Hierarchical Verbalizer for Few-Shot Hierarchical Text Classification

Ke Ji[1,2]*, **Yixin Lian**[2], **Jingsheng Gao**[2], **Baoyuan Wang**[2]†
[1] School of Computer Science and Engineering, Southeast University, China
[2] Xiaobing.AI
keji@seu.edu.cn
{lianyixin, gaojingsheng, wangbaoyuan}@xiaobing.ai

Abstract

Due to the complex label hierarchy and intensive labeling cost in practice, the hierarchical text classification (HTC) suffers a poor performance especially when low-resource or few-shot settings are considered. Recently, there is a growing trend of applying prompts on pre-trained language models (PLMs), which has exhibited effectiveness in the few-shot flat text classification tasks. However, limited work has studied the paradigm of prompt-based learning in the HTC problem when the training data is extremely scarce. In this work, we define a path-based few-shot setting and establish a strict path-based evaluation metric to further explore few-shot HTC tasks. To address the issue, we propose the hierarchical verbalizer ("HierVerb"), a multi-verbalizer framework treating HTC as a single- or multi-label classification problem at multiple layers and learning vectors as verbalizers constrained by hierarchical structure and hierarchical contrastive learning. In this manner, HierVerb fuses label hierarchy knowledge into verbalizers and remarkably outperforms those who inject hierarchy through graph encoders, maximizing the benefits of PLMs. Extensive experiments on three popular HTC datasets under the few-shot settings demonstrate that prompt with HierVerb significantly boosts the HTC performance, meanwhile indicating an elegant way to bridge the gap between the large pre-trained model and downstream hierarchical classification tasks. [1]

1 Introduction

Hierarchical text classification (HTC) is a long-standing research problem due to the wide range of real applications (Mao et al., 2019). However, prior works could still suffer poor performance in practice due to the nature of its sophisticated label hierarchy as well as the requirement of large-scale data annotation before training the model. Therefore, solving the HTC under the low-resource (Wang et al., 2022b) or few-shot setting becomes an urgent research topic.

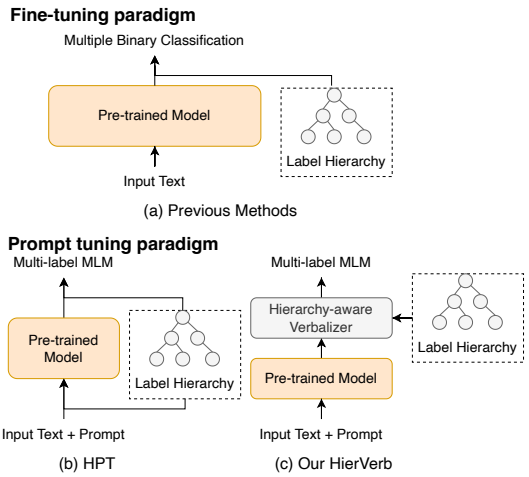

Figure 1: Illustration of methods for HTC problems. (a) Previous methods typically regard HTC as a downstream classification fine-tuning task. (b) HPT (Wang et al., 2022b) formulates HTC as a multi-label MLM problem following the prompt tuning paradigm. (c) Our HierVerb leverages hierarchy-aware verbalizers, which are more effective for few-shot tuning.

Existing state-of-the-art HTC models focus on inserting label hierarchy features through graph encoders and then fuse the features into the input layer (Wang et al., 2022b) or output layer (Zhou et al., 2020) of a text encoder such as Bidirectional LSTM or pre-trained language models (PLMs), as shown in Figure 1(a). And there is a trend of taking advantage of PLMs (Chen et al., 2021; Wang et al., 2022b) as the backbone of the text encoder through a fine-tuning paradigm. Despite the success of PLMs (Devlin et al., 2019; Raffel et al., 2020) in extensive NLP-related tasks, recently, a series of studies (Petroni et al., 2019; Davison et al., 2019; Chen et al., 2022) suggest that it's helpful to elicit

*Work done during an internship at Xiaobing.AI
†Corresponding Author
[1]Our code and few-shot dataset are publicly available at https://github.com/1KE-JI/HierVerb.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2918–2933
July 9-14, 2023

the knowledge contained in PLMs and point out the fine-tuning paradigm is suboptimal in few-shot settings due to distinct training strategies between the pre-training and fine-tuning stages. Inspired by "in-context learning" proposed by GPT-3 (Brown et al., 2020), lots of prompt-based (Petroni et al., 2019; Gao et al., 2021a; Schick and Schütze, 2021; Qin and Eisner, 2021) methods were proposed to bridge the gap between pre-training and downstream tasks via stimulating pre-trained model knowledge with a few hard or soft prompts. In prompt-based tuning, the input is usually wrapped through a natural language template and the tasks are converted as masked language modeling (MLM) for PLM. For example, in the sentiment classification task, the original input x will be wrapped as "x. It was `[MASK]`". The objective is to utilize MLM to predict the word that fills the `[MASK]`, and subsequently employ a *verbalizer* to map the predicted word to the final classification (e.g. "positive" -> label "Positive").

Although remarkable performances have been achieved via prompt tuning on the flat text classification where labels have no hierarchy, its effects on HTC problems remain unclear, as discussed in HPT (Wang et al., 2022b). As shown in Figure 1(b), HPT proposed a hierarchy-aware prompt tuning method that incorporates the label hierarchy knowledge into soft prompts through graph representation and achieves the new state-of-the-art results on several HTC popular datasets. However, even though the low-resource setting experiment was considered in HPT, the commonly used K-shot setting was not investigated. The limitation lies in the absence of a uniform definition of the K-shot setting in HTC. Besides, the way to utilize PLMs in few-shot settings through soft prompts and fuse hierarchy by graph encoder into the PLMs harms tapping the full potential of PLMs. Hence, it is crucial to exploit a new method to elicit knowledge from PLMs in a hierarchy-aware manner for few-shot learning.

Inspired by the prior works on verbalizer design (Gao et al., 2021a; Schick and Schütze, 2021) between model outputs and labels, as shown in Figure 4(a) and 4(b), which makes promising improvements over prompt-based tuning, it is natural to raise this question: is there any verbalizer design method specific to the HTC problems? The most current works can be mainly divided into three kinds of verbalizers: manual verbalizers, search-based verbalizers, and soft verbalizers. However, the main difference between previous works on verbalizers is the way of embedding the semantic space and they are all based on a strong assumption that there is no hierarchical dependency between downstream task labels, which raises a gap between rich flat prior knowledge in PLM and downstream task hierarchies. Thus these verbalizers are not suitable for hierarchical classification tasks, lacking awareness of hierarchy in their architectural design. To address these issues, we introduce a hierarchical-aware verbalizer (HierVerb) combined with the prompt tuning method to fully exploit the hierarchical knowledge within labels. The major contributions of this paper can be summarized as follows:

- To our best knowledge, we are the first to define the path-based few-shot setting on hierarchical text classification tasks and propose a path-based evaluation metric to further explore the consistency problem in HTC tasks.
- We propose HierVerb for few-shot HTC, which integrates the hierarchical information into the verbalizers through the flat hierarchical contrastive learning and hierarchy-aware constraint chain to better leverage the pre-trained language model for few-shot learning.
- Experimental results demonstrate that HierVerb significantly outperforms the current state-of-the-art HTC methods on three popular benchmarks (WOS, DBPedia, and RCV1-V2) under extreme few-shot settings (i.e., K <=8), validating the effectiveness of its design.

