\right) - \mathbb{1}\left(s_i^j \neq s_i^*\right) \times \log P\left(c_i \; |ctx(s_i^j)\right)\right),$$

where $P\left(c_i \; |ctx(s_i^j)\right)$ stands for the probability of inserting node c_i into s_i^j and $ctx(s_i^j)$ represents the contextual information of position s_i^j. $\mathbb{1}(\cdot)$ is the indicator function. It equals 1 if the condition holds; otherwise it equals 0.

Then, for a corpus of L trees $\{\mathcal{T}_1, \cdots, \mathcal{T}_L\}$, we aim to maximize the following objective function

$$\sum_{k=1}^{L}\sum_{i=1}^{N_k}\sum_{j=1}^{M_i} \left(\mathbb{1}\left(s_i^j = s_i^*\right) \times \log P\left(c_i \; |ctx(s_i^j)\right) + \mathbb{1}\left(s_i^j \neq s_i^*\right) \times (1 - \log P)\right),$$

where N_k is the number of physical nodes in the k-th tree $\mathcal{T}_k$. In this objective function, we aim to maximize $P\left(c_i \; |ctx(s_i^j)\right)$ when s_i^j is the true position of c_i. Otherwise, we aim to minimize it.

With any annotated tree $\mathcal{T}$, it is easy to transform it into the labeled data for training $P\left(c_i \; |ctx(s_i^j)\right)$. Specifically, for each physical object c_i, we find out each possible insertion position s_i^j, and get all the corresponding tuples $(c_i, ctx(s_i^j), l_i^j)$, where l_i^j equals 1 if position s_i^j is the correct position of c_i; otherwise it equals 0. In this way, we can build a huge set of such tuples from all the annotated trees to train the parameters in $P\left(c_i \; |ctx(s_i^j)\right)$.

Note that all the training data are generated with the assumption that when inserting c_i, all the nodes $c_1, c_2, \cdots, c_{i-1}$ before c_i are all correctly inserted. However, this is not always true in the inference process of a new document. In Subsection 3.5.3, we will show how the training data can be enriched to be tolerant to some insertion errors in the predecessor steps.

3.3 Put-or-Skip Module

Next, we will present how the put-or-skip module is built. It aims to estimate the probability of inserting a physical object c into a possible insertion position s, denoted as $P(c|ctx(s))$. It can be regarded as a binary classification problem. Here, $ctx(s)$ refers to the contextual information of position s. As shown in Fig.4, this context may include the siblings of s, and its immediate parent. We observe that this local context provides vital cues for this classification. Specifically, if s is the right position for c, the siblings of s, namely $g_1, g_2, \cdots, g_K$, might have the same format features and consecutive item number with c, and its immediate parent z might have a more prominent format style than c. Thus, the module is required to consider all the textual and visual features inside the local context.

To capture the textual features, we use a Bi-LSTM[27] to model the text inside each physical object in the context. Specifically, it successively receives every word in the text of each physical object x, and outputs a fixed-length vector, namely $\boldsymbol{v}_x$. Thus, we can obtain the text representation of $c, g_1, \cdots, g_K$ and z, namely $\boldsymbol{v}_c, \boldsymbol{v}_{g_1}, \cdots, \boldsymbol{v}_{g_K}$ and $\boldsymbol{v}_z$, respectively. In order to combine these text representations and extract relationships among these representations, we use another Bi-LSTM to calculate a final text representation $\boldsymbol{v}$. To capture the visual features, for each physical object x we integrate all its format information, including font family, font size, font color, bold, italic, centering and indent, into a vector $\boldsymbol{u}_x$. Thus, for the nodes of $c, g_1, \cdots, g_K$ and z, we get $\boldsymbol{u}_c, \boldsymbol{u}_{g_1}, \cdots, \boldsymbol{u}_{g_K}$ and $\boldsymbol{u}_z$, respectively. Similarly, we use the third Bi-LSTM to get a final visual representation $\boldsymbol{u}$. Next, we concatenate $\boldsymbol{v}$ and $\boldsymbol{u}$, and send the combination vector into a feed-forward networks to obtain a synthetic representation. Finally, we use a Sigmoid function to obtain the probability $\hat{P}(c|ctx(s))$.

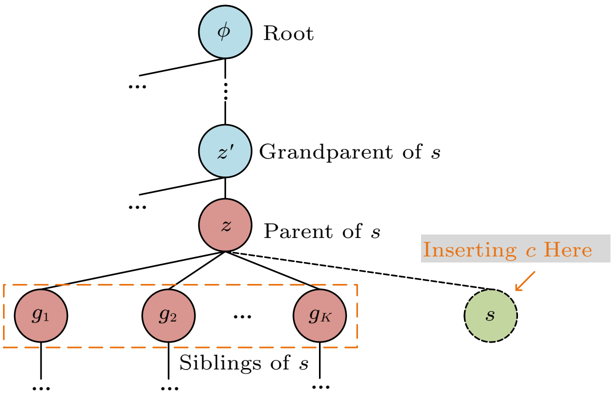


Fig.4. Context of determining whether physical object c is inserted into position s.

Readers may suggest that the context be expanded to consider all its other ancestors besides the immediate parent. However, this expansion definitely increases computational complexity.

3.4 Inference

For a new document, we aim to find out the optimal possible position sequence, $s_1^*, s_2^*, \cdots, s_N^*$ via maximizing the joint probability of inserting every physical object into a proper position. We have

$$(s_1^*, s_2^*, \cdots, s_N^*) = \underset{s_1, s_2, \cdots, s_N}{\arg\max} \sum_{i=1}^{N} \log P\left(c_i \mid ctx(s_i)\right),$$

where $s_i \in S_i$ and S_i represents the possible positions to insert c_i.

Note that searching the optimal path, s_1^*, s_2^*, $\cdots$, s_N^*, has exponential complexity, which makes the optimal result hard to search. The beam search is traditionally adopted for sequence or tree generation [28, 29], which considers multiple cases simultaneously in each step. In detail, we set a small integer bs as the beam size, which represents the number of candidate trees. At each step, we extend each candidate tree in the beam with the top bs most probable insertion positions. Thus, we obtain $bs \times bs$ candidate trees and remain the bs most probable candidate trees according to their joint probability. When all the physical objects are inserted, we select the final hierarchical tree with the highest joint probability. When we set $bs = 1$, the inference is the greedy method. The experiments in Section 5 will show that in different settings on bs the greedy method achieves the best tradeoff between effectiveness and efficiency.

3.5 Design Variants in HELD

3.5.1 *Traversal Orders of the Insertion Positions*

In the inference process, when inserting c_i we check all the possible insertion positions in a certain order. Specifically, we propose three traversal methods: traversal-all, root-to-leaf and leaf-to-root, and introduce them using the example in Fig.3.

First, for the traversal-all method, we inquire about each insertion position and find out the position with the highest probability. Next, for the root-to-leaf method, we inquire about each insertion position in the order from the root to the leaf node. Once we find a position with the probability of more than 0.5, we return it as the result. Finally, the leaf-to-root method is the same as the root-to-leaf method except that the traversal order changes from the leaf to the root node. For the example in Fig.3, the results from these three methods are p_2, p_1, and p_3, respectively. This example is deliberately generated to show the difference among them.

Note that different traversal methods involve different numbers of checks on the insertion positions. The theoretical analysis in Subsection 3.6 shows that to reach the proper insertion position the leaf-to-root method and the traversal-all method check the smallest and the largest number of positions, respectively. The results in Section 5 further empirically validate this. Additionally, the experimental results also show that the traversal-all and root-to-leaf methods achieve a similar accuracy. Hence, the root-to-leaf method is the best choice.

3.5.2 *Heading Extraction Explicitly or Implicitly*

We observe that all the internal nodes in the hierarchical tree $\mathcal{T}$ correspond to the headings of logical sections and all the leaf nodes of $\mathcal{T}$ correspond to concrete objects (e.g., paragraphs, tables, and charts). Additionally, a section is usually semantically summarizable and visually observable by its heading [6]. Thus, another possible solution to our task is: 1) classifying each physical object into heading or non-heading, 2) generating a hierarchical tree for all the heading nodes, 3) inserting each non-heading object as the leaf child of its first previous heading object in the input sequence of physical objects. In other words, this new solution suggests a separated step of explicit heading extraction before the hierarchical tree generation.

We find that a separated step of explicit heading extraction brings about some benefits to our task. First, it might alleviate the difficulty of classification in the put-or-skip module, seeing the example in Fig.1 without a separated step of explicit heading extraction. Let us consider the insertion of node g, which is a heading. This node should be located at the same level as the node f, which is a non-heading object. For this situation that heading nodes and non-heading nodes are siblings, the model usually fails since heading and non-heading nodes usually have features of great differences. With an additional step of heading extraction, tree building is much easier for only the heading nodes. The experiments in Section 5 further validate that this two-step solution increases the model accuracy.

