Hierarchical Text Classification as Sub-hierarchy Sequence Generation

SangHun Im, GiBaeg Kim, Heung-Seon Oh, Seongung Jo, Dong Hwan Kim
School of Computer Science and Engineering, Korea University of Technology and Education (KOREATECH) 
{tkrhskhdqn, fk0214, oohs, oowhat, hwan6615}@koreatech.ac.kr

Abstract
Hierarchical text classification (HTC) is essential for various real applications. However, HTC models are challenging to develop because they often require processing a large volume of documents and labels with hierarchical taxonomy. Recent HTC models based on deep learning have attempted to incorporate hierarchy information into a model structure. Consequently, these models are challenging to implement when the model parameters increase for a large-scale hierarchy because the model structure depends on the hierarchy size. To solve this problem, we formulate HTC as a sub-hierarchy sequence generation to incorporate hierarchy information into a target label sequence instead of the model structure. Subsequently, we propose the Hiarchy DECoder (HiDEC), which decodes a text sequence into a sub-hierarchy sequence using recursive hierarchy decoding, classifying all parents at the same level into children at once. In addition, HiDEC is trained to use hierarchical path information from a root to each leaf in a sub-hierarchy composed of the labels of a target document via an attention mechanism and hierarchy-aware masking. HiDEC achieved state-of-the-art performance with significantly fewer model parameters than existing models on benchmark datasets, such as RCV1-v2, NYT, and EURLEX57K.

Introduction
Hierarchical text classification (HTC) uses a hierarchy such as a web taxonomy to classify a given text into multiple labels. Moreover, classification tasks are essential in real-world applications because of the tremendous amount of data on the web that should be properly organized for applications such as product navigation (Kozareva 2015; Cevahir and Murakami 2016) and news categorization (Lewis et al. 2004; Sandhaus. 2008).

Recent HTC research using deep learning can be categorized into local and global approaches. In the local approach (Peng et al. 2018; Kowsari et al. 2017; Shimura, Li, and Fukimoto 2018; Banerjee et al. 2019; Dumais and Chen 2000; Wehrmann, Cerri, and Barros 2018), a classifier is built for each unit after the entire hierarchy is split into a set of small units. Subsequently, the classifiers are applied in sequence according to a path from a root to target labels in a top-down manner. In contrast, in the global approach (Zhao et al. 2018; Wang et al. 2021; Peng et al. 2021; Wang et al. 2022; Zhou et al. 2020; Mao et al. 2019; Chen et al. 2021; Sinha et al. 2018; Deng et al. 2021; Wu, Xiong, and Wang 2019; Yang et al. 2018), a classifier for all labels in the entire hierarchy is built, excluding the hierarchy structure through flattening. A document hierarchy information can be obtained using a structure encoder and merged with text features from a text encoder. The global approach achieves superior performance to the local approach owing to the effective design of the structure encoders, such as the graph convolution network (Kipf and Welling 2017) and Graphormer (Ying et al. 2021).

However, the global approach has scalability limitations because the structure encoders require a disproportionate number of parameters as the size of the hierarchies increases. For example, HiAGM (Zhou et al. 2020) and HiMatch (Chen et al. 2021), recent GCN-based structure encoders, require a weight matrix to convert text features to label features. In HGCLR (Wang et al. 2022), edge and spatial encodings were employed to represent the relationship between two nodes.

In existing global models, a large model size is inevitable because they attempt to incorporate the entire hierarchy in-
formation with respect to the document labels into a model structure. In contrast, employing a sub-hierarchy comprising a set of paths from a root to each target label is sufficient because most labels are irrelevant to a target document. To this end, we formulate HTC as a sub-hierarchy sequence generation using an encoder-decoder architecture to incorporate the sub-hierarchy information into a target label sequence instead of the model structure. Figure 1 shows the differences between the proposed approach and the existing global models. For example, given a hierarchy and two documents with different labels in Figure 1-(a), a global model attempts to capture the entire hierarchy information with respect to the document labels, as shown in Figure 1-(b). For further improvement, the document labels are converted into a sub-hierarchy sequence using a depth-first search on an entire hierarchy and parse tree notation, as shown in Figure 1-(c) and -(d).