2 Related Work

2.1 Hierarchical Text Classification

Current works for HTC focus on finding ways to insert the hierarchical label knowledge into the model, which proves to be beneficial for the problem induced by the imbalanced and large-scale label hierarchy faced in HTC problems (Mao et al., 2019). Several works (Zhang et al., 2022; Wu et al., 2019; Mao et al., 2019) applied the label-based attention module or utilized the meta-learning and reinforcement learning methods to leverage the label structure. However, as pointed out in HiAGM (Zhou et al., 2020), such methods mainly concentrate on optimizing decoding results based on the

constraint of hierarchical paths, it proposed to encode the holistic label structure with hierarchy encoders (graph or tree structure) which demonstrate to improve performance to a greater extent. Following the line of this research, Chen et al. (2021) exploited the relationship between text and label semantics using matching learning, and Wang et al. (2021) explicitly enriched the label embedding with concepts shared among classes. Yet since the label hierarchy representation remains unchanged regardless of the input, later works like HGCLR (Wang et al., 2022a) and HPT (Wang et al., 2022b) chose to migrate label hierarchy into text encoding instead of separately modeling text and labels. In addition to this, HPT achieves state-of-the-art by exploiting pre-trained language models through prompt tuning methods. Although the methods above are designed for HTC problems and prompt-based techniques are applied, the frequently faced few-shot issues in HTC are less investigated, not to mention a suitable solution working well on limited training samples in a hierarchy-aware manner.

2.2 Prompt Tuning

Recent years have observed the widespread and powerful use of pre-trained language models (PLMs) in various downstream NLP tasks (Devlin et al., 2019; Qiu et al., 2020; Han et al., 2021). Prompt engineering goes a step further by designing a prompt template to take the power of PLMs to unprecedented heights, especially in few-shot settings (Liu et al., 2021). Later works focus on automatically discovering better hard prompts described in a discrete space to use in the querying process (Jiang et al., 2020; Gao et al., 2021a). Besides, there come with many methods that learn continuous soft prompts directly in the feature space of PLMs (Li and Liang, 2021; Lester et al., 2021; Qin and Eisner, 2021). Such continuous prompts reduce the hassle of constructing template words and transform them into parameterized embeddings.

2.3 Verbalizer Design

Verbalizers aim to reduce the gap between model outputs and label words, which has always been a critical issue in prompt-based tuning. Most of the current works leverage human written verbalizers (Schick and Schütze, 2021) that prove to be effective to build bridges between them. However, these approaches are highly biased towards lexical semantics of manual verbalizers and require both

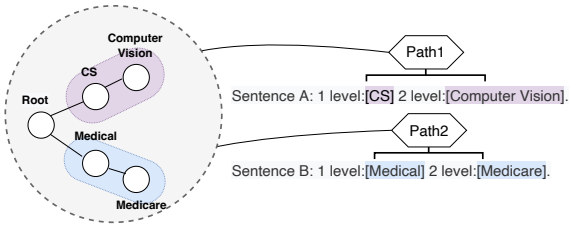

Figure 2: A few-shot HTC example. Path 1 contains labels (nodes) for *CS* and *Computer Vision* while Path 2 contains *Medical* and *Medicare*. Suppose we sample based on the hierarchical label paths in this figure, the sub-dataset consisting of sentences A and B is the support set of our 1-shot HTC.

domain expertise of downstream tasks and understanding of the PLMs' abilities (Schick et al., 2020). Schick et al. (2020) and other studies (Gao et al., 2021a; Shin et al., 2020) have designed search-based verbalizers for better verbalizer choices during the training optimization process, intending to reduce the bias caused by personal vocabulary and the cost of intensive human labor. Another line of researches (Hambardzumyan et al., 2021; Cui et al., 2022) claims it is hard to find satisfactory label words by searching large vocabulary with few examples and proposes to insert learnable embedding vectors as soft labels/verbalizers optimized during the training process. Nevertheless, the verbalizer design methods for hierarchical labels are less explored in previous works.

3 Preliminaries

3.1 Traditional HTC

In traditional HTC task, the structure of candidate labels $y_i \in Y$ are predefined as a Directed Acyclic Graph (DAG) $\mathcal{H} = (Y, E)$, where Y is the label set and E denotes the hierarchical connections within the labels. Specifically, $\mathcal{H}$ is a tree-like structure where every node except the root has one and only one parent. Hence the predicted hierarchical labels for one input sample correspond to single- or multi-path taxonomic labels in $\mathcal{H}$. It is worth noting that the HTC task is often viewed as a multi-label problem. Therefore the standard HTC task can be defined as follows: given an input text $\mathbf{x}=\{x_t\}_{t=1}^{T}$ and a label set Y, HTC aims to find a subset y from Y, in other words, to find one label path or multiple paths in $\mathcal{H}$, for $\mathbf{x}$.

3.2 Few-shot HTC

The few-shot problem has been extensively studied on tasks such as text classification, image segmentation, and named entity recognition (NER), while few works focus on the few-shot HTC task, which we call Few-HTC. It is easy to perform sampling strategies in flat single-label text classification to select K examples for each class added to the support set of K-shot learning. However, this sampling method is difficult to directly apply to HTC because an input sample may contain multiple labels. Hence it is harder to strictly meet the requirement of K shots for each corresponding class (Ding et al., 2021).

Inspired by the few-shot settings in named entity recognition (Yang and Katiyar, 2020; Ding et al., 2021) where they regard entity types as basic classes and sample few-shot sets based on each class through greedy sampling algorithms, we can define our few-shot settings based on the label paths in $\mathcal{H}$ since multiple slots in NER are analogous to multiple label paths in HTC. Figure 2 shows how we perform path-based sampling for building a Few-HTC support set.

Formally, the task of K-shot HTC is defined as follows: given a text $\mathbf{x}=\{x_t\}_{t=1}^{T}$ and a K-shot support set S for the target mandatory-leaf (Bi and Kwok, 2012) path set $C_{\mathcal{T}}$, the goal is to predict all golden paths on the label hierarchy tree for $\mathbf{x}$. We design a greedy sampling method specifically for HTC problems and the details of obtaining $C_{\mathcal{T}}$ and the support set S from the original HTC datasets are shown in Algorithm 1 to make sure each label path has exactly K-shot examples. To the best of our knowledge, we are the first to apply path-based few-shot settings on the HTC tasks.

Algorithm 1 Greedy sampling for Few-shot HTC

Input: shot K, original HTC dataset X{(x,y)} with label hierarchy $\mathcal{H}$
Output: K-shot support set S after sampling

$C_{\mathcal{T}} \leftarrow$ //Initialize the original set of mandatory − leaf paths
while $ori\_length \neq cur\_length$ **do**
  ori_length $\leftarrow$ //Obtain the length of $C_{\mathcal{T}}$
  Count the frequency of each C_i in X
  Remove paths $\{C_i\}$ with frequency less than K
  Remove samples containing $\{C_i\}$ in X
  cur_length $\leftarrow$ //Obtain the length of $C_{\mathcal{T}}$
end while
$\{C_i : A_i\} \leftarrow$ //Count the frequency of each C_i appeared individually in the filtered dataset X
Sort the path set $C_{\mathcal{T}}$ based on A
S $\leftarrow \phi$//Initialize an empty support set
$\{\text{Count}_i\} \leftarrow$ //Initialize the counts of all paths in $C_{\mathcal{T}}$ to zero
for $i = 1$ *to* $|C_{\mathcal{T}}|$ **do**
  while $\text{Count}_i < \text{K}$ **do**
    $\text{Sample(x, y)} \in \text{X}_{\text{s.t.}} C_i \in \text{y}$, w/o replacement
    $S \leftarrow S \cup \{(\text{x}, \text{y})\}$
    Update $\{\text{Count}_j\} \forall\ C_j \in \text{y}$
  end while
end for
return S

4 Hierarchical Verbalizer

In this section, we will introduce the proposed hierarchy-aware verbalizer in detail. We incorporate hierarchical information through our multi-verbalizer framework with prompt templates to elicit rich prior knowledge within the PLMs. Figure 3 shows the overall architecture of our proposed HierVerb. We first obtain the hidden states of the multiple mask tokens to represent the sentence and then project it to the verbalizer's space of different label layers.

