Natural Language Processing (2024), 1–18
doi:10.1017/nlp.2024.51

ARTICLE

Prompt tuning discriminative language models for hierarchical text classification

Jaco du Toit[1,2] and Marcel Dunaiski[1,2]

[1]Department of Mathematical Science, Computer Science Division, Stellenbosch University, Stellenbosch, South Africa and [2]School for Data Science and Computational Thinking, Stellenbosch University, Stellenbosch, South Africa
Corresponding author: Jaco du Toit; Email: jacowdutoit11@gmail.com

(Received 22 October 2023; revised 7 July 2024; accepted 12 July 2024)

Abstract
Hierarchical text classification (HTC) is a natural language processing task which aims to categorise a text document into a set of classes from a hierarchical class structure. Recent approaches to solve HTC tasks focus on leveraging pre-trained language models (PLMs) and the hierarchical class structure by allowing these components to interact in various ways. Specifically, the Hierarchy-aware Prompt Tuning (HPT) method has proven to be effective in applying the prompt tuning paradigm to Bidirectional Encoder Representations from Transformers (BERT) models for HTC tasks. Prompt tuning aims to reduce the gap between the pre-training and fine-tuning phases by transforming the downstream task into the pre-training task of the PLM. Discriminative PLMs, which use a replaced token detection (RTD) pre-training task, have also shown to perform better on flat text classification tasks when using prompt tuning instead of vanilla fine-tuning. In this paper, we propose the Hierarchy-aware Prompt Tuning for Discriminative PLMs (HPTD) approach which injects the HTC task into the RTD task used to pre-train discriminative PLMs. Furthermore, we make several improvements to the prompt tuning approach of discriminative PLMs that enable HTC tasks to scale to much larger hierarchical class structures. Through comprehensive experiments, we show that our method is robust and outperforms current state-of-the-art approaches on two out of three HTC benchmark datasets.

Keywords: Large language models; discriminative language models; hierarchical text classification; prompt tuning

1. Introduction

The objective of hierarchical text classification (HTC) is to assign a set of labels from a structured class hierarchy to a text document. HTC is a particular case of multi-label text classification, where a text document belongs to one or more paths in the class hierarchy as opposed to standard multi-label classification where a document belongs to one or more classes.

Many approaches have been proposed which aim to leverage the structured class hierarchy in order to improve performance on HTC tasks. Recent approaches (Zhou *et al.* 2020; Deng *et al.* 2021; Chen *et al.* 2021; Jiang *et al.* 2022, Wang *et al.* 2022a; Jiang *et al.* 2022) use graph encoders to obtain feature representations of the class hierarchy and incorporate these representations into the classification process in various ways. Moreover, state-of-the-art approaches (Jiang *et al.* 2022; Huang *et al.* 2022; Wang *et al.* 2022b) leverage pre-trained language models (PLMs) to solve HTC tasks. Wang *et al.* (2022b) state that even though the vanilla fine-tuning paradigm has proven successful, recent studies indicate that this paradigm may suffer from the gap between the pre-training and fine-tuning phases such that the fine-tuned models are unable to effectively utilise the knowledge of PLMs (Chen *et al.* 2022).

Prompt tuning (Schick and Schütze 2021) aims to reduce this gap by transforming the input sequence such that the downstream task resembles the pre-training task of the PLM. Therefore, Wang *et al.* (2022b) proposed the Hierarchy-aware Prompt Tuning (HPT) approach which effectively solves HTC tasks through the prompt tuning paradigm for PLMs that use a Masked Language Modelling (MLM) pre-training task (Devlin *et al.* 2019). Furthermore, Yao *et al.* (2022) proposed the Prompt Tuning framework for Discriminative PLMs (DPT) which applies the prompt tuning paradigm to discriminative PLMs (Clark *et al.* 2020; He, Gao, and Chen 2021a) to solve flat text classification tasks more effectively than vanilla fine-tuning approaches. However, the DPT approach requires representations of each class to form part of the input sequence which is fed to the PLM. Therefore, DPT does not scale to classification tasks with a large number of classes due to the sequence length limits of PLMs which typically only allow 512 input tokens.