Second, although heading extraction introduces additional computing overhead, the following computing of node insertion will greatly decrease. Specifically, the model is applied to the heading nodes for the tree generation, and the other non-heading nodes are inserted by the rule. Overall, this two-step solution gains much increase in time efficiency, which is also demonstrated by the experiments.

Note that distinguishing heading nodes from non-heading ones can be formulated as a sequence labeling task. In detail, we use Bi-LSTM [27] to extract textual and format features of the local context and then apply multi-layers CNNs [24] to consider the long-distance association. Inspired by the previous work [30], we add a self-attention layer [31]. Finally, a Sigmoid layer is used to classify whether a physical object is heading or not.

3.5.3 Tolerance to Insertion Errors in Predecessor Steps

Note that in the actual inference for a document, before inserting a node c_i some previous nodes in $\{c_1, , \cdots, c_{i-1}\}$ might be inserted into wrong positions. Thus, we need to deliberately make some training data with such insertion errors. Specifically, we simulate the tree-building process with some random insertion errors, where a node is inserted into one of the other possible positions except the right one. Based on the resultant tree, some extra training data can be generated accordingly. The experimental results show that the new training data bring about significant improvements in terms of effectiveness.

3.6 Theoretical Analysis on the Efficiency of Different Traversal Methods

In the following, for each traversal method, we will theoretically count the number of checks on the insertion positions, which is required to reach the ground-truth position. Note that in this analysis we assume that the ground-truth position for inserting each node is provided. This number is equal to the one when an ideal model with the 100% accuracy is adopted in inference.

Specifically, after estimating $P(c|ctx(s))$ the parent node of s will be counted once. To sum up the numbers on all the nodes, we get the final total, denoted as $N_{\rm all}, N_{\rm r2l}$ and $N_{\rm l2r}$ for the three methods of traversal-all, root-to-leaf and leaf-to-root respectively. We have

$$\begin{aligned} N_{\rm all} &= \sum_{i=1}^{N}(s_i + \mathbb{1}(\text{node } i \text{ is not on the} \\ &\quad \text{rightmost-branch})), \\ N_{\rm r2l} &= \sum_{i=1}^{N}(s_i + \mathbb{1}(\text{node } i \text{ has next sibling})), \\ N_{\rm l2r} &= \sum_{i=1}^{N}(s'_i + \mathbb{1}(\text{node } i \text{ is not on the} \\ &\quad \text{rightmost-branch})), \end{aligned}$$

where s_i and s'_i refer to the number of all the descendants and non-leaf descendants of a node i in the tree respectively, N refers to the number of nodes in the tree, and $\mathbb{1}(\cdot)$ is the indicator function. Fig. 5 shows an example with the inquiry numbers for these three methods.

Next, we calculate the relationships between these numbers as follows,

$$\begin{aligned} &N_{\rm all} - N_{\rm r2l} = I - l \gg 0, \\ &N_{\rm all} - N_{\rm l2r} = \sum_{i=1}^{N}(s_i - s'_i) \gg L \gg I > N_{\rm all} - N_{\rm r2l}, \end{aligned}$$

where I and L is the number of the internal node and the leaf node in the tree respectively, and l is the length of its rightmost-branch.

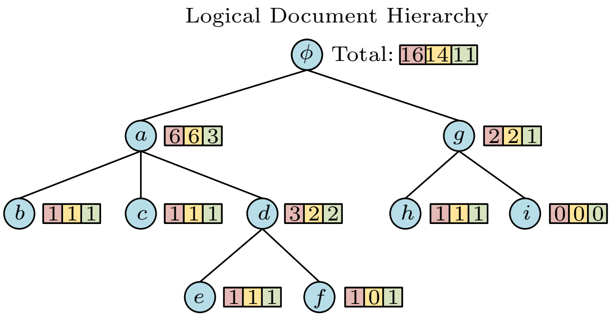


Fig.5. Inquiry number in hierarchy generation. The digits in the boxes represent the total inquiry number of each node under different traversal methods. Specifically, red, yellow and green boxes represent the traversal-all, root-to-leaf and leaf-to-root method, respectively.

Usually, we have $L \gg I \gg l$. This is also true for all the hierarchical trees used in this study. Thus, we have $N_{\mathrm{l2r}} < N_{\mathrm{r2l}} < N_{\mathrm{all}}$.

Finally, we argue that although these three numbers might not equal the inquiry numbers with the actual model when its accuracy is less than 100%, they are a good approximation of the actual number. This is also demonstrated in the experiments in Section 5.

4 Experiment Details

4.1 Dataset

Since the documents in published datasets for our task [8, 10–12] only contain tens of pages and have a shallow (four levels at the most) logical hierarchy, we build three datasets with a variable-depth logical hierarchy from long documents: 1) the Chinese dataset that contains prospectuses and annual reports from the China Securities Exchange market, 2) the English dataset that contains annual reports from Hong Kong Stock Exchange market, and 3) the arXiv dataset that contains English scientific publications from arXiv. The documents in the Chinese and English dataset can be downloaded from CNINFO[2] and the documents in the arXiv dataset can be downloaded from arXiv[3].

Each document is assigned to at least two annotators for annotating its logical document hierarchy. If the results on a document are different, another senior annotator will address the conflicts and output the final answer.

For three datasets, we split the training, validation and test set with the ratio of around 8:1:1. For the Chinese dataset, we split 1 030 documents into 830 for training, 100 for validation and 100 for test. For the English dataset, we split 1 110 documents into 910 for training, 100 for validation and 100 for test. For the arXiv dataset, we split 1 732 documents into 1 432 for training, 150 for validation and 150 for test.

4.2 Dataset Analysis

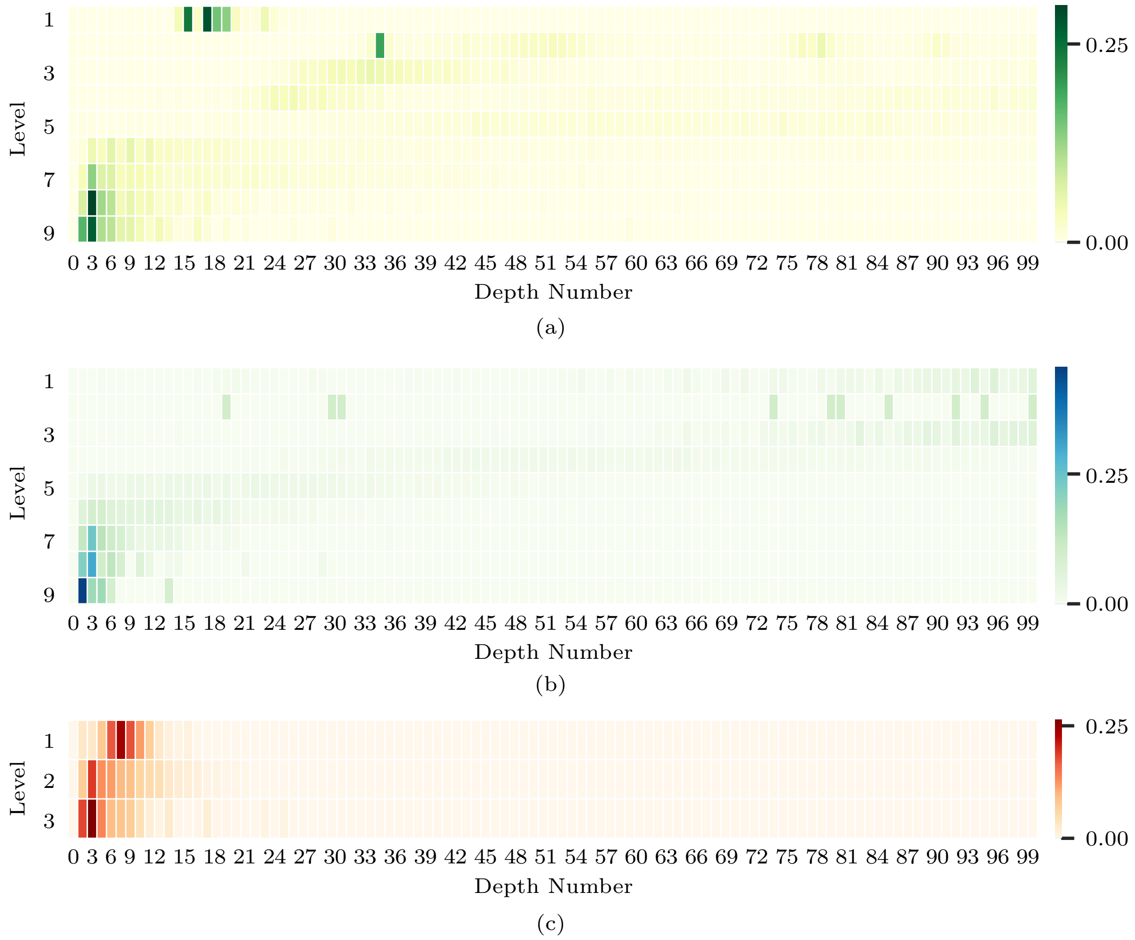
In this subsection, we further analyze these three datasets and illustrate some observations in Fig.6 and Fig.7.