4.1 Multi-verbalizer Framework

Since the label hierarchy is a tree structure in our problem, we think of HTC as a single-label or multi-label classification task performed at multiple levels, following Wang et al. (2022b). In this way, we can easily construct templates based on the depth of the hierarchy tree. Given a piece of training text x and the label hierarchy $\mathcal{H}$ with a depth of D, the template p is written simply as "`[CLS] It was 1 level:[MASK] 2 level:[MASK]...`*D* `level:[MASK].` *x* `[SEP]`". We use multiple `[MASK]` tokens for corresponding multi-level label predictions. Note that the number of `[MASK]` tokens is equal to the number of layers of $\mathcal{H}$.

For better learning of hierarchical verbalizer and text representation in few-shot settings, we use **BERT** (Devlin et al., 2019) as our text encoder. For an input text x wrapped with the template T:

$$T_{prompt}(x) = \{\text{[CLS] It was } t_i \text{ ... } t_D\text{. x [SEP]}\} \quad (1)$$

where t_i means "`i level:[MASK]`". Note that our template T is a dynamically wrapped sentence containing as many t as the number of hierarchy layers. We feed the input x wrapped with the template T to the encoder of the BERT to obtain the hidden states $h_{1:n}$:

$$h_{1:n} = \text{BERT}(T_{prompt}(x)_{1:n}) \quad (2)$$

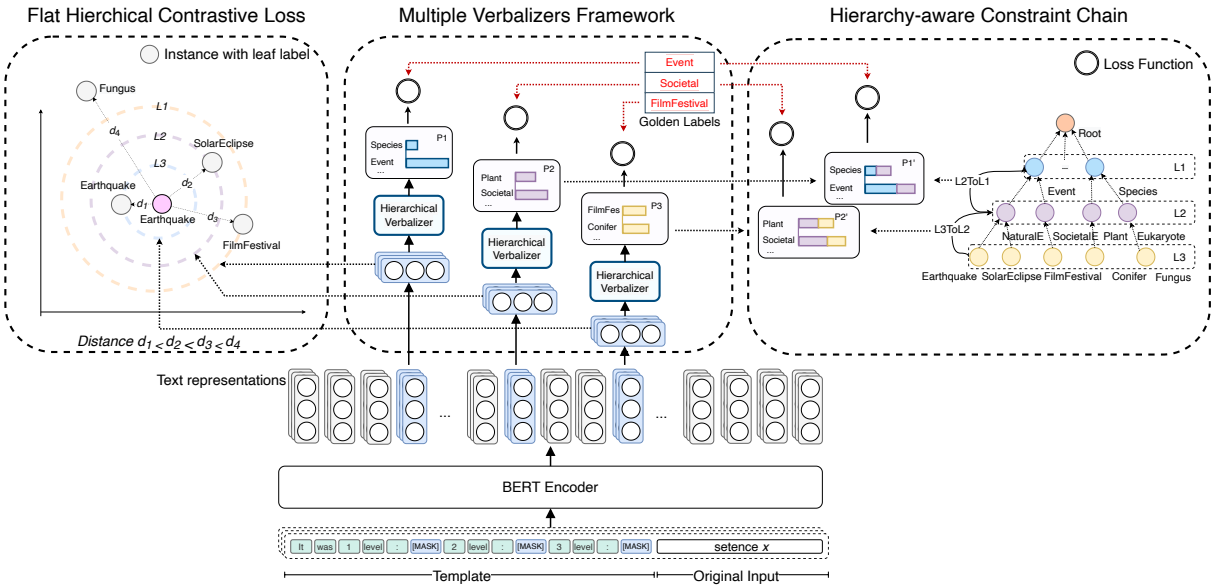

Figure 3: The architecture of HierVerb. There are two ways to insert hierarchical constraints into the model: (a) Hierarchy-aware Constraint Chain. (b) Flat Hierarchical Contrastive Loss. P1 to P3 denote the predicted probabilities of labels in each layer. P1′ and P2′ represent the propagated probabilities in each layer. L3 to L1 contain labels from a lower to a higher layer. d_1 to d_4 stand for the distances between different input instances. Worth noting that the instance with the leaf label "Solar Eclipse" positioned in the L2 circle, for example, shares the same gold labels in layer 2 with the instance marked pink in the center of the circle, and d_2 reflects the similarity distance between their output token embeddings of mask tokens corresponding to layer 2. The same applies to other instances.

where $h_{1:n} \in \mathbb{R}^{n \times r}$ and r is the hidden state dimension of BERT and n is the length of $T_{prompt}(x)$. For convenience, we pick out a subset $\{h^d\}(d \in [1, ..., D])$ which is the set of hidden state vectors corresponding to all `[MASK]` tokens.

On top of this, we use multi-verbalizer for depth-oriented learning and construct each verbalizer based on the full set of labels for the corresponding layer. Thus we have a list of verbalizers V = $\{V_d\}(d \in [1, ..., D])$. Each verbalizer is created as a virtual continuous vector $W_d \in \mathbb{R}^{r \times l_d}$ where l_d is the number of labels of d-th layer and we initialize the embedding W_d of each V_d by averaging the embeddings of its corresponding label tokens and label tokens of all its descendants in $\mathcal{H}$.

In our framework, the d-th mask is connected to the d-th verbalizer to play the role of predicting the d-th layer label.

We denote the distribution of the wrapped sentences in the corpus as $\mathcal{O}$. The probability distribution of all labels y_d on the layer d is:

$$P_{\mathcal{O}}(y_d|T_{prompt}(x), \mathcal{D} = d) = q(h^d W_d + b_d) \quad (3)$$

where $W_d \in \mathbb{R}^{r \times l_d}$ and $b_d \in \mathbb{R}^{l_d}$ are weights and bias and q is a function used to convert logits into probabilities. Hence the predicted probability of text i on label j of d-th layer is:

$$p_{ij}^d = P_{\mathcal{O}}(y_d = j|T_{prompt}(x), \mathcal{D} = d) \quad (4)$$

Following previous work (Zhou et al., 2020; Wang et al., 2022a), we use a binary cross-entropy loss function for multi-label classification. However, the definition of multi-label in our framework is slightly different from these works. The multi-label problem whose ground truth is a single path on the hierarchical dependency tree $\mathcal{H}$ can be redefined as a single-label prediction problem at each layer with the help of the multi-verbalizer. For such a single-path prediction, the loss function is defined as:

$$L_{idj}^C = -y_{ij}^d log(p_{ij}^d) \quad (5)$$

Instead, for multi-path problems:

$$L_{idj}^C = -y_{ij}^d log(p_{ij}^d) - (1 - y_{ij}^d) log(1 - p_{ij}^d) \quad (6)$$

To sum up, for each input text i, we can calculate the loss of the multi-verbalizer framework as:

$$\mathcal{L}_C = \sum_{d}^{D} \sum_{j}^{l_d} L_{idj}^C = \sum_{d}^{D} \sum_{j}^{l_d} L^C(p_{ij}^d, y_{ij}^d) \quad (7)$$

4.2 Hierarchy-aware Constraint Chain

In order to reduce the gap between the training objective of the pre-trained model and the hierarchical objective, we first use the hierarchical constraint chain to solve this problem.

