HILL: Hierarchy-aware Information Lossless Contrastive Learning for Hierarchical Text Classification

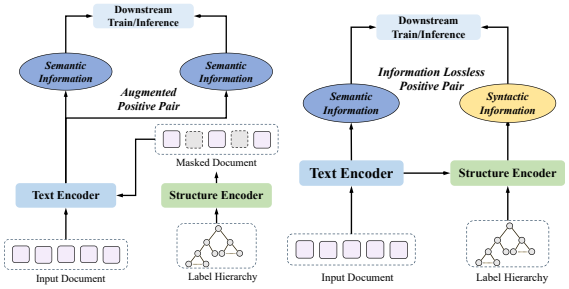
Anonymous ACL submission

Abstract

Existing self-supervised methods in natural language processing (NLP), especially hierarchical text classification (HTC), mainly focus on self-supervised contrastive learning, extremely relying on human-designed augmentation rules to generate contrastive samples, which can potentially corrupt or distort the original information. In this paper, we tend to investigate the feasibility of a contrastive learning scheme in which the semantic and syntactic information inherent in the input sample is adequately reserved in the contrastive samples and fused during the learning process. Specifically, we propose an information lossless contrastive learning strategy for HTC, namely **H**ierarchy-aware **I**nformation **L**ossless contrastive **L**earning (HILL), which consists of a text encoder representing the input document, and a structure encoder directly generating the positive sample. The structure encoder takes the document embedding as input, extracts the essential syntactic information inherent in the label hierarchy with the principle of structural entropy minimization, and injects the syntactic information into the text representation via hierarchical representation learning. Experiments on three common datasets are conducted to verify the superiority of HILL.

1 Introduction

Self-supervised learning (SSL) has exhibited remarkable success across various domains in deep learning, empowering models with potent representation capabilities. Based on these achievements, researchers have incorporated contrastive learning into hierarchical text classification (Wang et al., 2022a), a challenging sub-task within the realm of text multi-label classification. Beyond processing text samples, hierarchical text classification methods should handle a predefined directed acyclic graph in the corpus, referred to as the label hierarchy. While language models such as BERT (Devlin et al., 2019) are pretrained on textual data, their efficacy in handling structural input is limited. Consequently, researchers have introduced a Graph Neural Network (GNN)-based encoder to establish a dual-encoder framework for HTC (Zhou et al., 2020; Deng et al., 2021; Chen et al., 2021).



(a) Previous Methods (b) Ours

Figure 1: Comparison between HILL and previous methods. (a) Previous work use structure encoder in data augmentation. (b) Our method extracting syntactic information in information lossless learning paradigm.

Although the structure encoder contributes to representing the label hierarchy, the dual-encoder framework simply blends the outputs of encoders. In an effort to integrate the label hierarchy into BERT, Wang et al. (2022a) propose a contrastive learning framework, in which BERT functions as a siamese network (Bromley et al., 1993) accepting both the raw text and the masked text, where the mask is generated by the structure encoder. However, their contrastive learning process essentially relies on data augmentation, even with the involvement of the structure encoder in the masking process, as depicted in Figure 1(a). According to the data processing inequality (Cover and Thomas, 2006), applying data augmentation to the raw text may potentially erase sufficient semantic information relevant to downstream prediction.

To maximally preserve the semantic information in the text and effectively leverage the structural encoder in the contrastive learning process for HTC,

we tend to inject the essential information inherent in the label hierarchy into text representations rather than augmenting the input document. As shown in Figure 1(b), our structure encoder directly generates feature vectors by fusing textual and structural information, in contrast to masking the text as illustrated in Figure 1(a). Following the insights of Li and Pan (2016), where structural entropy encodes and decodes the essential structure of the original system to support its semantic analysis. Since hierarchical text classification holds both semantic and syntactic information, we aim to design our model with the guidance of structural entropy. Specifically, we implement a suite of algorithms to minimize the structural entropy of label hierarchies by constructing their coding trees. Subsequently, we design a structure encoder to perform representation learning on coding trees, in which the leaf-node embeddings are initialized by the text encoder while the non-leaf node embeddings are iteratively obtained from bottom to top. Afterward, the structure encoder generates an overall representation of the coding tree, which serves as a contrastive sample for the text encoder. Additionally, we provide a definition of information lossless learning and prove that the information retained by our approach is the upper bound of any augmented data. In comparison with other contrastive and supervised learning methods, our model achieves significant performance gains on three common datasets. Overall, the contributions of our work can be summarized as follows:

- To realize information lossless learning, we decode the essential structure of label hierarchies through the proposed algorithms under the guidance of structural entropy, supporting the semantic analysis for HTC.
- We propose a contrastive learning framework, namely HILL, which fuses the structural information from the label hierarchies into the given document embeddings, while the semantic information from the input document is maximally preserved.
- We define information lossless learning for HTC and prove that the information retained by HILL is the upper bound of any other augmentation-based methods.
- Experiments conducted on three common datasets demonstrate the effectiveness and efficiency of HILL.[1]

[1]The source code will be available after acceptance.

2 Related Works

Hierarchical Text Classification. Existing works for HTC could be categorized into local and global approaches (Zhou et al., 2020). Local approaches build multiple models for labels in different levels in the hierarchy, conveying the information from models in the upper levels to those in the bottom (Kowsari et al., 2017; Shimura et al., 2018; Banerjee et al., 2019; Huang et al., 2019). On the contrary, global studies treat HTC as a flat multi-label classification problem(Gopal and Yang, 2013; You et al., 2019; Aly et al., 2019; Mao et al., 2019; Wu et al., 2019; Rojas et al., 2020).

Recently, Zhou et al. (2020) introduce a dual-encoder framework consisting of a text and a graph encoder to separately handle the text and the label hierarchy. Based on HiAGM (Zhou et al., 2020), Chen et al. (2020a) jointly model the text and labels in the hyperbolic space. Chen et al. (2021) formulate HTC as a semantic matching problem. Deng et al. (2021) introduce information maximization to capture the interaction between text and label while erasing irrelevant information.

Given the success of Pretrained Language Models (PLMs), researchers attempt to utilize their powerful abilities in HTC. Wang et al. (2022a) propose a contrastive learning framework for HTC to make BERT learn from the structure-encoder-controlled text augmentation. Wang et al. (2022b) introduce prompt tuning and construct dynamic templates for HTC. Jiang et al. (2022) encode the global hierarchies with BERT, extract the local hierarchies, and feed them into BERT in a prompt-tuning-like schema. Despite their success, neither existing contrastive learning nor prompt tuning methods try to improve the structure encoder.

Contrastive Learning. Inspired by the pretext tasks in GPT (Radford and Narasimhan, 2018) and BERT (Devlin et al., 2019), researchers originally proposed contrastive learning in computer vision (Chen et al., 2020b; He et al., 2020), addressing the limitations of previous methods in training with massive unlabeled visual data. Numerous studies have shown that the key to contrastive learning lies in the construction of positive pairs (Tian et al., 2020; Caron et al., 2020; Jaiswal et al., 2020; Grill et al., 2020), especially in natural language processing (Wu et al., 2020; Yan et al., 2021; Meng et al., 2021; Pan et al., 2022).