On the one hand, we calculate the number of internal nodes at each level in each document and depict the distribution in Fig.6(a), Fig.6(b) and Fig.6(c) for the three datasets, respectively. For example, in the Chinese dataset, we find that most documents have 10–20 headings on level 1. However, for levels 3–7, most documents have a different number of headings; therefore the distribution is very uniform. In the English dataset, for every level, most documents have a different number of headings, which means that the degree of difference in the English dataset is greater than that in the Chinese dataset. In the arXiv dataset, the number of headings at each depth is relatively concentrated, since the logical hierarchies in scientific publications are usually pre-defined.

On the other hand, we aim to observe whether these documents follow some types of templates. First, we conclude 44 types of patterns to represent the item number in headings. For example, headings c, e, m, o in Fig.1 follow the pattern of "(1), (2), $\cdots$". Headings b, h, i, l in Fig.1 follow the pattern of "1., 2., $\cdots$". For each type of patterns, we can design one regex to match it; thus, we design 44 regexes to represent these patterns. Then, for a given document hierarchy, we can check each internal node in this hierarchy and decide whether it matches one of the 44 regexes. Thus, for the internal nodes on each level in all the documents in the dataset, we count the matching ratio of these 44 regexes and draw the distribution graph for the three datasets in Fig.7(a), Fig.7(b) and Fig.7(c) respectively. For example in Fig.7(a), we observe that most internal nodes in the first level match regexes 0, 1 and 3. Here, regex 0 means the node matches none of the 44 regexes. Almost all the internal nodes in the second level match regex 7. Clearly, in some deeper levels, such as levels 5–9, these internal nodes match more various regexes. Especially, in the English dataset, most headings do not match any

[2] http://www.cninfo.com.cn/, Apr. 2021.

[3] https://arxiv.org/, Apr. 2021.



(a)

(b)

(c)

Fig.6. We calculate the distribution of the number of nodes at different depth in the Chinese, English and arXiv datasets. (a) Depth-number distribution on the Chinese dataset. (b) Depth-number distribution on the English dataset. (c) Depth-number distribution on the arXiv dataset.

regex. Therefore, the hierarchies in these documents have differences to some degree; therefore the hierarchy is not definitely pre-defined. Moreover, the degree of difference in the English dataset is greater than that in the Chinese dataset.

4.3 Evaluation Methods

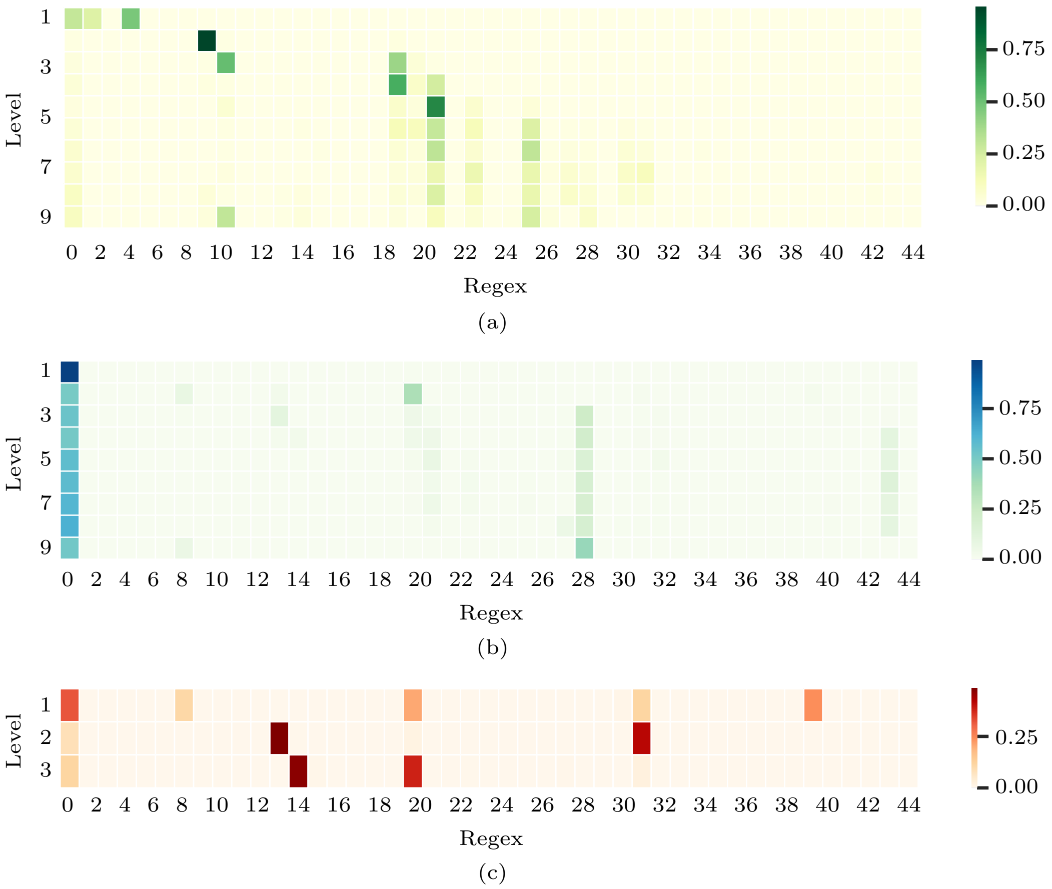
In this study, we propose a new metric to judge whether a certain node is inserted correctly or not. First, we define that physical object c_i is inserted correctly if its predicted path $\hat{r}_i$ completely equals its ground-truth path r_i. Here, the path of c_i means an ordered node list that consists of c_i and all its ancestors up to the root ϕ. Note that previous studies[7, 15, 16] define that node c_i is inserted correctly if it is put at the right level of the tree. They ignore the error that the depth of a node is correct while its path to the root is wrong. For example, in Fig.8 node g is considered as an error by the new measure since its path to the root, namely ($g \rightarrow f \rightarrow a \rightarrow \phi$), is not equal to the ground-truth path ($g \rightarrow c \rightarrow a \rightarrow \phi$). However, it is considered as a correct insertion since it is put into the right level of 3.

With this new definition of correct insertion, we can calculate the overall accuracy of node insertion. Additionally, for any tree level k we can also calculate the precision, recall, and $F1$ denoted as p_k, r_k and $F1_k$, respectively. In the experimental results, we use these measures in micro average for all the testing documents.

To evaluate the efficiency in processing new documents, we record the average execution time of each document. Additionally, we count the number of calls for the put-or-skip module, denoted as "#inquiry", since they are the major time consumption.

4.4 Baselines

HEPS Model[10]. This is a rule-based method for our task. The original HEPS model focuses on web

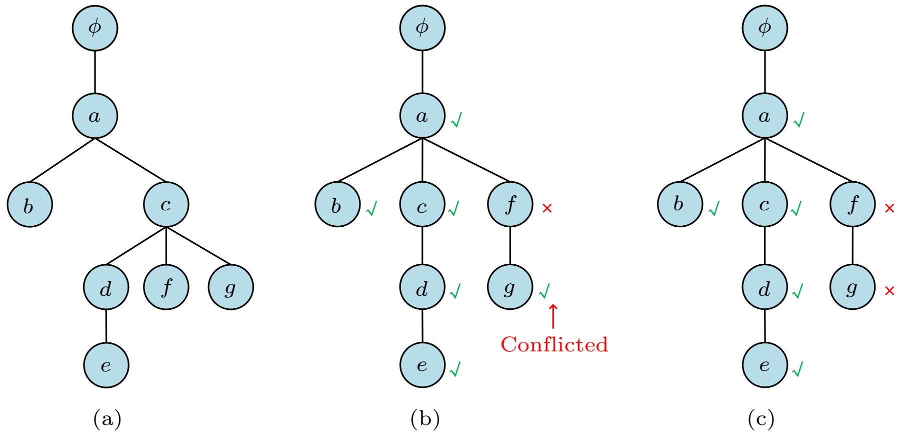


(a)

(b)

(c)

Fig. 7. We calculate the distribution of the number of matched key word-level features at different depth in the Chinese, English and arXiv datasets. (a) Depth-regex distribution on the Chinese dataset. (b) Depth-regex distribution on the English dataset. (c) Depth-regex distribution on the arXiv dataset.



(a)

(b)

(c)

Fig. 8. Comparison of old and new evaluation metrics. (a) Ground-truth hierarchy. (b) Predicted hierarchy and correctness of each node under the previous metric. (c) Predicted hierarchy and correctness of each node under our metric.

pages. Some of the features from web pages cannot be obtained in PDF files. Thus, this method is tailored to use only the features in PDF files.