Based on this idea, we propose a Hierarchy DECoder (HiDEC), which recursively decodes the text sequence into a sub-hierarchy sequence by sub-hierarchy decoding while remaining aware of the path information. The proposed method comprises hierarchy embeddings, hierarchy-aware masked self-attention, text-hierarchy attention, and sub-hierarchy decoding, similar to the decoder of Transformer (Vaswani et al. 2017). Hierarchy-aware masked self-attention facilitates learning all hierarchy information in a sub-hierarchy sequence. A hierarchy-aware mask captures the sub-hierarchy information by considering the dependencies between each label and its child labels from the sub-hierarchy sequence at once in the training step. Subsequently, two features generated from the hierarchy-aware masked self-attention and a text encoder are merged through text-hierarchy attention. Note that HiDEC does not require additional parameters for the structure encoder as used in global models, such as HiAGM, HiMatch, and HGCLR. Consequently, the parameters of the model increase linearly with respect to the classes in a hierarchy. In the inference step, sub-hierarchy decoding is recursively applied to expand from parent to child labels in a top-down manner. As a result, all parent labels at the same depth are expanded simultaneously. Thus, the maximum recursions are the depth of the entire hierarchy and not the sequence length.

A series of experiments show that HiDEC outperforms state-of-the-art (SOTA) models on two small-scale datasets, RCV1-v2 and NYT, and a large-scale dataset, EURLEX57K. Consequently, HiDEC achieves better performance with significantly fewer parameters on the three benchmark datasets. Thus, the proposed approach can solve scalability problems in large-scale hierarchies.

The contributions of this paper can be summarized as follows:

• This paper formulates HTC as a sub-hierarchy sequence generation using an encoder-decoder architecture. We can incorporate the hierarchy information into the sub-hierarchy sequence instead of the model structure, as all dependencies are aware by parse tree notation.

• This paper proposes a Hierarchy DECoder (HiDEC) that recursively decodes the text sequence into a sub-hierarchy sequence by sub-hierarchy decoding while remaining aware of the path information.

• This paper demonstrates the superiority of HiDEC by comparing SOTA models on three benchmark HTC datasets (RCV1-v2, NYT, and EURLEX57K). The results reveal the role of HiDEC in HTC through in-depth analysis.

Related Work

The critical point to HTC is the use of hierarchy information, denoted as relationships among labels. For example, the relationships include root-to-target labels (path information), parent-to-child, and the entire hierarchy (holistic information).

Research on HTC can be categorized into local and global approaches. In the local approach, a set of classifiers are used for small units of classes, such as for-each-class (Banerjee et al. 2019), for-each-parent (Dumais and Chen 2000; Kowsari et al. 2017), for-each-level (Shimura, Li, and Fukumoto 2018), and for-each-sub-hierarchy (Peng et al. 2018). In (Kowsari et al. 2017), HDLTex was introduced as a local model that combined a deep neural network (DNN), CNN, and RNN to classify child nodes. Moreover, HTrans (Banerjee et al. 2019) extended HDLTex to maintain path information across local classifiers based on transfer learning from parent to child. HMCN (Wehrmann, Cerri, and Barros 2018) applied global optimization to the classifier of each level to solve the exposure bias problem. Finally, HR-DGCNN (Peng et al. 2018) divided the entire hierarchy into sub-hierarchies using recursive hierarchical segmentation. Unfortunately, applying this to large-scale hierarchies is challenging because many parameters are required from a set of local classifiers for small units of classes.

In the global approach, the proposed methods employed a single model with path (Zhao et al. 2018; Wang et al. 2021; Peng et al. 2021; Mao et al. 2019; Sinha et al. 2018; Deng et al. 2021) or holistic (Zhou et al. 2020; Chen et al. 2021; Wang et al. 2022) information. For instance, HNATC (Sinha et al. 2018) obtained path information using a sequence of outputs from the previous levels to predict the output at the next level. In HiLAP-RL (Mao et al. 2019), reinforcement learning was exploited in that HTC was formalized as a pathfinding problem. Moreover, HE-AGRCNN (Peng et al. 2021) and HCSM (Wang et al. 2021) used capsule networks. Recent research (Zhou et al. 2020; Chen et al. 2021; Wang et al. 2022) has attempted to employ holistic information of an entire hierarchy using a structure encoder with GCN (Kipf and Welling 2017) and Graphormer (Ying et al. 2021). HiAGM (Zhou et al. 2020) propagated text through GCN, HiMatch (Chen et al. 2021) improved HiAGM by adapting semantic matching between text and label features from text features and label embeddings. In addition, HGCLR (Wang et al. 2022) attempted to unify a structure encoder with a text encoder using a novel contrastive learning method, where BERT (Devlin et al. 2019) and Graphormer were employed for the text and structure encoders, respectively. Therefore, classification was performed...
using a hierarchy-aware text feature produced by the text encoder. Finally, it reported SOTA performance on RCV1-v2 and NYT but is infeasible in large-scale hierarchies because of the large model size caused by incorporating holistic information into the model structure.