According to the label dependency tree $\mathcal{H}$, we maintain a parent-to-child mapping $\overrightarrow{M}$ between layers:

$$\overrightarrow{M}_d(y_j^d) = \{y_1^{d+1}, y_2^{d+1}, ..., y_n^{d+1}\} \quad (8)$$

where y_d is a label *j* belonging to the *d*-th layer and $\{y_n^{d+1}\}$ are its corresponding children nodes at the (*d*+1)-th layer.

Thus the propagated probability of text *i* on label *j* of *d*-th layer can be obtained through:

$$\tilde{p}_{ij}^d = (1-\beta)p_{ij}^d + \beta \sum p_{i\tilde{j}}^{d+1}, \tilde{j} \in \overrightarrow{M}_d(j) \quad (9)$$

which is implemented to quantify constraints from descendant nodes where β controls the degree of descendant node constraints. Since we are propagating from the bottom up, our computational constraints gradually propagate upward from the leaf nodes of the hierarchy tree. The loss of the constraint chain can be defined as:

$$\mathcal{L}_{HCC} = \sum_{d}^{D} \sum_{j}^{l_{d-1}} L^C(\tilde{p}_{ij}^d, y_{ij}^d) \quad (10)$$

4.3 Flat Hierarchical Contrastive Loss

Secondly, we design the flat hierarchical contrastive loss objective to learn the hierarchy-aware matching relationship between instances, instead of the relationship between instances and labels as proposed in Chen et al. (2021). It is non-trivial to match different instances due to the sophisticated semantics of each instance in the hierarchical setting. Given input sentence representation and the label hierarchy, there are two main goals we want to achieve through optimization: (1) For sentence pairs, the representations of intra-class correspondences at each level should obtain higher similarity scores than inter-class pairs. (2) The similarity between lower-level representations of intra-class pairs deserves more weight than that of relatively high-level ones. To achieve our goal, we flatten the hierarchy into a multi-level lattice structure and define our objective function based on the SimCSE estimator (Gao et al., 2021b), which is widely used in contrastive learning.

Denote $\mathbf{B} = \{(X_n, \{Y^d\}_n)\}$ as one batch where $\{Y^d\}_n$ is the original labels in *d*-th layer, $n \in N$, $d \in D$, where *N* denotes the batch size and *D* denotes the maximum depth of the label hierarchy $\mathcal{H}$. Following SimCSE, we can have 2N sets of hidden vectors for all corresponding `[MASK]` tokens $\mathbf{Z} = \{z \in \{h^d\} \cup \{\tilde{h}^d\}\}$ where $\tilde{h}^d$ is simply obtained by feeding original text into the encoder for the second time. Any sentence pairs in one batch can be defined as $\mathcal{P} = [(X_a, \{Y^d\}_a), (X_b, \{Y^d\}_b)]$, and we keep a lattice label matrix:

$$M_d(a,b) = \begin{cases} 1, & \{Y^d\}_a \cap \{Y^d\}_b \neq \phi \\ 0, & \{Y^d\}_a \cap \{Y^d\}_b = \phi \end{cases} \quad (11)$$

Thus the final flat hierarchical contrastive loss function is defined as:

$$L_{FHC} = \frac{-1}{N^2 D^2} \sum_{d}^{D} \sum_{u}^{d} \sum_{n}^{2N} \log \frac{\exp(\sum_{n'} S(h_n^u, h_{n'}^u) M_u(n,n'))}{\exp(\sum_{n'} S(h_n^u, h_{n'}^u))} \times \frac{1}{2^{(D-d)\times\alpha}} \quad (12)$$

where *S* is cosine similarity function, h_n^d is the hidden states of *d*-th `[MASK]` for sentence *n*, and α controls the relative penalty importance of different layers.

Considering that once $M_d(n,n')$ equals to one, all $M_u(n,n')$ can be assured to be equal to one by reason of the tree structure. Thereafter it assigns more weight to the contrastive loss of the lower layer whose *d* value is larger, and α intensifies the differentiation between all layers. This results in the inequality $Distance\ d_1 < d_2 < d_3 < d_4$ in Figure 3.

4.4 Classification Objective Function

Overall, our final training objective is the combination of multi-verbalizer framework loss, constraint chain loss, and flat hierarchical contrastive loss.

$$\mathcal{L} = \mathcal{L}_C + \lambda_1 \mathcal{L}_{HCC} + \lambda_2 \mathcal{L}_{FHC} \quad (13)$$

where $\lambda 1$ and $\lambda 2$ are the hyperparameters controlling the weights of corresponding loss and HCC and FHC stand for Hierarchy-aware Constraint Chain and Flat Hierarchical Contrastive Loss respectively.

	DBPedia	WOS	RCV1-V2
Level 1 Categories	9	7	4
Level 2 Categories	70	134	55
Level 3 Categories	219	NA	43
Level 4 Categories	NA	NA	1
Number of documents	381025	46985	804410
Mean document length	106.9	200.7	221.29

Table 1: Comparison of popular HTC datasets.