TOC Model[12]. The TOC model[12] is a sequence labeling-based model. Since the document length is greatly longer than those in the TOC model, we use CNNs to replace LSTMs in sequence labeling to improve efficiency.

Pembe and Güngör's Model[8]. Pembe and Güngör's model is a tree generation-based model. In this method logistic regression is adopted to select the proper position for node insertions.

4.5 Hyper-Parameters Configuration

Here, we introduce some hyper-parameters of the proposed HELD model. First, we use skip-gram [32] to pre-train character embeddings with 24-dimension. In the heading recognition step, we use a 9-layer Network In the Network [33] to extract contextual features. The kernel size of each layer is 5, 1, 5, 5, 1, 5, 5, 1, 5. The kernel number of each layer is 128, 64, 128, 256, 128, 256, 512, 256, 512 respectively. We adopt the batch normalization [34] immediately after each convolution and before all ReLU activate functions [35]. In the put-or-skip model (according to Subsection 3.3), we set the hidden dimension as 128, 512 and 64 for v_x, v_T and u_F, respectively. We use the "tanh" activate function in LSTMs. We set the hidden dimension as 128 for the FNN layer. Then, weight initialization in [36] is used to initialize parameters and the Adam [37] optimizer is used to update parameters. We set the mini-batch size as 128 and the learning rate as 0.000 05. We train the model on two Titan 1080Ti GPUs and use Horovod [38] to update parameters in a distributed way.

5 Experimental Results

5.1 Results

In this subsection, we aim to answer these research questions.

- *RQ*1. What is the effectiveness of the HELD model compared with other baselines?
- *RQ*2. What is the effectiveness and efficiency of different traversal methods in the HELD model?
- *RQ*3. What is the effectiveness and efficiency of the one-step and the two-step framework?
- *RQ*4. What is the effectiveness of adding tolerance to insertion errors in predecessor steps?
- *RQ*5. What is the effectiveness of different features in the put-or-skip module?
- *RQ*6. What is the effectiveness and efficiency of the beam size?
- *RQ*7. What is the influence of noise in document layout recognition?

For RQ1, we compare the proposed two-step HELD model with three baseline models, by evaluating $F1_k$ on the test set of the three datasets. The results are shown in Table 1, Table 2 and Table 3, respectively.

Note that we extract headings explicitly (two-step), use the traversal-all method and set the beam size as 1 in the inference process to obtain the best HELD model. The proposed HELD obtains the node accuracy of 0.973 1, 0.730 1 and 0.957 8 on each dataset, respectively. However, the HEPS model [10] obtains 0.376 4, 0.477 9, 0.837 5 node accuracy, the TOC model [12] obtains 0.940 3, 0.643 6, 0.890 8 node accuracy and the Pembe and Güngör's model [8] obtains 0.933 9, 0.656 3, 0.903 4 node accuracy, respectively. Clearly, on each

Table 1. Comparing HELD and Baseline Models on the Test Set of the Chinese Dataset

Model	*acc*	$F1_1$	$F1_2$	$F1_3$	$F1_4$	$F1_5$	$F1_6$	$F1_7$	$F1_8$	$F1_9$	$F1_{10}$	$F1_{11}$
HEPS [10]	0.376 4	0.803 0	0.679 4	0.491 3	0.478 1	0.292 6	0.128 7	0.013 8	0.000 0	0.000 0	0.000 0	0.000 0
TOC [12]	0.940 3	0.980 4	0.940 8	0.956 2	0.963 2	0.952 8	0.933 3	0.851 2	0.630 5	0.264 5	0.000 0	0.000 0
Pembe and Güngör's [8]	0.933 9	0.957 3	0.944 5	0.967 4	0.966 6	0.950 1	0.930 5	0.862 7	0.694 4	0.441 8	0.062 9	0.000 0
1step-HELD	0.958 3	0.991 8	0.957 7	0.974 8	0.969 5	0.961 7	0.948 6	0.917 9	0.869 3	0.681 0	0.462 0	0.248 3
2step-HELD (l2r)	0.972 0	0.989 2	0.948 3	0.976 9	0.983 1	0.976 1	0.966 2	0.947 1	0.914 5	0.829 9	0.849 1	0.705 9
2step-HELD (r2l)	0.972 6	0.989 2	0.948 6	0.977 9	0.983 2	0.977 1	0.967 0	0.945 4	0.917 9	0.853 1	0.795 0	0.734 7
2step-HELD (ta)	0.973 1	0.989 2	0.948 9	0.978 4	0.983 8	0.977 8	0.967 5	0.944 9	0.917 4	0.852 1	0.851 7	0.705 9
− Tolerance Errors	0.972 5	0.989 2	0.948 1	0.976 9	0.983 3	0.976 8	0.967 1	0.948 1	0.913 8	0.850 1	0.851 7	0.705 9

Note: − means removing.

Table 2. Comparing HELD and Baseline Models on the Test Set of the English Dataset

Model	*acc*	$F1_1$	$F1_2$	$F1_3$	$F1_4$	$F1_5$	$F1_6$	$F1_7$	$F1_8$	$F1_9$
HEPS [10]	0.477 9	0.688 2	0.663 0	0.626 5	0.487 1	0.297 6	0.174 4	0.052 5	0.018 7	0.000 0
TOC [12]	0.643 6	0.829 6	0.792 8	0.778 5	0.619 0	0.367 1	0.211 1	0.077 6	0.000 0	0.000 0
Pembe and Güngör's [8]	0.656 3	0.806 2	0.796 4	0.771 6	0.646 0	0.436 4	0.328 6	0.233 7	0.060 2	0.000 0
1step-HELD	0.611 7	0.700 8	0.686 5	0.705 3	0.598 4	0.483 0	0.359 1	0.260 0	0.048 5	0.000 0
2step-HELD (l2r)	0.707 5	0.855 5	0.836 5	0.811 9	0.702 6	0.557 7	0.420 0	0.313 6	0.329 6	0.398 2
2step-HELD (r2l)	0.729 1	0.826 4	0.825 0	0.813 3	0.719 6	0.602 2	0.465 8	0.473 8	0.488 3	0.532 5
2step-HELD (ta)	0.730 1	0.855 6	0.838 0	0.818 0	0.721 1	0.604 8	0.480 7	0.376 3	0.425 8	0.424 5
− Tolerance Errors	0.709 5	0.832 4	0.812 0	0.794 2	0.695 5	0.586 7	0.464 6	0.446 9	0.368 6	0.371 9

Note: − means removing.

Table 3. Comparing HELD and Baseline Models on the Test Set of the arXiv Dataset

Model	*acc*	$F1_1$	$F1_2$	$F1_3$	$F1_4$
HEPS [10]	0.837 5	0.938 5	0.897 5	0.596 3	0.321 8
TOC [12]	0.890 8	0.980 0	0.930 1	0.763 4	0.582 8
Pembe and Güngör's [8]	0.903 4	0.973 4	0.924 3	0.823 6	0.684 2
2step-HELD (l2r)	0.954 6	0.992 3	0.970 5	0.903 2	0.706 9
2step-HELD (r2l)	0.956 7	0.992 6	0.972 0	0.907 3	0.706 9
2step-HELD (ta)	0.957 8	0.992 6	0.973 0	0.910 4	0.717 8

dataset, the proposed HELD model has great improvement on the $F1$ value of every level ($F1_k$) and the total node accuracy (acc) compared with three baseline models. Note that, compared with the other two datasets, all the models obtain a lower accuracy on the English dataset, since the visual and the textual cues are more implicit in this dataset. Like the intuitive analysis in Section 1, the rule-based model (the HEPS model) obtains a low accuracy since the assumptions in this model are not always true. Sequence labeling based model (the TOC model) cannot predict well for the physical objects on deep levels since it considers the hierarchical depth as an absolute concept and neglects the containment and parallel relation between physical objects. As shown in Fig.7(a), Fig.7(b) and Fig.7(c), headings at the same level match different regexes and headings that match the same regex locate at different levels. That is to say, it is hard to directly predict the level of each heading, especially for the headings on deeper levels. Therefore, the TOC model obtains a lower accuracy, especially for the headings on deeper levels. Pembe and Güngör's model is based on the hierarchy generation; however, it is not good at extracting format and semantic features; thus it underperforms the proposed HELD model. Especially on the English dataset, many headings match no regex as shown in Fig.7(b); thus predicting the level of these headings depends on both format and semantic features. Therefore, Pembe and Güngör's model obtains a low accuracy on the English dataset. By combining format and semantic features to extract containment and parallel relation between physical objects, the proposed HELD model outperforms the other baselines and obtains the node accuracy of 0.973 1 and 0.730 1 on Chinese and English datasets, respectively.