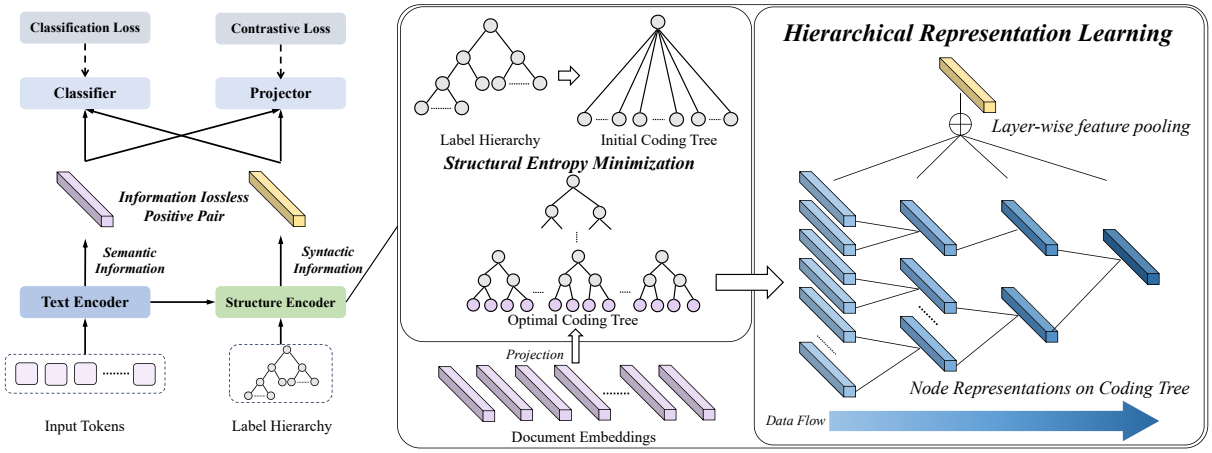


Figure 2: An example of our model with $K = 3$. We first feed the document D into the text encoder to extract the semantic information. Next, the structure encoder takes label hierarchy G_L as input and constructs the optimal coding tree T_L with Algorithm 1 under the guidance of structural entropy. In the hierarchical representation learning module, the leaf node embeddings are initialized by the document embeddings, and the representations of non-leaf nodes are learned from bottom to top. The structure encoder finally generates an information lossless positive view for the text encoder, which is formulated in Section 3.4 and proved in Appendix C.

Structural Entropy. Structural entropy (Li and Pan, 2016) is a natural extension of Shannon entropy (Shannon, 1948) on a structural system, which could measure the structure complexity of the system. Non-Euclidean data, especially graph data, is a typical structured system. The structural entropy of a graph is defined as the average length of the codewords obtained by a random walk under a specific coding scheme, namely coding tree (Li and Pan, 2016). In the past few years, structural entropy has been successfully applied in community security (Liu et al., 2019) graph classification (Wu et al., 2022b,a; Yang et al., 2023), text classification (Zhang et al., 2022), and node clustering (Wang et al., 2023).

3 Methodology

In this section, we first give a problem definition of hierarchical text classification. Next, we elaborate on the working process of the text encoder (in Section 3.2) and the structure encoder (in Section 3.3) of the proposed HILL. Theoretical analysis is further given to reveal the information lossless property of HILL for HTC in Section 3.4. Overall, the framework of HILL is shown in Figure 2.

3.1 Problem Definition

In hierarchical text classification, the label set is predefined and represented as a directed acyclic graph, namely the label hierarchy. Every label that appears in the corpus corresponds to a unique node on the hierarchy. Each non-root node is pointed by only one node in the upper levels, i.e. its parent node. In the ground-truth label set $\mathcal{Y}$ of any sample, a non-root label y_i always co-occurs with its parent nodes, put differently, for any $y_i \in \mathcal{Y}$, the parent node of y_i is also in $\mathcal{Y}$. Given a document D to be classified, where $D = \{w_1, w_2, \ldots, w_N\}$ is commonly treated as a sequence with N tokens, an HTC model should predict a subset $\hat{\mathcal{Y}}$ of the complete label set $\mathbb{Y}$.

3.2 Text Encoder

Our framework is compatible with multiple document representation models. To maintain consistency with previous works, we utilize BERT (Devlin et al., 2019) as the text encoder.

First, the input document D is tokenized into a sequence with N tokens and then padded with two special tokens:

$$\tilde{D} = \{[CLS], w_1, w_2, \ldots, w_N, [SEP]\}, \quad (1)$$

where $[CLS]$ and $[SEP]$ are respectively recognized as the beginning and the end of the document.

Next, the BERT encoder takes the padded document $\tilde{D}$ as input and generates hidden representations of each token, formally, $H_{BERT} = \mathcal{F}_{BERT}(\tilde{D})$, where $H_{BERT} \in \mathbb{R}^{(N+2)\times d_B}$ is the token embedding matrix while $\mathcal{F}_{BERT}(\cdot)$ denotes the holistic BERT model. Afterward, the $[CLS]$ embedding is taken as the representation of the entire document. That is, $h_D = H_{BERT}^{[CLS]} = H_{BERT}^{0}$, where $h_D \in \mathbb{R}^{d_B}$ is the document embedding, and d_B is the hidden size of BERT.

3.3 Structure Encoder

To implement information lossless contrastive learning in the structure encoder, we propose an algorithm to extract structural information from the label hierarchy via structural entropy (Li and Pan, 2016) minimization and a hierarchical representation learning mechanism to inject the structural information into text embeddings. Thereafter, the structure encoder generates a positive view of the document that retains both semantic and structural information losslessly.

Structural Entropy. In Li and Pan (2016), the structural entropy of a graph $G = (V_G, E_G)$ is defined as the average length of the codewords obtained by a random walk under a specific coding pattern named coding tree, formally,

$$H^T(G) = -\sum_{v \in T} \frac{g_v}{vol(G)} \log \frac{vol(v)}{vol(v^+)}, \quad (2)$$

where v is a non-root node of coding tree T which represents a subset of V_G, v^+ is the parent node of v on the coding tree. g_v represents the number of v's cut edges on G. $vol(G)$ denotes the volume of graph G while $vol(v)$ and $vol(v^+)$ is the sum of the degree of nodes partitioned by v and v^+.

The height of the coding tree should be fixed to formulate a certain coding scheme. Therefore, the K-dimensional structural entropy of the graph G determined by the coding tree T with a certain height K is defined as:

$$H_K(G) = \min_{\{T | height(T) \leq K\}} H^T(G). \quad (3)$$

More details about structural entropy and coding trees are provided in Appendix A.