In this paper, we propose a new approach for HTC tasks called Hierarchy-aware Prompt Tuning for Discriminative PLMs (HPTD). HPTD applies the prompt tuning paradigm to discriminative PLMs for HTC tasks by transforming the input sequence to resemble the replaced token detection (RTD) task used during pre-training. Our approach reduces the gap between the pre-training and fine-tuning phases, thereby utilising the discriminative PLM more effectively compared to previous approaches.

We make the following contributions:

- We propose a novel HPTD approach which uses hierarchy-aware prompts to fine-tune discriminative PLMs for HTC tasks.
- We propose several improvements to the DPT method that allows more space to be used by text tokens in the input sequence. This is achieved by assigning the same position IDs to class representations that belong to the same level of the hierarchy, as well as using a learnable embedding representation for each class.
- We show the effectiveness of our method by achieving state-of-the-art results on two out of three commonly used benchmark datasets.
- We analyse the robustness of our approach through rigorous experiments which include an ablation study, stability analysis, and performance changes under simulated low-resource settings.

2. Related work

2.1 Pre-trained language models

In recent years, the field of natural language processing (NLP) has undergone a significant paradigm shift due to the advent of pre-training language models through self-supervised tasks on unlabelled textual data.

Devlin *et al.* (2019) proposed the Bidirectional Encoder Representations from Transformers (BERT) model which uses the MLM and Next Sentence Prediction (NSP) pre-training tasks. MLM involves randomly masking a certain percentage of tokens in the input sequence and instructing the model to predict the masked tokens, while the NSP objective requires the model to predict whether two sentences follow each other or not.

He *et al.* (2021b) proposed Decoding-enhanced BERT with disentangled attention (DeBERTa) which improves BERT through an attention mechanism where attention weights between tokens are calculated using separate matrices on their contents and relative positions. Furthermore, DeBERTa uses an enhanced mask decoder which incorporates the absolute positions of tokens in the decoding layer during the MLM pre-training task.

Clark *et al.* (2020) proposed a pre-training approach called Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA). ELECTRA uses a generator-discriminator framework, where a generator corrupts the input token sequence, while the discriminator has the objective of distinguishing between original tokens and tokens that have been corrupted by the generator. The generator is trained to perform the MLM task as in the BERT architecture, by randomly masking a percentage of tokens and learning to predict the masked tokens. Therefore, the original input sequence is passed to the generator, which converts certain tokens to `[MASK]` tokens and 'fills' them with predicted tokens to create a corrupted token sequence. The corrupted token sequence is passed to the discriminator which uses the RTD task to predict which tokens were replaced by classifying each token as `original` or `replaced`. This is done by passing the final hidden state of each token through a fully connected layer which outputs a score that represents the probability of the token being `original` or `replaced`. During pre-training, the parameters of the embedding layer are shared between the generator and discriminator to leverage the effectiveness of the MLM task in learning word embeddings. After the pre-training phase, the generator is typically removed and the discriminator is used as a PLM which can be fine-tuned on downstream tasks.

He *et al.* (2021a) proposed DeBERTaV3 which improves the previous DeBERTa versions by replacing the MLM pre-training task with RTD. However, DeBERTaV3 uses a gradient-disentangled embedding sharing method between the generator and discriminator which improves the efficiency of training and model performance. The gradient-disentangled embedding sharing method shares the embedding layer parameters between the generator and discriminator as in the ELECTRA model but does not allow the error of the discriminator to influence the optimisation of the generator, thereby preventing the 'tug-of-war' dynamics which are caused by the conflicting objectives of the generator and discriminator.

2.2 Fine-tuning for text classification

Vanilla fine-tuning for text classification typically utilises the classification token (`[CLS]`) which is prepended to the start of the input sequence and fed to the PLM. The PLM obtains the final hidden representation of the `[CLS]` token and passes it through an additional classification head. Dodge *et al.* (2020) show that the vanilla fine-tuning paradigm can produce highly unstable results, where random weight initialisation and training data orders lead to significant variance in performance.