For RQ2, we compare three different traversal methods in the HELD model, by evaluating $F1_k$ and #inquiry on the test set of the three datasets. The results are shown in Fig.9(a), Table 1, Table 2 and Table 3, respectively.

The traversal-all method obtains the accuracy of 0.973 1, 0.730 1 and 0.957 8 on Chinese, English and arXiv datasets, respectively. The root-to-leaf method obtains the accuracy of 0.972 6, 0.729 1 and 0.956 7 on Chinese, English and arXiv datasets, respectively. Apparently, the traversal-all method outperforms the root-to-leaf method, since the traversal-all method inquiries all the possible insertion positions; however, the gap between them is subtle. Note that the leaf-to-root method obtains a lower accuracy than other traversal methods. Because the root-to-leaf method assigns a lower priority for those physical objects on the shallow level, these nodes on the shallow level have higher importance than the other nodes. The reason is that if the parent node is inserted into incorrect positions, all of its descendant nodes will be wrong under our evaluation measure.

To explore the efficiency of three traversal methods, we count the processing time for an average document and #inquiry (the number of inquiries) in Fig.9(a). The traversal-all method consumes 42.58 seconds and 2 168.35 inquiries on average to process a document. For comparison, the root-to-leaf method consumes 36.00 seconds and 2 023.05 inquiries to process a document on average (obtaining 1.2x speedup ratio) and the leaf-to-root method only consumes 17.64 seconds and 1 016.5 inquiries to process a document on average (obtaining 2.4x speedup ratio). Note that, the efficiency order of three traversal methods in practice is "leaf-to-root > root-to-leaf > traversal-all", which empirically validates the efficiency order in theory (shown in Subsection 3.6).

Therefore, we can use the traversal-all method if a higher accuracy is required, and use the root-to-leaf method if a higher efficiency is required. Since the leaf-to-root method obtains great improvement on efficiency, it can be used if the requirement of efficiency is much higher than that of effectiveness.

For RQ3, we compare one-step HELD (1step-HELD) with two-step HELD (2step-HELD), by evaluating $F1_k$ and #inquiry on the test set of three datasets. The results are shown in Fig.9(b), Table 1, Table 2 and Table 3, respectively.

Apparently, the best 2step-HELD model obtains the node accuracy of 0.973 1 and 0.730 1 on the Chinese and the English dataset, respectively. By comparison,

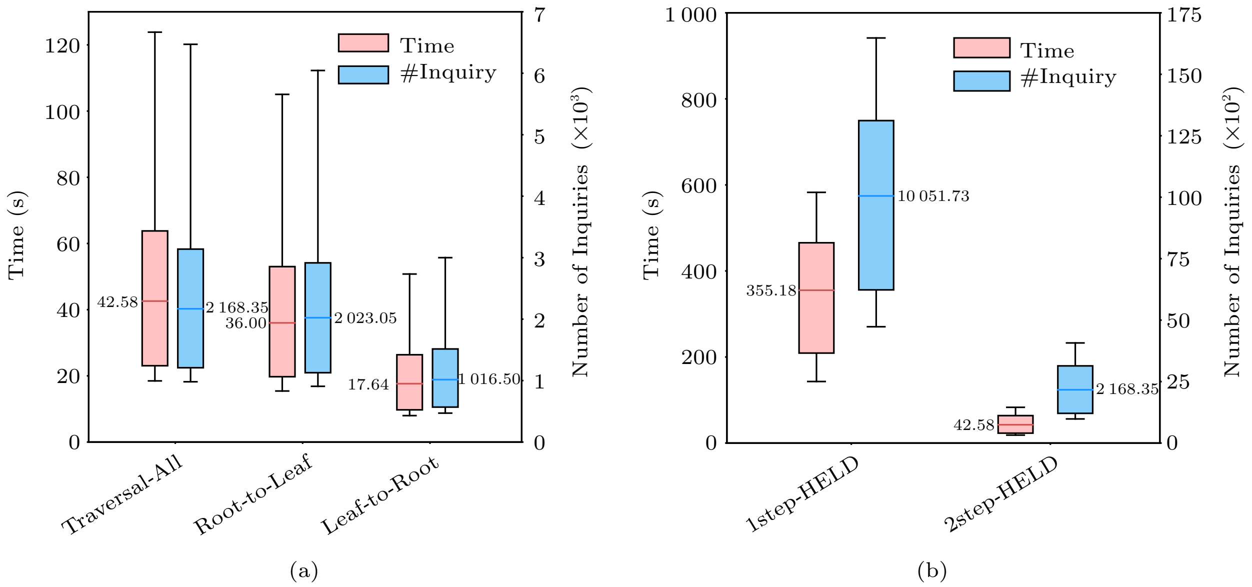


Fig.9. Results on efficiency of different models. (a) Comparing different traversal methods in the HELD model. (b) Comparing the 1step-HELD model and the 2step-HELD model.

the 1step-HELD model obtains the node accuracy of 0.958 3 and 0.611 7 on the Chinese and the English dataset, respectively. In other words, the 2step-HELD model greatly outperforms the 1step-HELD model on effectiveness. The reason is that generating hierarchy based on heading objects sequence alleviates the difficulty of classification in the put-or-skip module (according to analysis in Subsection 3.5.2). Note that, the 1step-HELD model obtains higher $F1_1$ and $F1_2$ than the 2step-HELD model on the Chinese dataset, because the 2step-HELD model might make some classification mistakes in the heading recognition step. In general, extracting heading objects explicitly leads to great improvement on effectiveness.

From the view of efficiency, as shown in Fig.9(b), the 2step-HELD model consumes less execution time and demands fewer inquiries. In our dataset, we observe that there are average 590 heading objects and average 2 300 physical objects in a document. Thus, the 2step-HELD model reduces around 75% nodes for generating the hierarchy, which causes the average number of inquiries to reduce by around 78%. The 1step-HELD model consumes 355.18 seconds to process a document on average. In the 2step-HELD model, the heading recognition step consumes 1.2 seconds (2.8% execution time) and the heading hierarchy generation step consumes 41.38 seconds (97.2% execution time) on average, which means that the major time consumption comes from the heading hierarchy generation step. Thus, the 2step-HELD model obtains around 8.3x speedup ratio (from 355.18 s to 42.58 s) in general compared with the 1step-HELD model.

In summary, to obtain higher effectiveness and efficiency, we extract headings explicitly in the HELD model.

For RQ4, we explore the tolerance to insertion errors of predecessor steps in the HELD model, by evaluating $F1_k$ on the test set of the Chinese and English datasets. The results are shown in Table 1, Table 2 and Table 3, respectively.

For comparison, we choose the best HELD model, which is 2step-HELD with the traversal-all method as the baseline. Note that the tolerance to insertion errors of predecessor steps is used in this model. Then, based on the best HELD model, we remove the tolerance to insertion errors of predecessor steps and obtain the last row in Table 1 and Table 2. Clearly, after removing the tolerance errors module, the HELD model obtains the node accuracy of 0.972 5 and 0.709 5 on the Chinese and the English dataset, respectively. The results show that it obtains 0.000 6 and 0.020 6 decrease in the node accuracy on the Chinese and the English dataset, respectively. In other words, the tolerance to insertion errors obtains greater improvement on the English dataset, since there exist more insertion errors and adding the tolerance to insertion errors can make the HELD model insert nodes correctly based on some insertion errors of predecessor steps.

In summary, to obtain a higher effectiveness, we add the tolerance to insertion errors of predecessor steps.

For RQ5, we design an ablation experiment to show the importance of contextual features in the HELD model. Note that, according to Subsection 3.3, we consider that contextual features contain the previous siblings and the parent of the current node. For comparison, we choose the best HELD model, which is the 2step-HELD with the traversal-all method, as the baseline. Then, we remove the features of previous siblings (− Siblings) and the parent (− Parent) in the put-or-skip module respectively to present the importance of another one. The experimental results are shown in Table 4 and Table 5. Clearly, removing parent features obtains the node accuracy of 0.965 9 and 0.700 7 on the Chinese and the English dataset, respectively, with the decrease in the node accuracy of 0.007 2 and 0.029 4, respectively. On the other hand, removing previous siblings' features obtains the node accuracy of 0.940 2 and 0.644 5 on the Chinese and English datasets, respectively, with the decrease in the node accuracy of 0.032 9 and 0.085 6, respectively. In other words, adding previous siblings' features can obtain prominent improvement in effectiveness for the put-or-skip module, since previous siblings often have the same format features and consecutive item numbers with the current node. Since the previous siblings and the parent features both lead to improvement on effectiveness, we use both of them in the put-or-skip module.