Structural Entropy Minimization. To minimize the structural entropy is to construct the optimal coding tree of graph G. Thus, we design two algorithms to heuristically construct a coding tree T with height K. In Algorithm 1, we take V_G as the leaf nodes, connect them directly to the root node v_r^T, and call Algorithm 2 to construct an initial coding tree T. [2] Algorithm 2 creates a new coding tree $\mathbf{T}$ of height 1 with $\mathbf{v}$ and iteratively compresses two child nodes from the children set $C(\mathbf{v})$ of root node $\mathbf{v}$ in the first *while* loop (lines 3-6), prioritizing the nodes that result in the largest reduction in structural entropy. Since tree $\mathbf{T}$'s height may exceed K, in the second *while* loop (lines 7-10), we iteratively remove non-leaf nodes of $\mathbf{T}$, prioritizing nodes with the smallest entropy increase upon deletion. Afterward, leaf nodes of $\mathbf{T}$ might have different heights, contradicting the definition of coding trees. Thus, we adopt the operation in line 11 to align leaf nodes. Algorithm 2 always returns a coding tree $\mathbf{T}$ with height 2. Algorithm 1 will iteratively invoke Algorithm 2 until the height of T reaches K. More precisely, each iteration within the *while* loop will increment the height of T by 1 by calling Algorithm 2 on the root node v_r^T or on all nodes in V_T^1, depending on the reduction in structural entropy[3]. More details about the proposed algorithms can be found in Appendix B.

Algorithm 1 Greedy Coding Tree Construction

Input: A graph $G = (V_G, E_G)$ and a positive integer K.
Output: Coding tree $T = (V_T, E_T)$ of the graph G with height K.

1: $V_T^1 := \{v_r^T\}, V_T^0 := C(v_r^T) := V_G$;
2: $T = Algo.\ 2(\mathbf{v} := v_r^T)$;
3: **while** $T.height < K$ **do**
4: $\quad T_1 = T.merge(Algo.\ 2(\mathbf{v} := v_r^T))$;
5: $\quad T_2 = T.merge(\{\mathbf{T} = Algo.\ 2(\mathbf{v} := \hat{v}) | \forall \hat{v} \in V_T^1\})$;
6: $\quad T = (H^{T_1}(G) < H^{T_2}(G))\ ?\ T_1 : T_2$;
7: **end while**
8: **return** T;

Hierarchical Representation Learning. After calling Algorithm 1($G_L = (V_G, E_G), K$), we get a coding tree $T_L = (V_{T_L}, E_{T_L})$ of label hierarchy G_L with $T_L.height = K$. For representation learning, reformulate the label hierarchy and its coding tree as triplets: $G_L = (V_{G_L}, E_{G_L}, X_{G_L})$, $T_L = (V_{T_L}, E_{T_L}, X_{T_L})$ where $X_{G_L} \in \mathbb{R}^{\mathbb{Y} \times d_V}$ is derived from document embedding h_D via two-dimensional projector, formally, $X_{G_L} = \phi_{proj}(h_D)$. $\phi_{proj}(\cdot)$ consists of a $(\mathbb{Y} \times 1)$ and a $(d_B \times d_V)$ feed-forward network, where d_V is the hidden size for vertices. Meanwhile, $X_{T_L} = \{X_{T_L}^0, X_{T_L}^1, \ldots, X_{T_L}^K\}$ represents the node embeddings of $V_{T_L}^i, i \in [0, K]$.

In Algorithm 1, the leaf nodes $V_{T_L}^0$ are initialized with V_{G_L}, thus $X_{T_L}^0 := X_{G_L}$. Furthermore, $\{V_{T_L}^k | k \in [1, K]\}$ is given by Algorithm 1 while

[2] Due to different scopes, we use T and $\mathbf{T}$ to distinguish the coding trees in Algorithm 1 and Algorithm 2.

[3] In the implementation, we apply pruning strategies to improve the efficiency, but they are omitted here for clarity.

Algorithm 2 2-level sub-coding tree construction.

Input: A node $\mathbf{v}$.

Output: Coding tree $\mathbf{T} = (V_{\mathbf{T}}, E_{\mathbf{T}})$ with height 2.

$v_r^{\mathbf{T}} := \mathbf{v}, V_{\mathbf{T}}^0 := C(\mathbf{v})$;
$\forall v \in V_{\mathbf{T}}^0, v.parent := v_r^{\mathbf{T}}, C(v_r^{\mathbf{T}}) := v \cup C(v_r^{\mathbf{T}})$
while $|C(v_r^{\mathbf{T}})| > 2$ **do**
  $(v_\alpha, v_\beta) = \underset{(v,v')}{argmax}\{H^{\mathbf{T}}(G) - H^{\mathbf{T}.compress(v,v')}(G)\}$
  $\mathbf{T}.compress(v_\alpha, v_\beta)$
end while
while $\mathbf{T}.height > 2$ **do**
  $v_i = \underset{v}{argmin}\{H^{\mathbf{T}.remove(v)}(G) - H^{\mathbf{T}}(G)\}$
  $\mathbf{T}.remove(v_i)$
end while
$\mathbf{T}.align()$
return $\mathbf{T}$

their node embeddings $\{X_{T_L}^k | k \in [1, K]\}$ need to be fetched. Based on the structure of coding trees, we design a hierarchical representation learning module. For $x_v^k \in X_{T_L}^k$ in the k-th layer,

$$x_v^k = \phi_{FFN}^k(\sum\nolimits_{n \in C(v)} x_n^{k-1}), \tag{4}$$

where $v \in V_T^k$, $x_v^k \in \mathbb{R}^{d_V}$ is the feature vector of node v, and $C(v)$ represents the child nodes of v in coding tree T_L. $\phi_{FFN}^i(\cdot)$ denotes a feed-forward network. The information from leaf nodes propagates layer by layer until it reaches the root node. Finally, to capture the multi-granular information provided by nodes at different levels, we utilize Equation 5 to integrate information from each layer of T_L:

$$h_T = \bigsqcup_{k=1}^{K} \eta(\{x_v^k | v \in V_{T_L}^k\}), \tag{5}$$

where $\bigsqcup(\cdot)$ indicates the concatenation operation. $\eta(\cdot)$ could be a feature-wise summation or averaging function. $h_T \in \mathbb{R}^{d_T}$ is the final output of the structure encoder.

3.4 Contrastive Learning Module

Positive Pairs and Contrastive Loss. We expect the text encoder and the structure encoder to learn from each other. Thus, the document embedding h_D and structural embedding h_T of the same sample form the positive pair in our model. Considering h_D and h_T might be in different distributions, we first project them into an embedding space via independent projectors, formally,

$$h = W_D^2 ReLU(W_D^1 \cdot h_D + b_D) \tag{6}$$

$$\hat{h} = W_T^2 ReLU(W_T^1 \cdot h_T + b_T), \tag{7}$$

where h and $\hat{h}$ are the projected vectors of h_D and h_T. W_D^1, W_D^2, b_D, W_T^1, W_T^2, and b_T are weights of projectors.

Next, we utilize NT-Xent loss (Chen et al., 2020b) to achieve contrastive learning. Let h_i and $\hat{h_i}$ denote the projected positive pair of the i-th document in a batch, the contrastive learning loss L_{clr} is formulated as:

$$L_{clr} = -log \frac{e^{\varepsilon(h_i, \hat{h_i})}/\tau}{\sum_{j=1, j \neq i}^{|\mathbf{B}|} e^{\varepsilon(h_j, \hat{h_j})/\tau}}, \tag{8}$$

where $|\mathbf{B}|$ denotes the batch size, τ is the temperature parameter, and $\varepsilon(h, \hat{h})$ is a distance metric implemented by cosine similarity $\frac{h \top \hat{h}}{||h|| \cdot ||\hat{h}||}$.