**Proposed Methods**

The HTC problem can be defined using a tree structure. A hierarchy is represented as a tree $G = (V, \bar{E})$ where $V = \{v_1, v_2, \ldots, v_N\}$ is a set of $C$-labels in the hierarchy, and $\bar{E} = \{(v_i, v_j) | v_i \in V, v_j \in child(v_i)\}$ is a set of edges between a label $v_i$ and a child $v_j$ of $v_i$. $D = \{d_1, d_2, \ldots, d_K\}$ is a collection of $K$ documents. A document $d_k$ has a sub-hierarchy $G^d_k = (V^d_k, \bar{E}^d_k)$ converted from assigned labels where $V^d_k = L^d_k \cup \{v_i^d_k | v_i^d_k \in ancestor(v_j^d_k), v_j^d_k \in L^d_k\}$ and $\bar{E}^d_k = \{(v_i^d_k, v_j^d_k) | v_j^d_k \in V^d_k, v_i^d_k \in parent(v_j^d_k)\}$, where $L^d_k = \{v_1^d_k, v_2^d_k, \ldots, v_{t^d_k}\}$ is a label set of document $d_k$. In other words, $G^d_k$ is constructed using all the labels assigned to $d_k$ and their ancestors. $G^d_0 = (V_{root}^d, \emptyset)$ is the initial sub-hierarchy of HiDEC, which has a root and no edges. Based on $G^d_0$, recursive hierarchy decoding is defined by expanding $G^d_p$ for $p$ times from $p=0$. The goal of training HiDEC is given by $G^d_p = G^d_k$.

In Figure 2-(a), the overall architecture of HiDEC is presented with a demonstration of the recursive hierarchy decoding. The remainder of this section presents the details of the proposed model.

**Text Encoder**

In the proposed model, a text encoder can use any model that outputs the text feature matrix of all the input tokens, such as GRU and BERT (Devlin et al. 2019). For simplicity, let us denote $d_k$ as $\mathbf{T} = [w_1, w_2, \ldots, w_N]$ where $w_n$ is a one-hot vector for an index of the $n$-th token. Initially, a sequence of tokens was converted into word embeddings $\mathbf{H}^0(= \mathbf{W}^0 \mathbf{T}) \in \mathbb{R}^{N \times e}$ where $\mathbf{W}^0$ is the weight matrix of the word embedding layer, and $e$ is an embedding dimension. Given $\mathbf{H}^0$, the hidden state $\mathbf{H}$ from the text encoder can be computed using Equation 1:

$$\mathbf{H} = \text{TextEncoder}(\mathbf{H}^0).$$

(1)

**Hierarchy DECoder (HiDEC)**

**Hierarchy Embedding Layer** The sub-hierarchy embeddings, as shown in Figure 1, is obtained by initially constructing a sub-hierarchy sequence from a document $d_k$. This process consists of two steps. First, a sub-hierarchy $G^d_k = (V^d_k, \bar{E}^d_k)$ of $d_k$ is built with its target labels. Second, a sub-hierarchy sequence $\mathbf{S}$ following a parse tree notation is generated from $G^d_k$. Three special tokens, “(”, “)”, and “[END]”, are used to properly represent the sub-hierarchy. The tokens “(” and “)” denote the start and end of a path from each label, respectively, whereas the “[END]” token indicates the end of a path from a root. For example, $\mathbf{S} = [(R (A (D (I [END])))) (B (F ([END])) (C ([END])))]$ is constructed in Figure 1 with a label set $\{C, F, I\}$. Once again, the tokens in $\mathbf{S}$ are represented as one-hot vectors for further processing. Subsequently, these tokens can be represented as $\mathbf{S} = [s_1, s_2, \ldots, s_M]$ where $s_i = \mathbb{I}_v$ is a one-hot vector for a label $v$ and the special tokens. Finally, the sub-hierarchy embeddings $\mathbf{U}^0$ are constructed after explicitly incorporating the level information, similar to Transformer’s position encoding (Vaswani et al. 2017), using Equations 2 and 3:

$$\mathbf{U}^0 = \mathbf{W}^0 \mathbf{S},$$

(2)

$$\mathbf{U}^0 = \text{LevelEmbedding}(\mathbf{U}^0).$$

(3)