For RQ6, the beam search is traditionally adopted for tree generation. In Subsection 3.4, we have introduced how to use the beam search in the proposed HELD model. Here, to explore the effectiveness and efficiency of using the beam search, we choose the best HELD model, which is the 2step-HELD with the traversal-all method, and set the beam size as 1 (using greedy search), as the baseline. Then, we set the beam size as 3 for comparison and the experimental results on the Chinese and English datasets are shown in Table 6 and Table 7, respectively. Clearly, setting the beam size as 3 obtains the node accuracy of 0.973 6 and 0.727 5 on Chinese and English datasets, respectively. In other words, setting beam size as 3 will not prominently improve the accuracy and even decrease the accuracy on the English dataset. The reason is that the put-or-skip module can distinguish different possible positions apparently in inference; thus it does not need to expend several possible positions in each search step.

Meanwhile, we also count the processing time and #inquiry for an average document compared with setting the beam size as 1 and 3. The results show that generating the logical hierarchy for each document consumes 42.58 seconds on average when setting the beam size as 1 and consumes 413.78 seconds on average when setting the beam size as 3. In other words, using beam search obtains about 10x decrease in efficiency.

In summary, since setting the beam size as 1 achieves the best tradeoff between effectiveness and efficiency, we use the greedy search (without beam search) in the HELD model.

For RQ7, we aim to explore the influence of noise in document layout recognition. As mentioned previously,

Table 4. Exploring Performance of Different Features in the Put-or-Skip Module on Test Set of the Chinese Dataset

Model	*acc*	$F1_1$	$F1_2$	$F1_3$	$F1_4$	$F1_5$	$F1_6$	$F1_7$	$F1_8$	$F1_9$	$F1_{10}$	$F1_{11}$
2step-HELD (ta)	0.973 1	0.989 2	0.948 9	0.978 4	0.983 8	0.977 8	0.967 5	0.944 9	0.917 4	0.852 1	0.851 7	0.705 9
− Parent	0.965 9	0.988 9	0.946 5	0.972 9	0.978 2	0.968 6	0.956 3	0.941 0	0.907 5	0.848 1	0.908 4	1.000 0
− Siblings	0.940 2	0.958 1	0.913 1	0.949 2	0.956 4	0.944 8	0.933 4	0.905 1	0.847 1	0.690 3	0.657 5	0.378 9

Table 5. Exploring Performance of Different Features in the Put-or-Skip Module on Test Set of the English Dataset

Model	*acc*	$F1_1$	$F1_2$	$F1_3$	$F1_4$	$F1_5$	$F1_6$	$F1_7$	$F1_8$	$F1_9$
2step-HELD (ta)	0.730 1	0.855 6	0.838 0	0.818 0	0.721 1	0.604 8	0.480 7	0.376 3	0.425 8	0.424 5
− Parent	0.700 7	0.722 5	0.760 8	0.777 2	0.690 2	0.593 7	0.490 8	0.454 0	0.464 3	0.454 5
− Siblings	0.644 5	0.824 2	0.792 3	0.761 4	0.643 1	0.464 2	0.332 7	0.260 9	0.147 7	0.000 0

Table 6. Exploring Performance of Beam Search on Test Set of the Chinese Dataset

Model	*acc*	$F1_1$	$F1_2$	$F1_3$	$F1_4$	$F1_5$	$F1_6$	$F1_7$	$F1_8$	$F1_9$	$F1_{10}$	$F1_{11}$
HELD ($bs = 1$)	0.973 1	0.989 2	0.948 9	0.978 4	0.983 8	0.977 8	0.967 5	0.944 9	0.917 4	0.852 1	0.851 7	0.705 9
HELD ($bs = 3$)	0.973 6	0.989 2	0.948 9	0.978 4	0.983 8	0.977 8	0.968 5	0.948 8	0.917 9	0.852 3	0.873 4	1.000 0

Table 7. Exploring Performance of Beam Search on Test Set of the English Dataset

Model	*acc*	$F1_1$	$F1_2$	$F1_3$	$F1_4$	$F1_5$	$F1_6$	$F1_7$	$F1_8$	$F1_9$
HELD ($bs = 1$)	0.730 1	0.855 6	0.838 0	0.818 0	0.721 1	0.604 8	0.480 7	0.376 3	0.425 8	0.424 5
HELD ($bs = 3$)	0.727 5	0.827 7	0.826 2	0.814 8	0.719 9	0.605 3	0.483 8	0.377 5	0.437 3	0.481 3

we adopt a commercial product, PDFLux[4], for document layout recognition, which detects physical objects (e.g., paragraphs, tables, graphs) on each document page. Since the document physical objects are labeled by annotations in each dataset, we can evaluate the performance of PDFLux. Here, we use a rigorous metric. First, we define the exact match of a predicted object if it detects the exact region without missing any text or containing any redundant text outside objects compared with the ground-truth object. Then, we can calculate the precision, recall and $F1$ value of the exact-matched physical objects.

The results of the Chinese and English datasets are shown in Table 8. PDFLux obtains 0.966 7 and 0.973 8 $F1$ in the Chinese and English datasets, respectively. In other words, around 97% predicted physical objects exactly match the ground-truth objects.

Table 8. Exploring Performance of Document Layout Recognition

Dataset	Precision	Recall	$F1$
Chinese dataset	0.966 8	0.966 6	0.966 7
English dataset	0.973 4	0.974 2	0.973 8

Next, based on the predicted physical objects, we can use the HELD model to recognize the logical hierarchy of each document. Thus, we evaluate the logical hierarchy based on the predicted physical objects as shown in Table 9 and Table 10. Compared with recognizing the logical hierarchy based on the labeled physical objects, recognizing the logical hierarchy based on predicted physical objects obtains 0.015 5 (from 0.973 1 to 0.957 6) and 0.006 3 (from 0.730 1 to 0.723 8) decrease in accuracy on the Chinese and the English dataset respectively. That is to say, the noise of document layout recognition has limited influence on the discovery of the logical document hierarchy.

5.2 Experimental Summary

In summary, the HELD model greatly outperforms the three baseline models on effectiveness. Meanwhile, since extracting headings explicitly leads to great improvement on effectiveness and efficiency, we choose the 2step-HELD model. To obtain higher generalization ability, we add the tolerance to insertion errors in the predecessor step. To achieve the tradeoff between effectiveness and efficiency, we choose root-to-leaf traversal order. Additionally, the leaf-to-root traversal method can be used if the requirement of efficiency is much higher in the real-world production.

5.3 Case Studies

In this subsection, we first show an example to compare the HELD model, the HEPS model (rule-based) and the TOC model (sequence labeling based). As shown in Fig.10(a), the difficulty of this example is that high-level and low-level headings use the same pattern (the pattern starts with a number and a punctuation, like "1. $\cdots$", "2.$\cdots$"), so that it is hard to decide the true level of each heading. The HEPS model predicts the incorrect position of "2. non-current asset" with its descendants and the TOC model also predicts incorrectly for the last three headings. However, the HELD model correctly predicts the hierarchy via text and contextual information, since it considers the parent and siblings information simultaneously for each heading.

Then, we use another example to show the limitation of the proposed HELD model. For example in Fig.10(b), the ground-truth hierarchy is shown on the right part. Note that, the next sibling of heading a, "(III) Changes $\cdots$", is heading d, "(V) Risks $\cdots$". The heading (IV)$\cdots$ is omitted for some reasons, like the errors of heading detection step or the errors of original document editing. In this scenario, the put-or-skip

Table 9. Comparing Logical Hierarchy Based on the Predicted and Labeled Document Layout in the Chinese Dataset

	Total	Level										
		1	2	3	4	5	6	7	8	9	10	11
Labeled layout	0.973 1	0.989 2	0.948 9	0.978 4	0.983 8	0.977 8	0.967 5	0.944 9	0.917 4	0.852 1	0.851 7	0.705 9
Predicted layout	0.957 6	0.964 6	0.927 3	0.973 7	0.962 3	0.960 8	0.953 5	0.942 3	0.910 3	0.846 6	0.834 5	0.666 7

Table 10. Comparing Logical Hierarchy Based on the Predicted and Labeled Document Layout in the English Dataset

	Total	Level								
		1	2	3	4	5	6	7	8	9
Labeled layout	0.730 1	0.855 6	0.838 0	0.818 0	0.721 1	0.604 8	0.480 7	0.376 3	0.425 8	0.424 5
Predicted layout	0.723 8	0.849 1	0.831 5	0.805 1	0.711 0	0.593 9	0.458 6	0.360 5	0.397 6	0.413 8

[4] https://pdflux.com/, Apr. 2021.