Information Lossless Contrastive Learning for HTC. Information lossless learning dose not imply retaining any input data. It is preserving the minimal but sufficient information, i.e., the mutual information required by the downstream task. This is how we define information lossless learning for hierarchical text classification:

Definition 1 *In HTC, the mutual information between inputs and targets can be written as:*

$$I((\mathcal{G}_{\mathcal{L}} \circ \mathcal{D}); Y), \tag{9}$$

where $\mathcal{G}_{\mathcal{L}} \in \mathbb{G}, \mathcal{D} \in \mathbb{D}$ are random variables for the label hierarchy G_L and the document D. $I(X_1; X_2)$ denotes the mutual information between random variables X_1 and X_2. $\circ$ indicates any input combination of $\mathcal{G}_{\mathcal{L}}$ and $\mathcal{D}$

Definition 2 *Given a function $\mathcal{F} \subseteq \mathbb{F}_{\mathbb{T}} \times \mathbb{F}_{\mathbb{G}}$, which is an arbitrary fusion of a text and a structure encoder. Define the optimal function $\mathcal{F}^*$ if and only if $\mathcal{F}^*$ satisfies:*

$$\mathcal{F}^* = \underset{\mathcal{F} \subseteq \mathbb{F}_{\mathbb{T}} \times \mathbb{F}_{\mathbb{G}}}{argmax} I(\mathcal{F}(\mathcal{G}_{\mathcal{L}} \circ \mathcal{D}); (\mathcal{G}_{\mathcal{L}} \circ \mathcal{D})). \tag{10}$$

That is, $\mathcal{F}^*$retains the most mutual information between the input random variables and the encoded random variables. Apparently, $\mathcal{F}^*$ is a deterministic mapping as the embedding $\mathcal{F}^*(G_L \circ D)$ is fixed for downstream prediction when given $(G_L \circ D)$. Thus, for any random variable ξ,

$$I(\mathcal{F}^*(\mathcal{G}_{\mathcal{L}} \circ \mathcal{D}); \xi) = I((\mathcal{G}_{\mathcal{L}} \circ \mathcal{D}); \xi). \tag{11}$$

When $\xi = Y$, we could have,

$$I(\mathcal{F}^*(\mathcal{G}_\mathcal{L} \circ \mathcal{D}); Y) = I((\mathcal{G}_\mathcal{L} \circ \mathcal{D}); Y). \quad (12)$$

Theorem 1 *Given a document D and the coding tree T_L of the label hierarchy G_L. Denote their random variable as $\mathcal{D}$, $\mathcal{T}_\mathcal{L}$, and $\mathcal{G}_\mathcal{L}$. For any augmentation function θ, we have,*

$$I((\mathcal{T}_\mathcal{L} \circ \mathcal{D}); Y) \geq I(\theta(\mathcal{G}_\mathcal{L}, \mathcal{D}); Y). \quad (13)$$

The proof for Theorem 1 is given in Appendix C.

Supervised Contrastive Learning. Following the training strategy of previous contrastive learning methods for HTC, we train HILL with classification loss and contrastive loss simultaneously. After executing the structure encoder, the document embedding h_D and its positive view h_T are concatenated and fed into the classifier along with the sigmoid function:

$$P = Sigmoid([h_D; h_T] \cdot W_c + b_c), \quad (14)$$

where $W_c \in \mathbb{R}^{(d_B+d_T)\times|\mathbb{Y}|}$ and $b_c \in \mathbb{R}^{|\mathbb{Y}|}$ are weights and bias of the classifier while $|\mathbb{Y}|$ is the volume of the label set. For multi-label classification, we adopt the Binary Cross-Entropy Loss as the classification loss:

$$L_C = -\frac{1}{|\mathbb{Y}|}\sum_{j}^{|\mathbb{Y}|} y_j log(p_j) + (1 - y_j) log(1 - p_j), \quad (15)$$

where y_j is the ground truth of the j-th label while p_j is the j-th element of P.

Overall, the final loss function can be formulated as:

$$L = L_C + \lambda_{clr} \cdot L_{clr}. \quad (16)$$

where λ_{clr} is the weight of L_{clr}.

Dataset	$\|\mathbb{Y}\|$	$Avg(\|\mathcal{Y})\|)$	Depth	# Train	# Dev	# Test
WOS	141	2.0	2	30,070	7,518	9,397
RCV1-v2	103	3.24	4	20,833	2,316	781,265
NYTimes	166	7.6	8	23,345	5,834	7,292

Table 1: Summary statistics of the three datasets.

4 Experiment

4.1 Experiment Setup

Datasets and Evaluation Metrics. Experiments are conducted on three popular datasets in HTC. RCV1-v2 (Lewis et al., 2004) and NYTimes (Sandhaus, Evan, 2008) consists of news articles, while WOS (Kowsari et al., 2017) includes abstracts of academic papers. Each of these datasets is annotated with ground-truth labels existing in a predefined hierarchy. We split and preprocess these datasets following (Wang et al., 2022a). The statistics of these datasets are shown in Table 1. The experimental results are measured with Micro-F1 and Macro-F1 (Gopal and Yang, 2013). Micro-F1 is the harmonic mean of the overall precision and recall of all the test instances, while Macro-F1 is the average F1-score of each category. Thus, Micro-F1 reflects the performance on more frequent labels, while Macro-F1 treats labels equally.

Implementation Details. For the text encoder, we use the BertModel of `bert-base-uncased` and update its parameters with Adam (Kingma and Ba, 2015) as the initial learning rate is 3e-5. For all three datasets, the batch size is set to 24, the hidden sizes d_B, d_V, d_T are all set to 768. ϕ^k_{FFN} in Equation 4 is implemented by K independent multi-layer perceptions which consists of 2-layer linear transformations and non-linear functions. The $\eta(\cdot)$ in Equation 5 is a summation. The learning rate of the structure encoder is set to 1e-3 for WOS, 1e-4 for RCV1 and NYTimes. The weight of contrastive loss λ_{clr} is respectively set to 0.001, 0.1, 0.3 for WOS, RCV1, and NYTimes. The optimal height K of coding trees goes to 3, 2, and 3.

Baselines. We compare HILL with self-supervised only models including HiAGM(Zhou et al., 2020), HTCInfoMax (Deng et al., 2021), HiMatch (Chen et al., 2021) and their BERT-based version and a contrastive learning model HGCLR (Wang et al., 2022a). HiAGM, HTCInfoMax, and HiMatch use different fusion strategies to model text-hierarchy correlations. Specifically, HiAGM proposes a multi-label attention (HiAGM-LA) and a text feature propagation technique (HiAGM-TP) to get hierarchy-aware representations. HTCInfoMax enhances HiAGM-LA with information maximization to model the interaction between text and hierarchy. HiMatch treats HTC as a matching problem by mapping text and labels into a joint embedding space. HGCLR makes BERT learn from the structure encoder with a controlled document augmentation.