K	Method	WOS(Depth 2)		DBpedia(Depth 3)		RCV1-V2(Depth 4)	
		Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
1	BERT (Vanilla FT)	2.99 ± 20.85 (5.12)	0.16 ± 0.10 (0.24)	14.43 ± 13.34 (24.27)	0.29 ± 0.01 (0.32)	7.32 ± 10.33 (9.32)	3.73 ± 0.10 (3.73)
	HiMatch (Chen et al., 2021)	43.44 ± 8.90 (48.26)	7.71 ± 4.90 (9.32)	-	-	-	-
	HGCLR(Wang et al., 2022a)	9.77 ± 11.77 (16.32)	0.59 ± 0.10 (0.63)	15.73 ± 31.07 (25.13)	0.28 ± 0.10 (0.31)	26.46 ± 1.27 (26.80)	1.34 ± 0.93 (1.71)
	HPT (Wang et al., 2022b)	50.05 ± 6.80 (50.96)	25.69 ± 3.31 (27.76)	72.52 ± 10.20 (73.47)	31.01 ± 2.61 (32.50)	27.70 ± 5.32 (28.51)	3.35 ± 2.22 (3.90)
	HierVerb	**58.95 ± 6.38 (61.76)**	**44.96 ± 4.86 (48.19)**	**91.81 ± 0.07 (91.95)**	**85.32 ± 0.04 (85.44)**	**40.95 ± 3.12 (41.22)**	**4.87 ± 1.71 (5.71)**
2	BERT (Vanilla FT)	46.31 ± 0.65 (46.85)	5.11 ± 1.31 (5.51)	87.02 ± 3.89 (88.20)	69.05 ± 26.81 (73.28)	8.07 ± 2.18 (9.13)	2.76 ± 6.01 (4.11)
	HiMatch (Chen et al., 2021)	46.41 ± 1.31 (47.23)	18.97 ± 0.65 (21.06)	-	-	-	-
	HGCLR (Wang et al., 2022a)	45.11 ± 5.02 (47.56)	5.80 ± 11.63 (9.63)	87.79 ± 0.40 (88.42)	71.46 ± 0.17 (71.78)	34.33 ± 4.81 (37.28)	2.51 ± 6.12 (6.12)
	HPT (Wang et al., 2022b)	57.45 ± 1.89 (58.99)	35.97 ± 11.89 (39.94)	90.32 ± 0.64 (91.11)	81.12 ± 1.33 (82.42)	38.93 ± 3.55 (40.47)	8.31 ± 5.26 (10.52)
	HierVerb	**66.08 ± 4.19 (68.01)**	**54.04 ± 3.24 (56.69)**	**93.71 ± 0.01 (93.87)**	**88.96 ± 0.02 (89.02)**	**48.00 ± 2.27 (49.21)**	**11.74 ± 1.58 (12.69)**
4	BERT (Vanilla FT)	56.00 ± 4.25 (57.18)	31.04 ± 16.65 (33.77)	92.94 ± 0.66 (93.38)	84.63 ± 0.17 (85.47)	17.94 ± 0.01 (18.00)	1.45 ± 0.01 (1.57)
	HiMatch (Chen et al., 2021)	57.43 ± 0.01 (57.43)	39.04 ± 0.01 (39.04)	-	-	-	-
	HGCLR (Wang et al., 2022a)	56.80 ± 4.24 (57.96)	32.34 ± 15.39 (33.76)	93.14 ± 0.01 (93.22)	84.74 ± 0.11 (85.11)	45.53 ± 4.20 (47.71)	8.56 ± 1.63 (9.92)
	HPT (Wang et al., 2022b)	65.57 ± 1.69 (67.06)	45.89 ± 9.78 (49.42)	94.34 ± 0.28 (94.83)	90.09 ± 0.87 (91.12)	52.62 ± 0.20 (52.73)	20.01 ± 0.31 (20.21)
	HierVerb	**72.58 ± 0.83 (73.64)**	**63.12 ± 1.48 (64.47)**	**94.75 ± 0.13 (95.13)**	**90.77 ± 0.33 (91.43)**	**56.86 ± 0.44 (57.11)**	**22.07 ± 0.32 (22.42)**
8	BERT (Vanilla FT)	66.24 ± 1.96 (67.53)	50.21 ± 5.05 (52.60)	94.39 ± 0.06 (94.57)	87.63 ± 0.28 (87.78)	57.27 ± 0.04 (57.51)	23.93 ± 0.45 (24.46)
	HiMatch (Chen et al., 2021)	69.92 ± 0.01 (70.23)	57.47 ± 0.01 (57.78)	-	-	-	-
	HGCLR (Wang et al., 2022a)	68.34 ± 0.96 (69.22)	54.41 ± 2.97 (55.99)	94.70 ± 0.05 (94.94)	88.04 ± 0.25 (88.61)	58.90 ± 1.61 (60.30)	27.03 ± 0.20 (27.41)
	HPT (Wang et al., 2022b)	76.22 ± 0.99 (77.23)	67.20 ± 1.89 (68.63)	95.49 ± 0.01 (95.57)	92.35 ± 0.03 (92.52)	59.92 ± 4.25 (61.47)	29.03 ± 6.23 (32.19)
	HierVerb	**78.12 ± 0.55 (78.87)**	**69.98 ± 0.91 (71.04)**	**95.69 ± 0.01 (95.70)**	**92.44 ± 0.01 (92.51)**	**63.90 ± 2.42 (64.96)**	**31.13 ± 1.63 (32.52)**
16	BERT (Vanilla FT)	75.52 ± 0.32 (76.07)	65.85 ± 1.28 (66.96)	95.31 ± 0.01 (95.37)	89.16 ± 0.07 (89.35)	63.68 ± 0.01 (64.10)	34.00 ± 0.67 (34.41)
	HiMatch (Chen et al., 2021)	77.67 ± 0.01 (78.24)	68.70 ± 0.01 (69.58)	-	-	-	-
	HGCLR (Wang et al., 2022a)	76.93 ± 0.52 (77.46)	67.92 ± 1.21 (68.66)	95.49 ± 0.04 (95.63)	89.41 ± 0.09 (89.71)	63.91 ± 1.42 (64.81)	33.25 ± 0.10 (33.50)
	HPT (Wang et al., 2022b)	79.85 ± 0.41 (80.58)	72.02 ± 1.40 (73.31)	96.13 ± 0.01 (96.21)	**93.34 ± 0.02 (93.45)**	**65.73 ± 0.80 (66.24)**	**36.34 ± 0.20 (36.57)**
	HierVerb	**80.93 ± 0.10 (81.26)**	**73.80 ± 0.12 (74.19)**	**96.17 ± 0.01 (96.21)**	93.28 ± 0.06 (93.49)	65.50 ± 1.41 (66.62)	35.10 ± 1.73 (36.24)

Table 2: F1 scores on 3 datasets. We report the mean F1 scores (%) over 3 random seeds. **Bold**: best results. **Underlined**: second highest.

5 Experiments

5.1 Experiments Setup

Experimental settings As mentioned in Preliminaries, we focus on few-shot settings that only K samples for each label path are available for training on a new HTC task called Few-HTC in this work. In order to better study the few-shot generalization ability of the model under different scales of training data, we conduct experiments based on K ∈ {1,2,4,8,16}.

Datasets and Implementation Details We evaluate our proposed method on three widely used datasets for hierarchical text classification: Web-of-Science (WOS) (Kowsari et al., 2017), DBpedia (Sinha et al., 2018) and RCV1-V2 (Lewis et al., 2004). WOS and DBPedia are for single-path HTC while RCV1-V2 includes multi-path taxonomic labels. The statistic details are illustrated in Table 1. For implementation details, please refer to Appendix A.

Evaluation Metrics Similar to previous work, we measure the experimental results with Macro-F1 and Micro-F1. To further evaluate the consistency problem between layers, we adopt path-constrained MicroF1 (C-MicroF1) and path-constrained MacroF1 (C-MacroF1) proposed in Yu et al. (2022) which we refer to collectively as C-metric. In C-metric, a correct prediction for a label node is valid only if all its ancestor nodes are correct predictions, otherwise, it is regarded as a misprediction. However, in the case of path splitting based on the mandatory-leaf nodes, the metric is still not sufficient to provide a comprehensive evaluation of hierarchical path consistency, because it ignores the correctness of a node's children nodes. Therefore, we propose a new path-constrained evaluation method based on the perspective of path correctness, which is called P-metric (PMacro-F1 and PMicro-F1). The details of our P-metric are shown in Appendix B.

Baselines We select a few recent state-of-the-art works as baselines: HiMatch (Using BERT as encoder) (Chen et al., 2021), HGCLR (Wang et al., 2022a) and HPT (Wang et al., 2022b). We also perform the vanilla fine-tuning method on the Few-shot HTC task, which we refer to as Vanilla FT in the following.