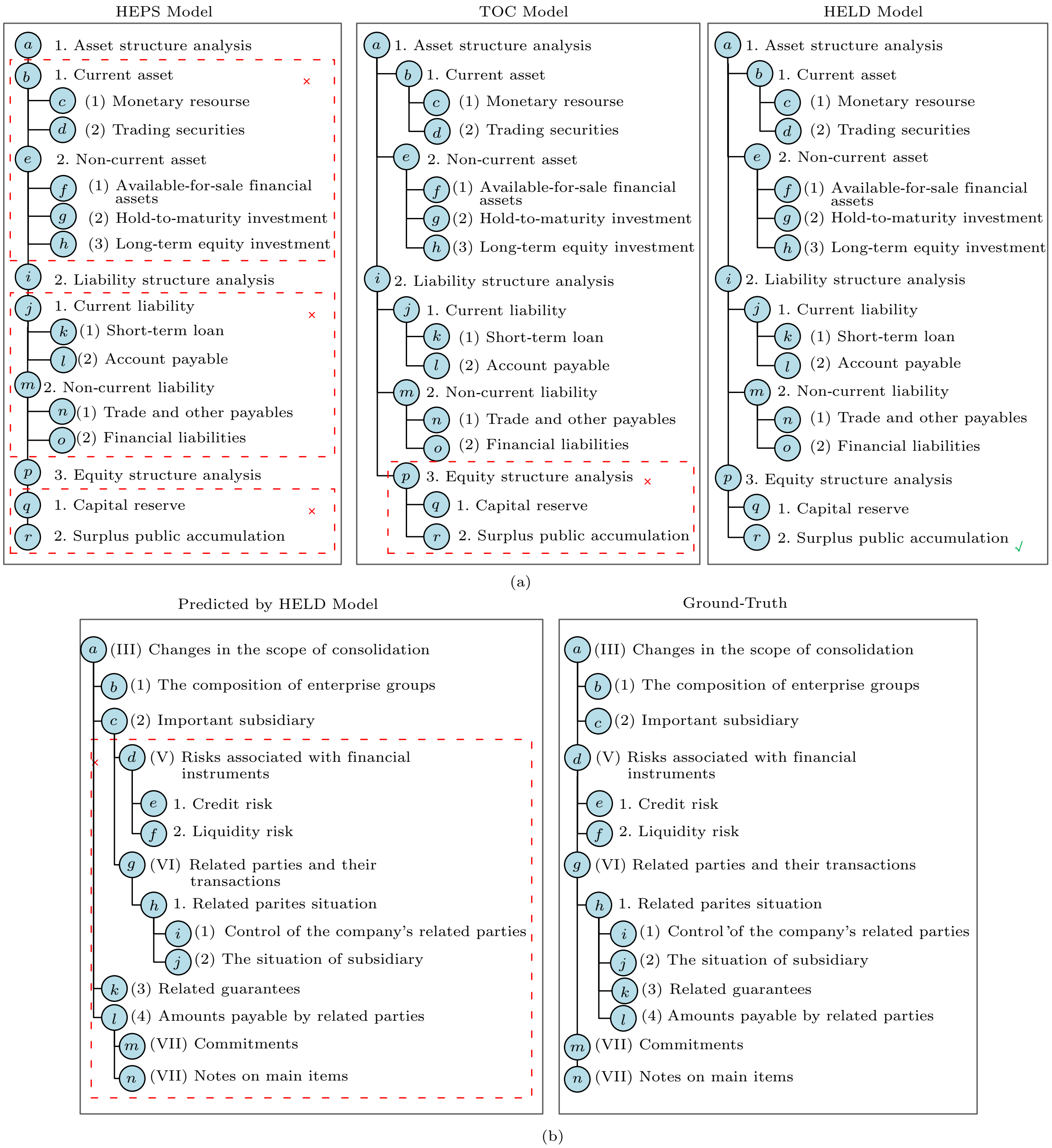

HEPS Model
TOC Model
HELD Model
a 1. Asset structure analysis
b 1. Current asset
c (1) Monetary resourse
d (2) Trading securities
e 2. Non-current asset
f (1) Available-for-sale financial assets
g (2) Hold-to-maturity investment
h (3) Long-term equity investment
i 2. Liability structure analysis
j 1. Current liability
k (1) Short-term loan
l (2) Account payable
m 2. Non-current liability
n (1) Trade and other payables
o (2) Financial liabilities
p 3. Equity structure analysis
q 1. Capital reserve
r 2. Surplus public accumulation
×
✓
(a)
Predicted by HELD Model
Ground-Truth
a (III) Changes in the scope of consolidation
b (1) The composition of enterprise groups
c (2) Important subsidiary
d (V) Risks associated with financial instruments
e 1. Credit risk
f 2. Liquidity risk
g (VI) Related parties and their transactions
h 1. Related parites situation
i (1) Control of the company's related parties
j (2) The situation of subsidiary
k (3) Related guarantees
l (4) Amounts payable by related parties
m (VII) Commitments
n (VII) Notes on main items
i (1) Control'of the company's related parties
(b)

Fig.10. Case studies. (a) Comparing the HEPS, TOC and HELD model. (b) Example to show the limitation of the HELD model.

module predicts heading d as the child of heading c by mistake, which causes that all the subsequent headings are predicted incorrectly. Thus, the HELD model has a limitation. It is hard to correctly predict the headings in the lower level once a heading in the higher level is predicted incorrectly. The main reason may be that the hierarchy generation process is a greedy search process in the HELD model. In the future, we aim to tackle this problem via the Monte Carlo Tree Search technique.

6 Downstream Application

We further explore how the logical document hierarchy can be leveraged in a downstream application of passage retrieval. Professionals usually need to retrieve the relevant passages in a document with hundreds of pages. Here, we define "passage" as paragraph, table, figure and so on in the document. Clearly, this task can be formulated as a learning-to-rank problem for all the content passages within a document. We show that the features extracted from the document hierarchy can significantly improve retrieval performance.

Generally, for any passage corresponding to a node in the logical tree, the features on the path from this node to the root may help on this task. Hence, besides the traditional BM25 feature four more features are extracted based on the document hierarchy. Specifically, "BM25AncMax" is the maximum BM25 score among the ancestor nodes of a given passage. The heading of a section usually contains the general description of its subsections. If the section heading hits the query keyword, the passages in this section are more likely to be relevant. "SameWordAnc" is the number of the same words between a given passage and its ancestors. We merge all the ancestors of the passage into one text and calculate the intersection number of words between the passage and the merged text. The ancestor nodes contain a general summary about the content under them, thereby the passage that contains more intersecting words has higher importance. "Pos" and "PosRatio" are the absolute and relative indexes of a given passage among its siblings. Note that, $PosRatio = \frac{Pos}{c}$, where c is the number of siblings of the passage. These two features point out the positional information of children, where the first and the last child often provide an integrated description that may have a higher rank. Based on these features, Gradient Boosting Decision Tree (GBDT) [39] is adopted to rank the passages.

The dataset contains 110 IPO prospectus in the Chinese market and a set of 138 queries are applied to each document. We spilt the dataset by queries with 88 queries for training and 50 queries for testing. Table 11 shows the measures of mAP and the recall on this testing for different sets of used features. The baseline model is to use only the BM25 feature. Then, each of the four new features is combined to get another four baseline models. Finally, we combine the four features with BM25 to get the final model. The experimental results show that each of the four features can improve the retrieval performance and the BM25AncMax and the Pos feature get relative prominent improvement. With all the four features together, we obtain 0.189 increase in mAP and 0.135 increase in recall@1. In conclusion, logical document hierarchy can be employed to significantly improve the performance of the downstream passage retrieval task.

Table 11. Results of Passage Retrieval

Adding Feature	mAP	recall@k		
		k=1	k=5	k=10
Only BM25	0.149	0.083	0.296	0.412
BM25 + BM25AncMax	0.269	0.184	0.376	0.465
BM25 + SameWordAnc	0.223	0.126	0.336	0.487
BM25 + Pos	0.254	0.165	0.403	0.519
BM25 + PosRatio	0.219	0.127	0.335	0.478
BM25 + All Four Features	0.338	0.218	0.471	0.576

It is worth mentioning that the proposed four new features heavily depend on the path from a node to the root. Any errors in this path might seriously worsen this application. Therefore, we argue again that the proposed measure which checks the path from any node to the root is more reasonable.

7 Conclusions

In this paper, we conducted a systematic study on the task of extracting logical document hierarchy from long documents in terms of methods, evaluations, and applications. We showed that the proposed HELD model with the root-to-leaf traversal order and explicit heading extraction is suitable to achieve the tradeoff between effectiveness and efficiency. Furthermore, we demonstrated that the downstream passage retrieval task significantly benefits from the extracted tree. We also argued that this proposed measure should be adopted in future studies of this task.

References

[1] Bloechle J L. Physical and logical structure recognition of pdf documents [PhD Thesis]. University of Fribourg, 2010.

[2] Mao S, Rosenfeld A, Kanungo T. Document structure analysis algorithms: A literature survey. In *Proc. the 2003 Document Recognition and Retrieval X*, Jan. 2003, pp.197-207. DOI: 10.1117/12.476326.

[3] Pembe F C, Gungor T. Heading-based sectional hierarchy identification for HTML documents. In *Proc. the 22nd International Symposium on Computer and Information Sciences*, Nov. 2007. DOI: 10.1109/ISCIS.2007.4456839.