4.2 Results and Analysis

The main results are presented in Table 2. In the supervised-only models, HTCInfoMax enhances HiAGM through the maximization of mutual information principles. HiMatch treats HTC as a

Model	WOS		RCV1-v2		NYTimes		Average	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Supervised Learning Models								
TextRCNN (Zhou et al., 2020)	83.55	76.99	81.57	59.25	70.83	56.18	78.65	64.14
HiAGM (Zhou et al., 2020)	85.82	80.28	83.96	63.35	74.97	60.83	81.58	68.15
HTCInfoMax (Deng et al., 2021)	85.58	80.05	83.51	62.71	-	-	-	-
HiMatch (Chen et al., 2021)	86.20	80.53	84.73	64.11	-	-	-	-
Supervised Learning Models (BERT-based)								
BERT †	85.63	79.07	85.65	67.02	78.24	65.62	83.17	70.57
BERT (Chen et al., 2021)	86.26	80.58	86.26	67.35	-	-	-	-
BERT+HiAGM †	86.04	80.19	85.58	67.93	78.64	66.76	83.42	71.67
BERT+HTCInfoMax †	86.30	79.97	85.53	67.09	78.75	67.31	83.53	71.46
BERT+HiMatch †	86.70	81.06	86.33	68.66	-	-	-	-
Contrastive Learning Models								
HGCLR (Wang et al., 2022a)	87.11	81.20	86.49	68.31	78.86	67.96	84.15	72.49
HILL(Ours)	**87.28**	**81.77**	**87.31**	**70.12**	**80.47**	**69.96**	**85.02**	**73.95**

Table 2: Experimental results of our proposed model on three datasets and their average performance. The supervised learning models (in the upper part) originally take TextRCNN (Lai et al., 2015) as the text encoder. For fairness, we compared with their BERT-based versions implemented by Wang et al. (2022a) (in the middle part). The best results are marked in **bold**. "-" means not reported or not applicable.

matching problem, and the reported results stand out as the best among these models. The replacement of TextRCNN with BERT has minimal impact on the relative ranking of their outcomes. This implies that the text encoder primarily influences the overall effectiveness of their models, while their specific merits are determined by their efforts beyond the text encoder. To some extent, it also indicates that their text encoder and structure encoder operate independently.

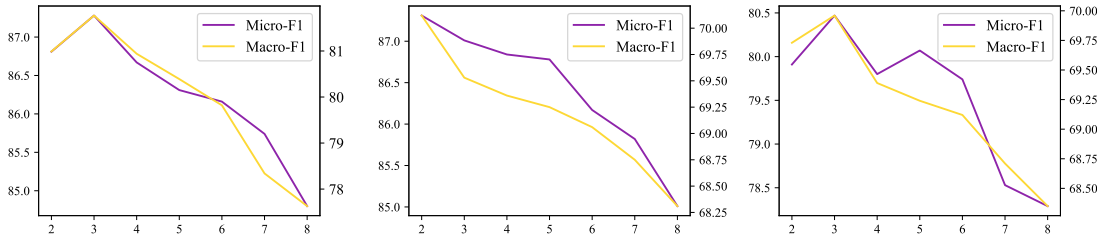
HGCLR is the inaugural contrastive learning approach for HTC. Despite the involvement of the structure encoder, the positive sample construction in HGCLR still relies on data augmentation. Conversely, our model effectively utilizes the syntactic information extracted by the structure encoder, enabling cooperation between the text encoder and the structure encoder. Specifically, the proposed HILL surpasses all supervised learning models and the contrastive learning model across all three datasets. Our model demonstrates average improvements of 1.85% and 3.38% on Micro-F1 and Macro-F1 compared to vanilla BERT. In comparison with HGCLR, our model achieves nearly a 2% performance boost on both RCV1-v2 and NYTimes. Additionally, the improvement in WOS is notable, though slightly less than that observed in the other two datasets. A likely reason is that BERT is pretrained on news corpus, and the domain knowledge acquired may differ from that of paper abstracts. However, this could also indicate effective collaboration between the text encoder and the structural encoder in our framework. As the text encoder learns robust features, the structural encoder becomes increasingly powerful, and vice versa.

Ablation Models	RCV1-v2	
	Micro-F1	Macro-F1
HILL	87.31	70.12
r.p. GIN (Xu et al., 2019)	86.48	69.30
r.p. GAT (Velickovic et al., 2018)	86.51	68.12
r.p. GCN (Kipf and Welling, 2017)	86.24	68.71
r.m. L_{clr}	86.51	68.60
r.m. Algorithm 1	86.67	67.92

Table 3: Performance when replacing or removing some components of HILL on the test set of RCV1-v2. r.p. stands for the replacement and r.m. stands for remove. The results of r.m. L_{clr} and r.m. Algorithm 1 are both obtained from 5 runs under different random seeds, each of which is distinguished from HILL's at a significant level of 95% under a one-sample t-test.

4.3 Ablation Studies

The necessity of proposed methods. We conduct ablation studies by removing one component of our model at a time while keeping other conditions consistent whenever possible. The results of the ablation studies are presented in Table 3. To demonstrate the effectiveness of proposed hierarchical representation learning, we replaced the structure encoder with three commonly used graph neural networks including GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018), and GIN (Xu et al., 2019). All of them are fed with the label hierarchy G_L and the document embedding h_D to initialize node embeddings. For GCN and GAT, the number of layers is set to 2 while other parameters



(a) WOS (b) RCV1-v2 (c) NYTimes

Figure 3: Test performance of HILL with different height K of the coding tree on three datasets.

are the default settings in PyTorch-Geometric (Fey and Lenssen, 2019). Regarding GIN, the combine function is a 2-layer multi-layer perception with $\epsilon = 0$, and the iteration is set to 3. We find that the hierarchical learning module outperforms all graph encoders on RCV1-v2, which empirically proves that syntactic information extraction is successful in HILL. Moreover, the performance of the supervised-only model (r.m. L_{clr}) declines by 0.92% and 2.17%, underscoring the necessity of contrastive learning. Additionally, we directly feed the initial coding tree, i.e., a coding tree within the root node v_r connecting to the leaf nodes, into the structure encoder. The model (r.m. Algorithm 1) exhibits performance decreases of 0.73% and 3.14%, emphasizing the effectiveness of structural entropy minimization.

The Height K of Coding Trees. The height of the coding tree affects the performance of HILL. Higher coding trees may involve more explosive gradients. To investigate the impact of K, we run HILL with different heights K of the coding tree while keeping other settings the same. Figure 3 shows the test performance of different height coding trees on WOS, RCV1-v2, and NYTimes. As K grows, the performance of HILL sharply degrades. The optimal K seems to be unrelated to the heights of label hierarchies, since the heights of the three datasets are 2, 4, and 8, while the optimal K is 3, 2, and 3. On the contrary, the optimal K is more likely to positively correlate with the volumes of label set $\mathbb{Y}$, 141, 103, and 166.

Model	Trainable Parameters (M)			Training Time Per Epoch (s)		
	WOS	RCV1-v2	NYTimes	WOS	RCV1-v2	NYTimes
Ours	7.35	7.27	7.40	1005.84	639.82	721.98
HGCLR	19.07	18.80	19.27	1898.65	1224.74	1390.60

Table 4: The number of trainable parameters and the average training time of our model and HGCLR on WOS, RCV1-v2, and NYTimes.