5.2 Main Results

Main experimental results are shown in Table 2. As is shown, HierVerb wins over all comparison models by a dramatic margin under nearly all sit-

K	Method	WOS			
		PMicro-F1	PMacro-F1	CMicro-F1	CMacro-F1
1	Ours	**39.77**	**37.24**	**55.18**	**39.42**
	HPT	19.97	17.47	49.10	22.92
	HGCLR	0.0	0.0	2.21	0.09
	Vanilla FT	0.0	0.0	0.96	0.04
2	Ours	**50.15**	**47.98**	**62.90**	**49.67**
	HPT	28.27	26.51	56.64	33.50
	HGCLR	1.39	1.49	45.01	4.88
	Vanilla FT	1.43	1.42	45.75	4.95
4	Ours	**62.16**	**59.70**	**72.41**	**61.19**
	HPT	50.96	48.76	69.43	55.27
	HGCLR	29.94	27.70	57.43	34.03
	Vanilla FT	22.97	20.73	55.10	27.50

Table 3: Consistency experiments on the WOS dataset using two path-constraint metrics. PMicro-F1 and PMacro-F1 are our proposed path-based consistency evaluation P-metric. We report the mean F1 scores (%) over 3 random seeds. For display, here we call BERT (Vanilla FT) as Vanilla FT. **Bold**: best results.

K	Ablation Models	WOS	
		Micro-F1	Macro-F1
1	Ours	**58.95**	**44.96**
	$r.m.$ FHC loss	58.13	44.63
	$r.m.$ HCC loss	58.26	44.27
	$+r.m.$ HCC+FHC loss	58.35	44.48
	$+r.m.$ multi-verb (Vanilla SoftVerb)	56.11	41.35
2	Ours	**66.08**	**54.04**
	$r.m.$ FHC loss	65.40	53.89
	$r.m.$ HCC loss	65.87	53.94
	$+r.m.$ HCC+FHC loss	65.23	53.47
	$+r.m.$ multi-verb (Vanilla SoftVerb)	62.31	49.33
4	Ours	**72.58**	**63.12**
	$r.m.$ FHC loss	72.51	62.70
	$r.m.$ HCC loss	72.05	62.52
	$+r.m.$ HCC+FHC loss	72.22	62.22
	$+r.m.$ multi-verb (Vanilla SoftVerb)	69.58	58.83
8	Ours	**78.12**	69.98
	$r.m.$ FHC loss	77.81	**70.28**
	$r.m.$ HCC loss	77.95	69.80
	$+r.m.$ HCC+FHC loss	77.88	69.85
	$+r.m.$ multi-verb (Vanilla SoftVerb)	75.99	66.99
16	Ours	**80.93**	**73.80**
	$r.m.$ FHC loss	80.76	73.54
	$r.m.$ HCC loss	80.73	73.69
	$+r.m.$ HCC+FHC loss	80.92	73.61
	$+r.m.$ multi-verb (Vanilla SoftVerb)	79.62	70.95

Table 4: Ablation experiments on WOS. $r.m.$ stands for $remove$ and $+r.m.$ stands for $remove$ on the basis of the previous step. We report the mean F1 scores (%) over 3 random seeds. **Bold**: best results.

uations. Appendix C more intuitively shows the performance gap between different models.

In the case of no more than 4 shots on WOS, 8.9%, 9.18%, and 6.91% micro-F1 absolute improvement and 19.27%, 18.3%, and 16.87% macro-F1 absolute improvement from the best baseline methods are achieved, respectively. Under 1-shot situations, compared with all baseline models, there is an average of 57.58% micro, 74.79% macro-F1 absolute improvement on DBPedia, and 20.46% micro-F1, 2.06% macro-F1 absolute improvement on RCV1-V2. Although the RCV1-V2 dataset provides no label name which has a negative effect on our verbalizer initialization, our method still achieves state-of-the-art on both Micro-F1 and Macro-F1 under almost all few-shot experiments.

There are three main reasons why HierVerb performs better under the few-shot setting: (1) Not require additional learning parameters. Previous methods like HPT and HGCLR improve the performance by adding extra parameters to the GNN layers, which could lead to overfitting for few-shot settings; (2) Multi-Verb is better than the single-flat verb. The previous methods are to first stretch the hierarchical label into a flattened one-dimensional space and then do multi-label prediction, more like a normal multi-label classification task with hierarchical dependencies on labels. In contrast, HierVerb advocates preserving the original hierarchical concept in the architecture through a multi-verb framework. (3) Our hierarchical loss is optimized from a semantic perspective for better generalization.

5.3 Consistency Between Multi-layers

Table 3 further studies the consistency performance. Since our method is optimized from a semantic perspective, more consideration is given to the potential semantic dependency between different labels rather than directly fitting specific downstream data, our method still maintains excellent consistency performance in the absence of sufficient labeled training corpora. It is clear that HGCLR and BERT (Vanilla FT) using the direct fitting method only achieve 0 points in PMicro-F1 and PMacro-F1 under the 1 shot setting. As for HPT, extra graph parameter learning hurts the generalization of PLMs. The complete experiments and analyses on the other two datasets are shown in Appendix D.

5.4 Ablation Study

The main parts of our work are the multi-verbalizer framework, hierarchy-aware constraint chain, and

flat hierarchical contrastive loss.

To illustrate the effect of these parts, we test our model by gradually removing each component of our model at a time by default, as shown in Table 4. We implement Vanilla Soft Verbalizer (Hambardzumyan et al., 2021) in our own version which we refer to as SoftVerb in the following for convenience. Similar to HierVerb, the SoftVerb also uses multiple `[MASK]` tokens, but only uses a single flat verbalizer to map the label. Compared to SoftVerb which uses a single flat verbalizer, using multi-verbalizer and integrating hierarchical information into the verbalizer of each layer through FHC and HCC leads to better performance.

5.5 Effects of Model Scales

In previous experiments like § 5.2, we show that HierVerb is powerful on bert-base-uncsaed. To further study the ability of HierVerb to utilize the prior knowledge of the pre-trained language model, we conduct experiments on bert-large-uncased. Table 5 demonstrates that HierVerb consistently outperforms all baseline models in all shot settings. We find that the gap is even significantly larger for HierVerb and all other baseline models compared to using bert-base-uncased. For example, under 1-shot setting, HierVerb achieves a 27.92% increase in macro-F1 and an 11.54% increase in micro-F1, compared with HPT. But in the case of bert-base-uncased, the improvements of macro-F1 and micro-F1 are 19.27% and 8.9% respectively, which further emphasizes that our model is superior to all baseline models in the ability to mine the prior knowledge of the language model, and this effect is more significant when the scale of the language model increases.

5.6 Performance Benefit in a Full-shot Setup

We conduct experiments on HierVerb in a full-shot setting. Instead of carefully selecting hyperparameters, we directly use the parameter set from the few-shot settings. For baseline models, we reproduce their experiments according to the settings in their original paper. Although HierVerb is designed to be more favored for few-shot settings, the performance of full-shot setup is still quite competitive compared with HPT. As shown in Table 6, our overall micro-F1 score is only 0.10 lower than HPT (which requires to learn extra parameters of GNN), while achieving a macro-F1 score 0.13% higher than HPT. In fact, HierVerb outperforms BERT (Vanilla FT) and HiMatch by a significant margin.