Prompt tuning is a fine-tuning approach which aims to minimise the gap between pre-training and fine-tuning by transforming the input sequence such that the downstream task resembles the pre-training task of the PLM (Schick and Schütze 2021). The rationale behind this approach is that the model is able to utilise the knowledge of the PLM more effectively during the fine-tuning phase which improves performance on downstream NLP tasks.

Prompt tuning approaches have mostly been applied to PLMs which use the MLM pre-training task (Schick and Schütze 2021; Hambardzumyan, Khachatrian, and May 2021; Qin and Eisner 2021). For example, they transform the input token sequence $\mathbf{x} = [x_1, \ldots, x_T]$, where T is the number of tokens in the sequence, to '$\mathbf{x}$ is about `[MASK]`' and pass it to the PLM which tries to predict the `[MASK]` token.

Prompt tuning is broadly divided into two categories: hard prompts (Schick and Schütze 2021) which select existing tokens from the vocabulary of the PLM to form the prompts that are added to the input sequence (as in the example above), and soft prompts (Hambardzumyan *et al.* 2021; Qin and Eisner 2021) which initialise a continuous vector representation for prompts and learn these representations during training so that manually chosen prompts are not required.

2.3 Hierarchical text classification

HTC problems define a structured class hierarchy for each dataset as a directed acyclic graph $H = (C, E)$. $C = \{c_1, \ldots, c_L\}$ is the set of all class nodes, where L is the total number of classes and E is the set of all edges that form the hierarchical structure's parent–child relations. The goal of

HTC is to categorise a text document $\mathbf{x} = [x_1, \ldots, x_T]$, where T is the number of tokens in the text document, into a label set $Y \subseteq C$. Specifically, we assume that each node, apart from the root node, has only one parent node such that the graph describes a tree structure. Therefore, the label set Y comprises one or more paths in the hierarchical class structure H. Existing solutions for HTC problems attempt to leverage the class hierarchy information through various methods which are categorised into two groups: local and global approaches.

Local approaches leverage the hierarchical class structure by constructing one or more classifiers at each node, each non-leaf node, or each level in the class hierarchy and combining the results from these 'local classifiers' to form the final prediction (Koller and Sahami 1997; Dumais and Chen 2000; Kowsari *et al.* 2017; Shimura, Li, and Fukumoto 2018; Banerjee *et al.* 2019). Kowsari *et al.* (2017) proposed a model that comprises a deep neural network classifier for each non-leaf node in the structured class hierarchy. Other local approaches use the transfer learning paradigm to initialise the weights of classifiers in lower levels with the weights of their parent classifiers (Shimura *et al.* 2018; Banerjee *et al.* 2019).

Global approaches build a single classification model which considers the overall hierarchical class structure (Gopal and Yang 2013; Mao *et al.* 2019; Wu, Xiong, and Wang 2019; Zhou *et al.* 2020; Peng *et al.* 2021; Deng *et al.* 2021; Chen *et al.* 2021; Huang *et al.* 2022; Wang *et al.* 2022a; Jiang *et al.* 2022; du Toit and Dunaiski 2023). Some of the earlier proposed global methods used recursive regularisation (Gopal and Yang 2013), reinforcement learning (Mao *et al.* 2019), meta-learning (Wu *et al.* 2019), and capsule networks (Peng *et al.* 2021). Gargiulo *et al.* (2019) also proposed an approach which regularised the class hierarchy and incorporated the structural information into the classifier model to solve HTC tasks with an extremely high number of classes such as the large-scale biomedical semantic indexing and question-answering challenge (BioASQ) (Nentidis *et al.* 2024). However, more recent work has shown that encoding the hierarchical class structure through a graph encoder and incorporating this information into the classification process can significantly improve performance (Zhou *et al.* 2020; Deng *et al.* 2021; Chen *et al.* 2021; Jiang *et al.* 2022; Wang *et al.* 2022a, b).