Time-and-Memory-Saving Contrastive Learning. The structure encoder of HILL is considerably smaller than that of HGCLR, as the kernel of hierarchical representation learning consists of only K multi-layer perceptions, while Graphormer (Ying et al., 2021) is built upon multi-head attention. In this comparison, we assess the number of learnable parameters and the training speed of HILL in contrast to HGCLR. Specifically, we set the hidden state sizes of both HILL and HGCLR to 768, with batch sizes set to 16. The count of trainable parameters is determined by the $numel(\cdot)$ function in PyTorch (Paszke et al., 2019), excluding those related to BERT. As indicated in Table 4, our model exhibits significantly fewer parameters than HGCLR, averaging 7.34M compared to 19.04M. Additionally, the training speed is evaluated after 20 epochs of training[4]. The average training time for HILL is 789.2s, which is half the time taken by HGCLR (1504.7s). Overall, this analysis suggests that the efficient architecture of HILL contributes to its status as a time- and memory-saving model.

5 Conclusion

In this paper, we design a suite of methods to address the limitations of existing contrastive learning models for HTC. In particular, we propose HILL, in which the syntactic information is sufficiently fused with the semantic information after structural entropy minimization and hierarchical representation learning. Theoretically, we give the definition on information lossless learning for HTC and the information extracted by HILL is proved to be the upper bound of other contrastive learning based methods. Experimental results demonstrate the effectiveness and efficiency of our model against state-of-the-arts.

[4]Both of them converge around the 20th epoch.

Limitations

In Table 2, we only provide the results of HILL and HGCLR when using BERT as the text encoder. Due to the focus on designing the structure encoder, we do not report results on a smaller model, for instance, TextRCNN, or a larger language model as the text encoder. On the other hand, we adopt supervised contrastive learning in accordance with the settings of HGCLR. The performance of HILL under contrastive-supervised two-stage training remains to be explored.

References

Rami Aly, Steffen Remus, and Chris Biemann. 2019. Hierarchical multi-label classification of text with capsule networks. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 2: Student Research Workshop*, pages 323–330. Association for Computational Linguistics.

Siddhartha Banerjee, Cem Akkaya, Francisco Perez-Sorrosal, and Kostas Tsioutsiouliklis. 2019. Hierarchical transfer learning for multi-label text classification. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6295–6300. Association for Computational Linguistics.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. In *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, pages 737–744. Morgan Kaufmann.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Boli Chen, Xin Huang, Lin Xiao, Zixin Cai, and Liping Jing. 2020a. Hyperbolic interaction model for hierarchical multi-label classification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7496–7503. AAAI Press.

Haibin Chen, Qianli Ma, Zhenxi Lin, and Jiangyue Yan. 2021. Hierarchy-aware label semantics matching network for hierarchical text classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4370–4379. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020b. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Thomas M. Cover and Joy A. Thomas. 2006. *Elements of information theory (2. ed.)*. Wiley.

Zhongfen Deng, Hao Peng, Dongxiao He, Jianxin Li, and Philip S. Yu. 2021. Htcinfomax: A global model for hierarchical text classification via information maximization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3259–3265. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap your own latent - A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *2020*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 9726–9735. Computer Vision Foundation / IEEE.

Wei Huang, Enhong Chen, Qi Liu, Yuying Chen, Zai Huang, Yang Liu, Zhou Zhao, Dandan Zhang, and Shijin Wang. 2019. Hierarchical multi-label text classification: An attention-based recurrent network approach. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A survey on contrastive self-supervised learning. *ArXiv*, abs/2011.00362.

Ting Jiang, Deqing Wang, Leilei Sun, Zhongzhi Chen, Fuzhen Zhuang, and Qinghong Yang. 2022. Exploiting global and local hierarchies for hierarchical text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 4030–4039. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, K. Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. Hdltex: Hierarchical deep learning for text classification. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 364–371.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2267–2273. AAAI Press.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397.

Angsheng Li and Yicheng Pan. 2016. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*, 62:3290–3339.

Yiwei Liu, Jiamou Liu, Zijian Zhang, Liehuang Zhu, and Angsheng Li. 2019. REM: from structural entropy to community structure deception. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12918–12928.

Yuning Mao, Jingjing Tian, Jiawei Han, and Xiang Ren. 2019. Hierarchical text classification with reinforced label assignment. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 445–455. Association for Computational Linguistics.

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. COCO-LM: correcting and contrasting text sequences for language model pretraining. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 23102–23114.

Lin Pan, Chung-Wei Hang, Avirup Sil, and Saloni Potdar. 2022. Improved text classification via contrastive adversarial training. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 11130–11138. AAAI Press.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Kervy Rivas Rojas, Gina Bustamante, Arturo Oncevay, and Marco Antonio Sobrevilla Cabezudo. 2020. Efficient strategies for hierarchical text classification: External knowledge and auxiliary tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2252–2257. Association for Computational Linguistics.

Sandhaus, Evan. 2008. The new york times annotated corpus.

Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423.

Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. 2018. HFT-CNN: learning hierarchical category structure

for multi-label short text categorization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 811–816. Association for Computational Linguistics.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. 2020. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yifei Wang, Yupan Wang, Zeyu Zhang, Song Yang, Kaiqi Zhao, and Jiamou Liu. 2023. USER: unsupervised structural entropy-based robust graph neural network. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 10235–10243. AAAI Press.

Zihan Wang, Peiyi Wang, Lianzhe Huang, Xin Sun, and Houfeng Wang. 2022a. Incorporating hierarchy into text encoder: a contrastive learning approach for hierarchical text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 7109–7119. Association for Computational Linguistics.

Zihan Wang, Peiyi Wang, Tianyu Liu, Binghuai Lin, Yunbo Cao, Zhifang Sui, and Houfeng Wang. 2022b. HPT: hierarchy-aware prompt tuning for hierarchical text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 3740–3751. Association for Computational Linguistics.

Jiawei Wu, Wenhan Xiong, and William Yang Wang. 2019. Learning to learn and predict: A meta-learning approach for multi-label classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4353–4363. Association for Computational Linguistics.

Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. 2022a. Structural entropy guided graph hierarchical pooling. In *International Conference on Machine Learning*, pages 24017–24030. PMLR.

Junran Wu, Shangzhe Li, Jianhao Li, Yicheng Pan, and Ke Xu. 2022b. A simple yet effective method for graph classification. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 3580–3586. ijcai.org.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. CLEAR: contrastive learning for sentence representation. *CoRR*, abs/2012.15466.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5065–5075. Association for Computational Linguistics.

Zhenyu Yang, Ge Zhang, Jia Wu, Jian Yang, Quan Z. Sheng, Hao Peng, Angsheng Li, Shan Xue, and Jianlin Su. 2023. Minimum entropy principle guided graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pages 114–122. ACM.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 28877–28888.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5812–5822.