K	Method	WOS	
		Micro-F1	Macro-F1
1	HierVerb	**61.29**	**47.70**
	HPT	49.75	19.78
	HGCLR	20.10	0.50
	BERT (Vanilla FT)	10.78	0.25
2	HierVerb	**67.92**	**56.92**
	HPT	60.09	35.44
	HGCLR	44.92	3.23
	BERT (Vanilla FT)	20.50	0.34
4	HierVerb	**73.88**	**64.80**
	HPT	69.47	53.22
	HGCLR	68.12	52.92
	BERT (Vanilla FT)	67.44	51.66
8	HierVerb	**78.56**	**71.01**
	HPT	77.96	68.26
	HGCLR	71.48	56.91
	BERT (Vanilla FT)	73.98	62.82
16	HierVerb	**82.09**	**75.01**
	HPT	80.69	72.51
	HGCLR	78.01	67.87
	BERT (Vanilla FT)	78.52	69.64

Table 5: Using the same hyperparameter settings mentioned above, we conduct experiments on WOS with the bert-large-uncased (330M) encoder. **Bold**: best results.

Methods	WOS	
	Micro-F1	Macro-F1
HierVerb	87.00	**81.57**
HPT	**87.10**	81.44
HGCLR	87.08	81.11
HiMatch	86.70	81.06
BERT (Vanilla FT)	85.63	79.07

Table 6: Full-shot experiments on WOS using bert-base-uncased. **Bold**: best results.

6 Conclusion

In this paper, we define the few-shot settings on HTC tasks and a novel evaluation method based on the perspective of path correctness, which is valuable in practical applications. We propose a novel approach to adapt flat prior knowledge in PLM to downstream hierarchical tasks. The proposed HierVerb learns hierarchical-aware verbalizers through flat contrastive learning and constraint chain, which elegantly leverages the prior knowledge of PLMs for better few-shot learning. We perform few-shot settings on HTC tasks and extensive experiments show that our method achieves state-of-the-art performances on 3 popular HTC datasets while guaranteeing excellent consistency performance.

Limitations

Since the appearance of large pre-trained models such as GPT-3 (Brown et al., 2020), there has been a wave of using large models without fine-tuning to do in-context learning directly to complete various NLP tasks, or to freeze the parameters of large models and then only optimize task-oriented parameters. The proposed HierVerb is a lightweight method especially suitable for the case of insufficient labeled training data, but it is difficult to directly extend to a large-scale language model (i.e, >=175B) because large language models are hard to fine-tune in many situations. In future work, we plan to study our method on a larger scale language model in which only parts of parameters specific to downstream HTC tasks need to be learned and further, extend our model to the zero-shot learning scenario.

Ethics Statement

All datasets for our research are publicly available and all experimental results are based on three different random seeds. We obtain these experimental results using the experimental setup mentioned in this work. For the sake of energy saving, we will not only open source the few-shot datasets under all random seeds and the code, but also release the checkpoints of our models from the experiments to reduce unnecessary carbon emissions.

References

Wei Bi and James Kwok. 2012. Mandatory leaf node prediction in hierarchical multilabel classification. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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 (Volume 1: Long Papers)*, pages 4370–4379, Online. Association for Computational Linguistics.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*, pages 2778–2788.

Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. 2022. Prototypical verbalizer for prompt-based few-shot tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7014–7024, Dublin, Ireland. Association for Computational Linguistics.

Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pretrained models. 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)*, pages 1173–1178, Hong Kong, China. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun. 2022. OpenPrompt: An open-source framework for prompt-learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 105–113, Dublin, Ireland. Association for Computational Linguistics.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-NERD: A few-shot named entity recognition dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021a. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. 2021. Pre-trained models: Past, present and future. *AI Open*, 2:225–250.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kamran Kowsari, Donald E Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S Gerber, and Laura E Barnes. 2017. Hdltex: Hierarchical deep learning for text classification. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 364–371. IEEE.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

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)*, pages 445–455, Hong Kong, China. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5569–5578, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the*

2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4222–4235, Online. Association for Computational Linguistics.

Koustuv Sinha, Yue Dong, Jackie Chi Kit Cheung, and Derek Ruths. 2018. A hierarchical neural attention-based text classifier. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 817–823, Brussels, Belgium. Association for Computational Linguistics.

Xuepeng Wang, Li Zhao, Bing Liu, Tao Chen, Feng Zhang, and Di Wang. 2021. Concept-based label embedding via dynamic routing 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 (Volume 1: Long Papers)*, pages 5010–5019, Online. Association for Computational Linguistics.

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)*, pages 7109–7119, Dublin, Ireland. Association for Computational Linguistics.

Zihan Wang, Peiyi Wang, Tianyu Liu, Yunbo Cao, Zhifang Sui, and Houfeng Wang. 2022b. Hpt: Hierarchy-aware prompt tuning for hierarchical text classification. *arXiv preprint arXiv:2204.13413*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. 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)*, pages 4354–4364, Hong Kong, China. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375, Online. Association for Computational Linguistics.

Chao Yu, Yi Shen, and Yue Mao. 2022. Constrained sequence-to-tree generation for hierarchical text classification. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1865–1869.

Xinyi Zhang, Jiahao Xu, Charlie Soh, and Lihui Chen. 2022. La-hcn: label-based attention for hierarchical multi-label text classification neural network. *Expert Systems with Applications*, 187:115922.

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*, pages 1106–1117, Online. Association for Computational Linguistics.

A Implementation Details

All our models are implemented with PyTorch (Paszke et al., 2019) framework, Huggingface transformers (Wolf et al., 2020), and OpenPrompt toolkit (Ding et al., 2022). Following previous work (Wang et al., 2022b), we use `bert-base-uncased` from Transformers as our base architecture. The hidden size r is 768, and the number of layers and heads are 12. The batch size is 5. For WOS and DBPedia, the learning rate is $5e^{-5}$, besides we use a learning rate of $1e^{-4}$ to fasten the convergence of its hierarchical label words' embeddings and train the model for 20 epochs and apply the Adam Optimizer (Kingma and Ba, 2014) with a linearly decaying schedule with warmup steps at 0 and evaluate on the development set after every epoch. Since the labels of RCV1 do not contain excessively rich natural text semantics, the training iteration on RCV1 is the same as HPT (Wang et al., 2022b) with 1000 epochs and we set early stopping to 10 and learning rate to $3e^{-5}$ which is also used for the optimization of verbalizers. For baseline models, we keep the hyperparameter settings from their original papers except for setting early stopping to 10 for a fair comparison. We list the details of the other hyperparameters in Table 7.

B Path-based Evaluation Metric

Specifically, in P-metric, we evaluate the confusion matrix of all label path ids instead of the original label ids. Besides, only if all $\{y_i\}$ labels involved in one path are predicted accurately, the corresponding path id is regarded as correct in the confusion matrix. We count the total number of golden labels as $Count_{gold}$ and at the same time record the predicted labels that do not form a complete path with other predicted labels as invalid and count their total as $Count_{invalid}$.

We define:

$$\gamma = 1 - 2 \times (\frac{1}{(1+e^{-a})} - 0.5) \qquad (14)$$

where a = $\frac{Count_{invalid}}{Count_{gold}}$ and multiply γ with PMacro-F1 and PMicro-F1 obtained from the confusion matrix to get our final PMacro-F1 and PMicro-F1 so that we can penalize the evaluation score to get a fairer evaluation when the model smartly predicts a particularly large number of labels that do not form a complete path, considering that we are building confusion matrix based on the path. Figure 5 shows the inconsistency problem.