**Hierarchy-Aware Masked Self-Attention** This component is responsible for capturing hierarchy information, similar to the structure encoder in global models (Zhou et al. 2020; Chen et al. 2021; Wang et al. 2022). However, only a sub-hierarchy from the entire hierarchy, which is thought to be highly relevant information for classification, is used based on the self-attention mechanism used by Transformer
Algorithm 1: Recursive Hierarchy Decoding in Inference

Indices: Hierarchy depth $P$, Number of attentive layers $R$
Input: Text feature matrix from text encoder $H$
Output: Predicted label set $L$

//HiDEC
1: $L = \emptyset$
2: $G_0 = \{ \{v_{root}\}, \emptyset \}$
3: for $p = 0, \ldots, P - 1$ do
   //Sub-hierarchy embedding
   4: Convert $G_p$ to sub-hierarchy sequence $S_p$
   5: Compute $U^0$ from $S_p$ with Eq.2, 3
   6: Generate masking matrix $M$ with Eq.5
   //Attention layers
   7: for $r = 0, \ldots, R - 1$ do
      8: $U^{r+1} = \text{Attention}(U^r, H, M)$ with Eq.4, 6, 7
   9: end for
10: $U = U^R$
11: //Sub-hierarchy expansion
12: for $s_i \in S_p$ do
13: if $s_i \notin$ special token set then
14: $v_i = s_i$
15: for $v_j \in \text{child}(v_i)$ do
16: $c_{ij} = U_i \cdot W_S \cdot I_v_j$ with Eq.8
17: $p_i = \text{sigmoid}(c_i)$ with Eq.9
18: Get $y_i$ from $p_i$ by thresholding
19: for $v_k \in y_i$ do
20: $V_p = V_p \cup \{ v_k \}$
21: $E_p = E_p \cup \{ (v_i, v_k) \}$
22: end for
23: end for
24: end if
25: $G_{p+1} = (V_p, E_p)$
26: end for
//Label assignment
27: for $i = 0, \ldots, |V_p|$ do
28: if $v_i \in \text{leaf}(G_p)$ then
29: $L = L \cup \{ v_i \}$
30: else if $v_i = \text{"[END]"}$ then
31: $L = L \cup \{ \text{parent}(v_i) \}$
32: end if
33: end for
34: return $L$

To compute self-attention scores, we applied hierarchy-aware masking to incorporate hierarchy information. The self-attention mechanism of Transformer was exploited with a minor modification concerning hierarchy-aware masking. The hierarchy-aware masked self-attention of $r$-th layer is computed using Equation 4:

$$U^r = \text{MHA}(W_Q U^{r-1}, W_K U^{r-1}, W_V U^{r-1}, M),$$

where MHA is the multi-head attention, the same as that of Transformer. $W_Q, W_K, W_V$ are projection weight matrices for the query, key, and value, respectively. Moreover, $M$ is the hierarchy-aware mask defined as follows:

$$M_{ij} = \begin{cases} -1e9 & \text{if } v_i \notin \text{ancestor}(v_j) \\ 0 & \text{else} \end{cases}$$

We ignore the dependency between two labels if they are not the same label and not an ancestor by setting $M_{ij} = -1e9$. This setting makes the model attend to the path information relevant to the input documents and ignores the hierarchy information at the lower-level labels than each label to learn a sub-hierarchy structure at once in a training step. Note that the dependencies of the three special tokens with respect to the other tokens, including themselves, are considered.

Text-Hierarchy Attention In text-hierarchy attention, we can compute the attention scores of labels by dynamically reflecting the importance of tokens in an input document. A new sub-hierarchy matrix $\hat{U}^r$ of $r$-th layer is computed by combining the text feature matrix $H$ from the encoder and $U^r$ without a masking mechanism using Equation 6:

$$\hat{U}^r = \text{MHA}(W_Q \hat{U}^r, W_K H, W_V H, -).$$

Subsequently, the output of $r$-th layer $\hat{U}^r$ is obtained using a position-wise feed-forward network (FFN) using Equation 7:

$$\hat{U}^r = \text{FFN}(\hat{U}^r).$$

Consequently, the output of the final layer, $U$, in HiDEC is used in the sub-hierarchy expansion.