Chong Zhang, He Zhu, Xingyu Peng, Junran Wu, and Ke Xu. 2022. Hierarchical information matters: Text

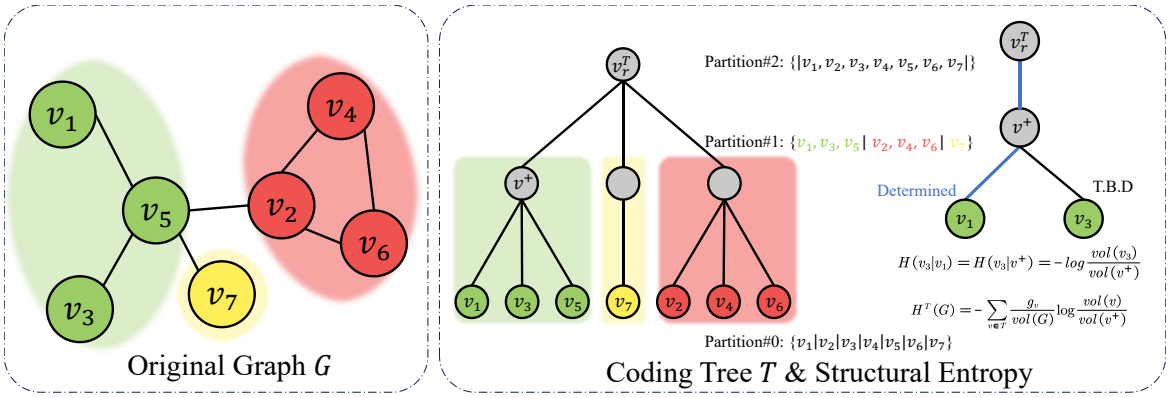


Figure 4: An illustration of coding trees and structural entropy. The coding tree T provides us with multi-granularity partitions of the original graph G, as shown by the three partitions in the example. Structural entropy is defined as the average amount of information of a random walk between two nodes in V_G, considering all nodes partitioned (encoded and decoded) by coding tree T. Under the guidance of structural entropy, coding tree T could reveal the essential structure of graph G.

classification via tree based graph neural network. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 950–959. International Committee on Computational Linguistics.

Jie Zhou, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and Gongshen Liu. 2020. Hierarchy-aware global model for hierarchical text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1106–1117. Association for Computational Linguistics.

A A Brief Introduction to Structural Entropy and Coding Trees

The exhaustive definitions and theorems of coding trees and structural entropy are originated by Li and Pan (2016). Here, we just briefly introduce some key concepts which are crucial to this paper. An illustration of coding trees and structural entropy is shown in Figure 4.

Coding tree. A coding tree $T = (V_T, E_T)$ of graph $G = (V_G, E_G)$ is a tree that satisfies:

I. Denote v_r^T as the unique root node of T, V_ζ as the leaf-node set of T, and $T.height$ the height of T. The node set of T is V_T, which consists of a few subsets, that is, $V_T = \{V_T^0, V_T^1, \ldots, V_T^{T.height}\}$. $V_T^{T.height} := v_r^T, V_T^0 := V_\zeta$.

II. Each node $v \in V_T$ is the **marker** (Li and Pan, 2016) of a subset of V_G. For instance, v_r^T is the marker V_G, while each $v_\zeta \in V_\zeta$ marks a single node in V_G.

III. For any $V \in \{V_T^0, V_T^1, \ldots, V_T^{T.height}\}$. All nodes in V has the same height (depth) and any subset of V represents a **partition** of V_G. Specifically, $V_T^0 = V_\zeta$ is an element-wise partition for V_G as there exists a one-to-one correspondence for nodes in V_ζ and V_G. While $V_T^{T.height} = v_r^T$ is the overall partition for V_G as v_r^T marks the integrity of V_G.

Structural Entropy. As coding trees can be regarded as coding patterns for graphs, the structural entropy of a graph is defined as the average amount of information under a determined coding tree. Specifically, as depicted in Figure 4, when a random walk on graph G progresses from node v_1 to node v_3, a portion of the information is encoded by their parent node v^+, and at this point, only the information from $v+$ to v_3 remains to be determined. Meanwhile, the conditional entropy $H(v_3|v_1)$ is then reduced to $H(v_3|v^+)$. [5] Thus, we have:

$$H(v_3|v_1) = H(v_3|v^+) = -log\frac{vol\,(v_3)}{vol\,(v^+)}, \quad (17)$$

where $vol(v_3)$ denotes the degree of v_3 while (v^+) is the total degree of its children, i.e. $vol(v^+) = vol(v_1) + vol(v_3) + vol(v_5)$. Structural entropy of a graph G is defined as the average amount of information required to determine during a random walk between two accessible nodes. According to

[5] Conditional entropy should be defined with random variables, but we omit them here for simplicity.

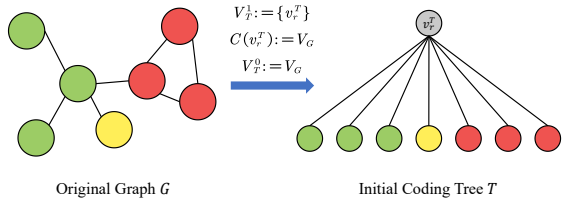
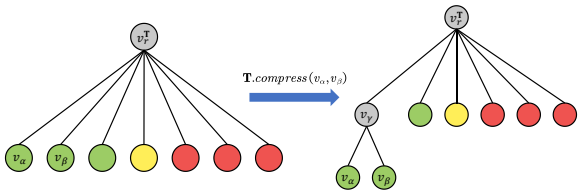


Figure 5: The initialization stage of Algorithm 1, in which a 1-height coding tree is constructed.

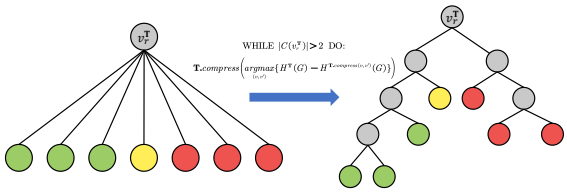
the derivation procedure provided by Li and Pan (2016), we have:

$$H^T(G) = -\sum_{v \in T} \frac{g_v}{vol(G)} \log \frac{vol(v)}{vol(v^+)}, \qquad (18)$$

where g_v represents the number of v's cut edges on G, $vol(G)$ denotes the volume of graph G, and $\frac{g_v}{vol(G)}$ indicates the probability of a random walk on G involving the leaf nodes marked by v.



(a) An example of compressing leaf node v_α and v_β.



(b) The final result after executing lines 3-6.

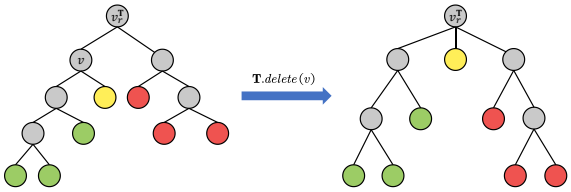
Figure 6: An illustration of $\mathbf{T}.compress(\cdot)$ operation. (a) A single execution of $\mathbf{T}.compress(v_\alpha, v_\beta)$. (b) The final state after executing lines 3-6 in Algorithm 2.

B Explanations for The Proposed Algorithms

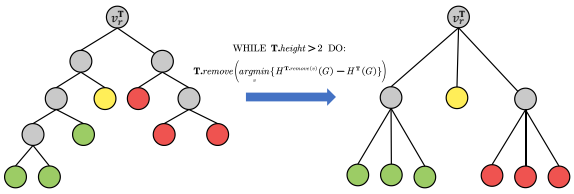
Definitions for Algorithm 1 and Algorithm 2. Given a coding tree $T = (V_T, E_T)$, we define some attributes and member functions of T as follows,

A. Given any two nodes $v_\alpha, v_\beta \in V_T$. If $(\alpha, \beta) \in E_T$, call v_α the parent of v_β, and v_β a child of v_α, which is denoted as $v_\beta \in C(v_\alpha)$, $v_\beta.parent = v_\alpha$.