Hyper-parameter	Dataset	Value
truncate length	All	512
warmup steps	All	0
$\lambda 1$	All	1
$\lambda 2$	WOS&DBPedia	1e-2
$\lambda 2$	RCV1-V2	1e-4
α	All	1
β	WOS&DBPedia	1
β	RCV1-V2	1e-2

Table 7: Hyper-parameter settings

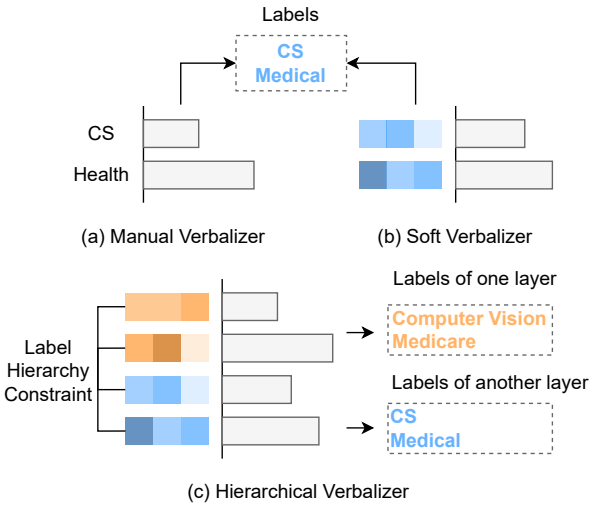

Figure 4: Comparison between previous verbalizer design methods and our HierVerb: (a) Manual Verbalizer. (b) Soft Verbalizer utilizes learnable vectors for each label word. (c) Our HierVerb fuzes hierarchical constraints into the soft verbalizer, effective for hierarchical problems.

C Performance Gap between Different Models

The performance gap on three datasets between different models is clearly shown in Figure 6-8. The gap keeps growing as the shots become fewer. It can be clearly seen that both HierVerb's Micro-F1 and Macro-F1 change very slightly from 1 to 16 shots on DBPedia while other models are particularly dependent on the increase of labeled training samples.

D Complete Consistency Experiments

We further conduct consistency experiments on two other datasets. The results are shown in Ta-

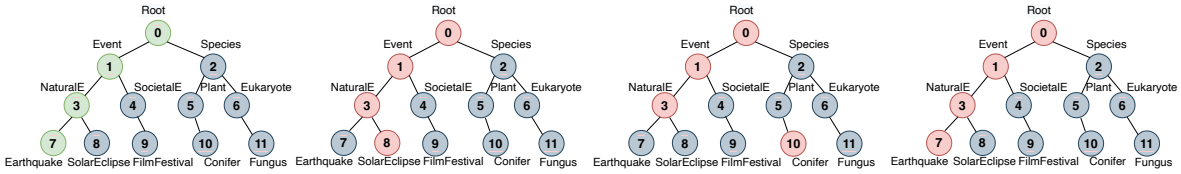

Figure 5: An example of taxonomic hierarchy for HTC. Assume we have an input sentence to predict, then we have (a) Golden labels of the input sentence, (b) and (c) Predicted labels lead to "label path inconsistency", and (d) Correctly predict all the labels in the path consisting of the node set {1,3,7}. We pick out these predicted labels in (c) that cannot form a complete path with any other predicted labels and record them as $Y_{invalid}$={1,3,10}, then we calculate the size of the set $Y_{invalid}$ and add it to $Count_{invalid}$.

K	Method	DBPedia				RCV1-V2			
		PMicro-F1	PMacro-F1	CMicro-F1	CMacro-F1	PMicro-F1	PMacro-F1	CMicro-F1	CMacro-F1
1	Ours	**83.56**	**77.96**	**89.80**	**81.78**	-	-	**39.41**	**5.16**
	HPT	61.08	57.80	82.84	66.99	-	-	21.92	2.87
	HGCLR	0.0	0.0	28.05	0.24	-	-	23.26	1.04
	Vanilla FT	0.0	0.0	28.08	0.24	-	-	19.37	1.02
2	Ours	**88.58**	**86.35**	**93.61**	**88.96**	-	-	**45.11**	**12.32**
	HPT	82.36	81.41	92.31	86.43	-	-	38.24	7.00
	HGCLR	54.55	3.72	67.70	26.41	-	-	24.24	0.89
	Vanilla FT	53.83	3.71	67.72	26.89	-	-	23.60	0.81
4	Ours	**91.90**	**91.38**	**95.74**	**92.87**	-	-	**54.67**	**23.80**
	HPT	87.61	87.04	94.50	90.42	-	-	50.68	20.54
	HGCLR	55.34	3.76	67.54	28.60	-	-	44.74	9.02
	Vanilla FT	55.15	3.74	67.44	28.32	-	-	22.42	0.63

Table 8: Consistency experiments on the DBPedia and RCV1-V2 datasets using two path-constraint metrics. PMicro-F1 and PMacro-F1 are our proposed path-based consistency evaluation P-metric. Since the label distribution of the original test set of RCV1-V2 is not mandatory-leaf in $\mathcal{H}$ while WOS and DBPedia are, we use only C-metric on RCV1-V2 to evaluate the consistency performance. We report the mean F1 scores (%) over 3 random seeds. **Bold**: best results. All experiments use their respective metrics as a signal for early stopping.

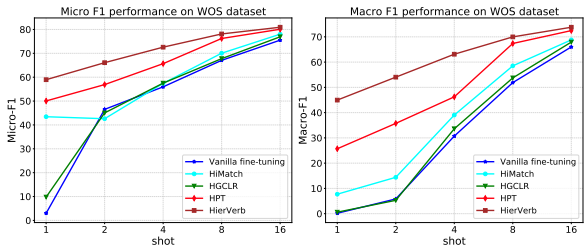

Figure 6: Performance gap on WOS dataset

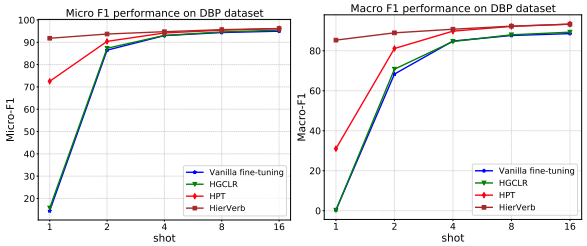

Figure 7: Performance gap on DBPedia dataset

ble 8. In all experiments, HGCLR and Vanilla FT consistently perform poorly on both P-Metric and C-Metric, while HierVerb and HPT achieved relatively high results, indicating that the prompt-based method can better use the prior knowledge in the pre-trained model to elicit potential semantic associations between natural language texts of all labels belonging to the same path.

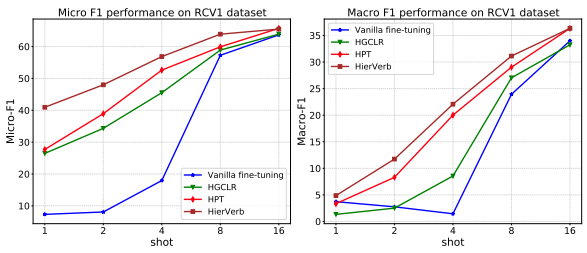

Figure 8: Performance gap on RCV1-V2 dataset

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
section Limitations

☒ A2. Did you discuss any potential risks of your work?
Our work is only for academic research purposes.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
section Abstract and section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
section 5

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
section 5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
section 5

☒ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
We haven't used the existing packages for evaluation. We use the code written by ourselves. The code we use will publish the code upon acceptance.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.