Sub-Hierarchy Decoding Sub-hierarchy decoding is crucial in generating a sub-hierarchy using recursive hierarchy decoding. This results in a target sub-hierarchy if HiDEC functions as expected. For each label, the classification to child labels is performed using the sub-hierarchy matrix $U$ using Equations 8 and 9:

$$c_{ij} = U_i \cdot W_S \cdot I_v_j \quad \forall v_j \in \text{child}(v_i),$$

$$p_i = \text{sigmoid}(c_i),$$

where $c_{ij}$ is a similarity score of child $v_j$ under a parent $v_i$, and $p_i$ is the probability of a child $v_j$ obtained using a task-specific probability function such as sigmoid. The three special tokens are excluded when selecting the parent $v_i$.

We reduced the label space of HTC by focusing on the child labels of a parent label of interest. During training, we use binary cross-entropy loss functions as shown in Equation 10:

$$L = -\frac{1}{MJ} \sum_{i=0}^{M} \sum_{j=0}^{J} y_{ij} \log(p_{ij}) + (1 - y_{ij}) \log(1 - p_{ij}),$$

where $J = |\text{child}(v_i)|$ indicates the number of child labels for parent $v_i$. Moreover, $y_{ij}$ and $p_{ij}$ denote a target label of $j$-th child label of $v_i$ and its output probability, respectively.

At the inference time, recursive hierarchy decoding is performed using a threshold. The details of the recursive hierarchy decoding are described in Algorithm 1. The number of decoding steps is the same as the maximum depth of the hierarchy. At each decoding step, all tokens except the special tokens are expanded. Decoding ends if the tokens are leaf labels or "[END]". Finally, the labels associated with "[END]" or leaf labels are assigned to the input as predictions.
Matrix from the last layer of BERT was used as the context. The output hidden state of the GRU-based models. With BERT as a text encoder model, we set $\ell r$ and the mini-batch size to 5e-5 and 64, respectively. The $\ell r$ was controlled using a linear schedule with a warm-up rate of 0.1. Gradient clipping with a maximum gradient norm of 1.0 was performed to prevent gradient overflow.

All models were implemented using PyTorch (Paszke et al. 2019) and trained using NVIDIA A6000. The average score of the ten different models was utilized as the proposed model performance, where the model with the best performance was selected using Micro-F1 on the validation data.

**Comparison Models**

We selected various baseline models from recent work. For RCV1-v2 and NYT, TextRNN (Lai et al. 2015), HiAGM (Zhou et al. 2020), HiMatch (Chen et al. 2021), HTCInfoMax (Deng et al. 2021), and HGCLR (Wang et al. 2022) were chosen. TextRNN comprises bi-GRU and CNN layers and is a hierarchy-unaware model used as a text encoder in HiAGM and HiMatch. HiAGM combines text and label features from the text encoder and GCN-based structure encoder, respectively, using text propagation. HTCInfoMax and HiMatch improved HiAGM with a prior distribution and semantic matching loss, respectively. They can be improved by replacing the text encoder with PLM. HGCLR directly embeds hierarchy information into the text encoder using Graphormer during training. For EURLEX57K, BiGRU-ATT (Xu et al. 2015), HAN (Yang et al. 2016), CNN-LWAN (Mullenbach et al. 2018), BiGRU-LWAN (Chalkidis et al. 2019), and HiMatch (Chen et al. 2021) were chosen. BiGRU-ATT and HAN are strong baselines for text classification tasks with an attention mechanism. CNN and BiGRU-LWAN extended BiGRU-ATT with label-wise attention. Owing to the large model size, applying HiMatch to EURLEX57K is infeasible. According to the paper (Chen et al. 2021), the text-propagation module weakly influences the performance compared with the original approach. Therefore, we simplified HiMatch by removing the text-propagation module.