B. Function $compress(\cdot, \cdot)$. As illustrated in Figure 6(a), given any $v_\alpha, v_\beta \in C(v_r^T)$. $compress(v_\alpha, v_\beta)$ will spawn a new node v_γ, remove v_α, v_β from $C(v_r^T)$, make v_γ be their parent, and add v_γ to $C(v_r^T)$. After that, $v_\alpha.parent = v_\beta.parent = v_\gamma$, and $v_\gamma \in C(v_r^T)$ while $v_\alpha, v_\beta \notin C(v_r^T)$.



(a) An example of deleting an intermediate node v.



(b) The final result after executing lines 7-10.

Figure 7: An illustration of $\mathbf{T}.delete(\cdot)$ operation. (a) A single execution of $\mathbf{T}.delete(v)$. (b) The final state after executing lines 7-10 in Algorithm 2.

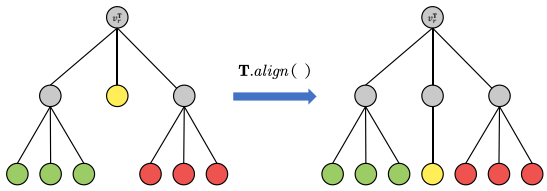


Figure 8: An illustration of $\mathbf{T}.align()$ operation. $\mathbf{T}.align()$ will align the height (depth) of all the leaf nodes to satisfy the definition of coding trees.

C. Function $delete(\cdot)$. As depicted in Figure 7(a), given any $v \in V_T, v \neq v_r$. $delete(v)$ could remove v from V_T and attach all its children to the parent of v. That is, $\forall v_\mu \in C(v), v_\mu.parent := v.parent$.

D. Function $align()$. For any leaf node $v_\zeta \in V_\zeta$, $align()$ will insert a new node between v_ζ and $v_\zeta.parent$ until the depth of v_ζ reaches K. $align()$ ensures that all leaf nodes of T reach the same height (depth) K thereby satisfying the definition of a coding tree. Figure 8 shows an example of $align()$ operation.

E. Function $merge(\cdot)$. For any node $\hat{v} \in V_T$, $Algorithm\ 1(\hat{v})$ returns a new coding tree $\mathbf{T}$ with height equals to 2. $T.merg(\mathbf{T})$ will replace the sub-tree of T rooted by $\hat{v}$ with $\mathbf{T}$.

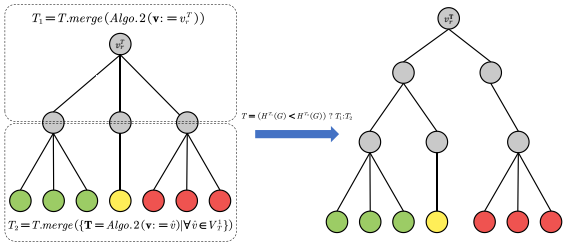


Figure 9: An illustration of lines 3-7 in Algorithm 1.

Note that all the operations above update E_T accordingly. In no case does E_T contain self-loops or skip connections. That is, for any $(v_\alpha, v_\beta) \in E_T$, $|v_\alpha.height - v_\beta.height| \equiv 1$.

Illustrations for Algorithm 1 and Algorithm 2. Here, we present several diagrams to deliver a running example of proposed algorithms.

As shown in Figure 5, the original graph G is fed into Algorithm 1 and initialized as a 1-height coding tree T, in which all nodes in V_G are treated as leaf node and directly connect to a new root node v_r^T. Thereafter, Algorithm 1 will call Algorithm 2 several times to construct a coding tree of the specified height K.

Each invocation of Algorithm 2 takes a non-leaf node $\mathbf{v}$ in tree T as input and yields a (sub-)coding tree $\mathbf{T}$ with a height of 2 wherein $\mathbf{v}$ acting as the root. Algorithm 2 first initializes $\mathbf{T}$ in a similar procedure to that illustrated in Figure 5. Subsequently, in the first $while$ loop (lines 3-6), we systematically compress the structural entropy by iteratively combining two children of root node $\mathbf{v}$ with $\mathbf{T}.compress(\cdot,\cdot)$, prioritizing those nodes resulting in the largest structural entropy reduction. Ultimately, the maximal reduction in structural entropy is achieved, resulting in the creation of a full-height binary tree, as depicted in Figure 6(b). Given that the full-height binary tree may exceed the specified height of 2, we rectify this by condensing the tree through the invocation of $\mathbf{T}.delete(\cdot)$ in the second $while$ loop (lines 7-10). The complete deletion process is illustrated in Figure 7(b). It is important to note that after condensation, tree $\mathbf{T}$ comprises leaf nodes with varying heights, which violates the definition of coding trees. To address this, in line 11, we employ $\mathbf{T}.align()$ to introduce inter-nodes. An example of $\mathbf{T}.align()$ is illustrated in Figure 8.

Once $\mathbf{T}$ returned, Algorithm 1 merges $\mathbf{T}$ into T at the original position of $\mathbf{v}$. Since $\mathbf{v}$ is selected from either v_r^T or V_T^1, both of which derive subtree(s) with height 1, merging $\mathbf{T}$ of height 2 will increase the height of T by 1. As depicted in Figure 9, Algorithm 1 aims to iteratively invoke Algorithm 2 and integrate the returned $\mathbf{T}$ until T reaches height K. The decision of whether Algorithm 2 is invoked on v_r^T or V_T^1 depends on which of the two selections results in less structural entropy for the merged coding tree T.

C Proof for Theorem 1

Proof. According to Li and Pan (2016), structural entropy decodes the essential structure of the original system while measuring the structural information to support the semantic modeling of the system. Thus, we have,

$$I(\mathcal{F}^*(\mathcal{G}_\mathcal{L} \circ \mathcal{D}); Y) = I(\mathcal{F}^*(\mathcal{T}_\mathcal{L} \circ \mathcal{D}); Y). \quad (19)$$

Considering the data processing inequality (Cover and Thomas, 2006) for data augmentation, we would have,

$$I((\mathcal{G}_\mathcal{L}, \mathcal{D}); Y) \geq I(\theta(\mathcal{G}_\mathcal{L}, \mathcal{D}); Y), \quad (20)$$

where θ is a general data augmentation function acting on $(\mathcal{G}_\mathcal{L} \circ \mathcal{D})$. Integrating the above equations, we have,

$$\begin{aligned} I((\mathcal{T}_\mathcal{L} \circ \mathcal{D}); Y) &\overset{(a)}{=} I(\mathcal{F}^*(\mathcal{T}_\mathcal{L} \circ \mathcal{D}); Y) && (21)\\ &\overset{(b)}{=} I(\mathcal{F}^*(\mathcal{G}_\mathcal{L} \circ \mathcal{D}); Y) && (22)\\ &\overset{(a)}{=} I((\mathcal{G}_\mathcal{L} \circ \mathcal{D}); Y) && (23)\\ &\overset{(c)}{\geq} I(\theta(\mathcal{G}_\mathcal{L} \circ \mathcal{D})); Y). && (24) \end{aligned}$$

where (a), (b), and (c) could be referred to Equation 12, Equation 19, and Equation 20, respectively. Here, we have concluded the proof that the information encoded by HILL is lossless, which is the upper bound of any other augmentation-based methods. □