[4] Geva M, Berant J. Learning to search in long documents using document structure. In *Proc. the 27th International Conference on Computational Linguistics*, Aug. 2018, pp.161-176.

[5] Howard T, Bruce C. Inference networks for document retrieval. *ACM SIGIR Forum*, 2017, 51(2): 124-147. DOI: 10.1145/3130348.3130361.

[6] Summers K. Automatic discovery of logical document structure [PhD Thesis]. Cornell University, 1998.

[7] Luong M T, Nguyen T D, Kan M Y. Logical structure recovery in scholarly articles with rich document features. *International Journal of Digital Library Systems*, 2010, 1(4): 1-23. DOI: 10.4018/jdls.2010100101.

[8] Pembe F C, Güngör T. A tree-based learning approach for document structure analysis and its application to Web search. *Natural Language Engineering*, 2014, 21(4): 569-605. DOI: 10.1017/S1351324914000023.

[9] Ramakrishnan C, Patnia A, Hovy E, Burns G A. Layout-aware text extraction from full-text pdf of scientific articles. *Source Code for Biology Medicine*, 2012, 7(1): Article No. 7. DOI: 10.1186/1751-0473-7-7.

[10] Manabe T, Tajima K. Extracting logical hierarchical structure of HTML documents based on headings. *Proceedings of the VLDB Endowment*, 2015, 8(12): 1606-1617. DOI: 10.14778/2824032.2824058.

[11] Rahman M M, Finin T. Understanding the logical and semantic structure of large documents. arXiv:1709.00770, 2017. https://arxiv.org/abs/1709.00770, April 2021.

[12] Bentabet N I, Juge R, Ferradans S. Table-of-contents generation on contemporary documents. In *Proc. the 2019 International Conference on Document Analysis and Recognition*, Sept. 2019, pp. 100-107. DOI: 10.1109/ICDAR.2019.00025.

[13] Conway A. Page grammars and page parsing: A syntactic approach to document layout recognition. In *Proc. the 2nd International Conference on Document Analysis and Recognition*, Oct. 1993, pp.761-764. DOI: 10.1109/ICDAR.1993.395626.

[14] Tsujimoto S, Asada H. Understanding multi-articled documents. In *Proc. the 10th International Conference on Pattern Recognition*, June 1990, pp.124-133. DOI: 10.1109/ICPR.1990.118163.

[15] Constantin A, Pettifer S, Voronkov A. PDFX: Fully-automated PDF-to-XML conversion of scientific literature. In *Proc. the 2013 ACM Symposium on Document Engineering*, Sept. 2013, pp.177-180. DOI: 10.1145/2494266.2494271.

[16] Tkaczyk D, Szostek P, Fedoryszak M, Dendek P J, Bolikowski. CERMINE: Automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition*, 2015, 18(4): 317-335. DOI: 10.1007/s10032-015-0249-8.

[17] Summers K. Toward a taxonomy of logical document structures. In *Proc. the Dartmouth Institute for Advanced Graduate Studies: Electronic Publishing and the Information Superhighway*, May 30-June 2, 1995, pp.124-133.

[18] Baird H S, Jones S E, Fortune S J. Image segmentation by shape-directed covers. In *Proc. the 10th International Conference on Pattern Recognition*, June 1990, pp.820-825. DOI: 10.1109/ICPR.1990.118223.

[19] Nagy G, Seth S, Viswanathan M. A prototype document image analysis system for technical journals. *Computer*, 1992, 25(7): 10-22. DOI: 10.1109/2.144436.

[20] Kopec G E, Chou P A. Document image decoding using Markov source models. In *Proc. the 1993 IEEE International Conference on Acoustics Speech and Signal Processing*, April 1993, pp.85-88. DOI: 10.1109/ICASSP.1993.319753.

[21] Xiao Y, Yumer E, Asente P, Kraley M, Kifer D, Giles C L. Learning to extract semantic structure from documents using multimodal fully convolutional neural network. In *Proc. the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp.4342-4351. DOI: 10.1109/CVPR.2017.462.

[22] Augusto Borges Oliveira D, Palhares Viana M. Fast CNN-based document layout analysis. In *Proc. the 2017 IEEE International Conference on Computer Vision Workshops*, Oct. 2017, pp.1173-1180. DOI: 10.1109/ICCVW.2017.142.

[23] Wong K Y, Casey R G, Wahl F M. Document analysis system. *IBM Journal of Research and Development*, 1982, 26(6): 647-656. DOI: 10.1147/rd.266.0647.

[24] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[25] Noh H, Hong S, Han B. Learning deconvolution network for semantic segmentation. In *Proc. the 2015 IEEE International Conference on Computer Vision*, Dec. 2015, pp.1520-1528. DOI: 10.1109/ICCV.2015.178.

[26] He D, Cohen S, Price B, Kifer D, Giles C L. Multi-scale multi-task FCN for semantic page segmentation and table detection. In *Proc. the 14th IAPR International Conference on Document Analysis and Recognition*, Nov. 2017, pp.254-261. DOI: 10.1109/ICDAR.2017.50.

[27] Schuster M, Paliwal K K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997, 45(11): 2673-2681. DOI: 10.1109/78.650093.

[28] Zhou G, Luo P, Cao R, Xiao Y, Lin F, Chen B, He Q. Tree-structured neural machine for linguistics-aware sentence generation. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, February 2018, pp.5722-5729.

[29] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks. In *Proc. the 27th International Conference on Neural Information Processing Systems*, December 2014, pp.3104-3112.

[30] Tan Z, Wang M, Xie J, Chen Y, Shi X. Deep semantic role labeling with self-attention. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, Feb. 2018, pp.4929-4936.

[31] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L, Polosukhin I. Attention is all you need. In *Proc. the 31st International Conference on Neural Information Processing*, December 2017, pp.5998-6008.

[32] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. In *Proc. the 2013 International Conference on Learning Representations*, May 2013.

[33] Lin M, Chen Q, Yan S. Network in network. arXiv:1312.4400, 2013. https://arxiv.org/abs/1312.4400, Jan. 2021.

[34] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. the 32nd International Conference on Machine Learning*, July 2015, pp.448-456.

[35] Nair V, Hinton G E. Rectified linear units improve restricted Boltzmann machines. In *Proc. the 27th International Conference on Machine Learning*, Jun. 2010, pp.807-814.

[36] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proc. the IEEE International Conference on Computer Vision*, Dec. 2015, pp.1026-1034. DOI: 10.1109/ICCV.2015.123.

[37] Kingma D P, Ba J. Adam: A method for stochastic optimization. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[38] Sergeev A, Del Balso M. Horovod: Fast and easy distributed deep learning in TensorFlow. arXiv:1802.05799, 2018. https://arxiv.org/abs/1802.05799, Jan. 2021.

[39] Friedman J H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 2001, 29(5): 1189-1232. DOI: 10.1214/aos/1013203451.

Rong-Yu Cao received his B.E. degree in software engineering from Dalian University of Technology, Dalian, in 2016, and now he is a Ph.D. student at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include natural language processing and document analysis.

Yi-Xuan Cao received his B.E. degree in transportation engineering from Tongji University, Shanghai, in 2015, and now is a Ph.D. student at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include natural language processing and information extraction.

Gan-Bin Zhou received his Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2018. He is an engineerer in WeChat Search Application Department, Tencent Holdings Ltd, Beijing. His research interests include natural language processing and text generation.

Ping Luo received his Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2007. He is an associate professor in the Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing. His general area of research is knowledge discovery and machine learning.

JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY

Volume 37, Number 3, May 2022

Content

Special Section on Self-Learning with Deep Neural Networks

Special Section of CVM 2022

Regular Paper

JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY
《计算机科学技术学报》

Volume 37 Number 3 2022 (Bimonthly, Started in 1986)
Indexed in: **SCIE, Ei, INSPEC, JST, AJ, MR, CA, DBLP**

Edited by:
THE EDITORIAL BOARD OF JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY
Zhi-Wei Xu, Editor-in-Chief, P.O. Box 2704, Beijing 100190, P.R. China
Managing Editor: Feng-Di Shu E-mail: jcst@ict.ac.cn http://jcst.ict.ac.cn Tel.: 86-10-62610746

Sponsored by: Institute of Computing Technology, CAS & China Computer Federation
Supervised by: Chinese Academy of Sciences
Undertaken by: Institute of Computing Technology, CAS
Published by: Science Press, Beijing, China
Printed by: Beijing Baochang Color Printing Co. Ltd
Distributed by:
China: All Local Post Offices
Other Countries: Springer Nature Customer Service Center GmbH, Tiergartenstr. 15, 69121 Heidelberg, Germany
Available Online: https://link.springer.com/journal/11390

ISSN 1000-9000

国内统一刊号: CN11-2296/TP 国内邮发代号: 2-578 RMB ¥160.00