**Experimental Results**

Table 2 summarizes the performance of HiDEC and the other models on (a) two small-scale datasets, RCV1-v2 and NYT; and (b) a large-scale dataset, EURLEX57K. HiDEC achieved the best performance on the three datasets regardless of whether BERT was used (Devlin et al. 2019) as the text encoder. This highlights the effectiveness of HiDEC using sub-hierarchy information rather than entire-hierarchy information. In addition, we found that HiDEC highly benefits from PLM compared to other models, as the performance gains of HiDEC with and without BERT are relatively large. For example, in MicroF1 on RCV1-v2, the gain between HiDEC and HiDEC with PLM was 2.42. In contrast, the gain between HiMatch (Chen et al. 2021) and Hi-
Table 2: Performance comparison on three datasets, (a) RCV1-v2 and NYT, (b) EURLEX57K. * denotes models without hierarchy information.

<table>
<thead>
<tr>
<th>Model</th>
<th>RCV1-v2 Micro</th>
<th>RCV1-v2 Macro</th>
<th>NYT Micro</th>
<th>NYT Macro</th>
<th>EURLEX57K Micro</th>
<th>EURLEX57K Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Pretrained Language Models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextRCNN* (Zhou et al. 2020)</td>
<td>81.57</td>
<td>59.25</td>
<td>70.83</td>
<td>56.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HiAGM (Zhou et al. 2020)</td>
<td>83.96</td>
<td>63.35</td>
<td>74.97</td>
<td>60.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTCInfoMax (Deng et al. 2021)</td>
<td>83.51</td>
<td>62.71</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HiMatch (Chen et al. 2021)</td>
<td>84.73</td>
<td>64.11</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HiDEC</td>
<td><strong>85.54</strong></td>
<td><strong>65.08</strong></td>
<td><strong>76.42</strong></td>
<td><strong>63.99</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/ Pretrained Language Models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT* (Wang et al. 2022)</td>
<td>85.65</td>
<td>67.02</td>
<td>78.24</td>
<td>65.62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HiAGM (Wang et al. 2022)</td>
<td>85.58</td>
<td>67.93</td>
<td>78.64</td>
<td>66.76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTCInfoMax (Wang et al. 2022)</td>
<td>85.53</td>
<td>67.09</td>
<td>78.75</td>
<td>67.31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HiMatch (Chen et al. 2021)</td>
<td>86.33</td>
<td>68.66</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HGCLR (Wang et al. 2022)</td>
<td>86.49</td>
<td>68.31</td>
<td>78.86</td>
<td>67.96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HiDEC</td>
<td><strong>87.96</strong></td>
<td><strong>69.97</strong></td>
<td><strong>79.99</strong></td>
<td><strong>69.64</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Model parameter comparison. * denotes models without hierarchy information.

<table>
<thead>
<tr>
<th>Model</th>
<th>RCV1-v2</th>
<th>NYT</th>
<th>EURLEX57K</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Pretrained Language Models</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextRCNN*</td>
<td>18M</td>
<td>18M</td>
<td>19M</td>
</tr>
<tr>
<td>HiAGM</td>
<td>31M</td>
<td>42M</td>
<td>5,915M</td>
</tr>
<tr>
<td>HiMatch</td>
<td>37M</td>
<td>52M</td>
<td>6,211M</td>
</tr>
<tr>
<td>HiDEC</td>
<td><strong>20M</strong></td>
<td><strong>20M</strong></td>
<td><strong>21M</strong></td>
</tr>
<tr>
<td>w/ Pretrained Language Models</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT*</td>
<td>109M</td>
<td>109M</td>
<td>112M</td>
</tr>
<tr>
<td>HGCLR</td>
<td>120M</td>
<td>121M</td>
<td>425M</td>
</tr>
<tr>
<td>HiDEC</td>
<td>123M</td>
<td>123M</td>
<td><strong>127M</strong></td>
</tr>
</tbody>
</table>

Match with PLM was 1.60. On EURLEX57K, training the hierarchy-aware models was infeasible because of the large model size\(^2\) caused by the structure encoder for 4,271 labels, except for HiDEC and simplified HiMatch. However, this does not apply to BERT because of the model size. Similar to RCV1-v2 and NYT, HiDEC with BERT exhibited the best performance.

Model Parameters

Table 3 summarizes the parameters for different models on three benchmark datasets. The label sizes are RCV1-v2 (103) < NYT (166) << EURLEX57K (4,271). The table shows that the parameters of the existing models increase dramatically as the label size increases. Note that HiDEC requires significantly fewer parameters even though the label size increases. In the extreme case on EURLEX57K, HiDEC only requires 21M parameters, which are 295x smaller than HiMatch with 6,211M. Consequently, the parameters in HiDEC increase linearly with respect to the labels in a hierarchy because no extra parameters are required for the structure encoder and sub-tasks. HiAGM and HiMatch require merging parameters projecting text features into label features for text propagation. In addition, HGCLR needs edge and spatial encoding parameters for the structure encoder. However, HiDEC does not require these parameters because attentive layers play the same role. Only the label embeddings increase according to the label size in a hierarchy.

Ablation Studies

Table 4 shows the ablation studies of each component in HiDEC without PLM on RCV1-v2 and EURLEX57K. HiDEC differs from the original Transformer decoder (Vaswani et al. 2017) in the absence of a residual connection and the existence of hierarchy-aware masking and level embedding. We observed that adding the residual connection and eliminating hierarchy-aware masking and level embedding also had a negative effect. Among them, adding a residual connection had the most negative effect. We presumed that the essential information from previous features for HTC was hindered by the residual connection.

\(^2\) Except for HiDEC and simplified HiMatch, we encountered out-of-memory with NVIDIA A6000 48GB for other models in training.
Figure 3: Heatmaps of the attention scores in HiDEC on RCV1-v2. (a) and (b) are heatmap of the hierarchy-aware masked self-attention scores and text-hierarchy attention scores, respectively. Attention scores over 0.3 are clipped in all the heatmaps and the similar colors indicate the same level.

Interpretation of Attentions

We investigated the roles of hierarchy-aware masked self-attention and text-hierarchy attention by visualizing the attention scores on RCV1-v2, as shown in Figure 3. The self-attention scores are shown in Figure 3-(a). In (α), the attention score between “(” and “C17” is relatively high where “(” is a starting path from “C17”. The score shows that the special tokens “(” and “)” were appropriately associated with the corresponding labels. In (β), the dependency between child “E21” and parent “ECAT” was well-described because the attention scores for child labels under parent “ECAT” were high. In (γ), the results show a dependency between the label assignment sequence – [“(“, “END“, “)”] and the label “E21”. From these three examples, we can conclude that hierarchy-aware masked self-attention effectively captures the path dependencies. Figure 3-(b) shows the attention scores between the input tokens and a sub-hierarchy sequence. Some tokens, such as the “rating” and “moody,” have high attention scores for the descendants of “CCAT” and itself, where “CCAT” denotes “CORPORATE/INDUSTRIAL”. In contrast, some tokens like “issuer,” “municipal,” and “investors” have high attention scores for the descendants of “EACT” and itself, where “EACT” denotes “ECONOMICS”. This result indicates that the labels are associated with different tokens to different degrees.

Level-Wise Performance

Figure 4 depicts the level-wise performance of the models using a GRU-based text encoder on NYT. It shows the effectiveness of the hierarchy-aware models by comparing TextRCNN increases as the level increases. Among them, HiDEC consistently achieved the best performance at all levels. Note that significant improvements were obtained at the deep levels, implying that sub-hierarchy information is more powerful in capturing the structure information of a target document than the entire hierarchy information.

Conclusion

This paper addressed the scalability limitations of recent HTC models due to the large model size of the structure encoders. To solve this problem, we formulated HTC as a sub-hierarchy sequence generation using an encoder-decoder architecture. Subsequently, we propose Hierarchy DECoder (HiDEC) which recursively decodes the text sequence into a sub-hierarchy sequence by sub-hierarchy decoding while staying aware of the path information. HiDEC achieved state-of-the-art performance with significantly fewer model parameters than existing models on benchmark datasets, such as RCV1-v2, NYT, and EURLEX57K. In the future, we plan to extend the proposed model to extremely large-scale hierarchies (e.g., MeSH term indexing or product navigation) and introduce a novel training strategy combining top-down and bottom-up methods that can effectively use a hierarchy structure.
Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1G1A1003312) and (No. NRF-2021R1I1A3052815).

References


Peng, H.; Li, J.; Wang, S.; Wang, L.; Gong, Q.; Yang, R.; Li, B.; Yu, P. S.; and He, L. 2021. Hierarchical Taxonomy-Aware and Attentional Graph Capsule RCNNs for Large-
Scale Multi-Label Text Classification. *IEEE Transactions on Knowledge and Data Engineering*, 33: 2505–2519.