Chen *et al.* (2021) proposed the Hierarchy-aware label semantics Matching network (HiMatch) which formulates HTC as a semantic matching problem by mapping the text embeddings and class hierarchy into the same embedding space. Wang *et al.* (2022a) proposed Hierarchy-Guided Contrastive Learning (HGCLR) which tries to embed the hierarchical class structure information into BERT through a contrastive learning approach. Huang *et al.* (2022) introduced the Hierarchy-Aware T5 model with Path-Adaptive Attention Mechanism (PAAMHiA-T5) which uses a T5 model (Raffel *et al.* 2020) for level-dependent label generation and the path-adaptive attention mechanism for path-specific label generation. Jiang *et al.* (2022) proposed Hierarchy-guided BERT with Global and Local hierarchies (HBGL) which uses BERT as a structure encoder to obtain hierarchical class embeddings which are fed to BERT along with the input text, and a mask token for each level in the hierarchy such that it predicts the labels at each level by generating the tokens to fill the masked tokens.

Wang *et al.* (2022b) proposed the HPT method which aims to bridge the gap between the MLM pre-training task of BERT and the downstream HTC task. HPT modifies the input sequence that is fed to BERT to transform the HTC task into a MLM task. Suppose the input text token sequence is given as $\mathbf{x}$ and K is the number of levels in the hierarchical class structure. HPT transforms the input sequence which is fed to BERT to '$\mathbf{x}$ `[V`$_1$`] [PRED]` $\cdots$ `[V`$_K$`] [PRED]`', where the task is to predict the `[PRED]` tokens. The prompt tokens (`[V`$_1$`]` $\cdots$ `[V`$_K$`]`) are soft prompts which represent the prompts for each level in the class hierarchy. HPT uses a graph attention network (GAT) (Veličković et al., 2018) which aggregates information from the classes in each of the levels to obtain hierarchy-aware prompt embeddings. To obtain the level-wise prompt embeddings, they create K virtual nodes $(P_1, \ldots, P_K)$ and connect each virtual node P_i to all of the nodes in the i-th level of the class hierarchy. This allows the virtual nodes to capture all required node information of the associated level. The embeddings for the virtual nodes are randomly initialised, and the embeddings for the class nodes are initialised by averaging the embeddings of their class name tokens. These nodes and their associated connections are passed through the GAT to obtain the embeddings for each of the virtual nodes which are used to initialise the embeddings of the level-wise prompt tokens. The modified input sequence is fed through BERT which obtains the final hidden states for each of the tokens. The final hidden states of the `[PRED]` tokens are used for classification in each of the associated levels. Furthermore, HPT uses a zero-bounded multi-label cross-entropy loss function as opposed to the standard binary cross-entropy loss function.

2.4 Prompt tuning for discriminative language models

Yao *et al.* (2022) introduced the DPT method which applies the prompt tuning paradigm to discriminative PLMs to solve NLP tasks such as text classification. For a flat text classification task, let L be the number of classes such that $\{c_1, \dots, c_L\}$ is the class set and $\{t_1, \dots, t_L\}$ are the associated class name tokens. For a text token sequence $\mathbf{x}$, DPT modifies the input sequence to '$\mathbf{x}$ Class: $t_1, \dots, t_L$', where the model's task is to predict whether each of the class tokens $\{t_1, \dots, t_L\}$ are `original` (correct) or were `replaced` (incorrect). For class names with more than one token, they include each of the tokens in the input sequence and use the first token as a representation of the class in the classification step. The modified input sequence is fed to the discriminative PLM which obtains the final hidden state for each token. The score for class c_i is computed as $s_i = 1 - \sigma(\mathbf{h}_{\text{RTD}}\mathbf{h}_{t_i})$, where σ is the sigmoid function, $\mathbf{h}_{t_i}$ is the final hidden state of token t_i, and $\mathbf{h}_{\text{RTD}}$ is the RTD head, that is, the final layer of the PLM that maps the hidden states to an output representing whether the token is `original` or `replaced`. Therefore, a class is assigned to a text instance if the class score for that instance is above a predefined threshold. The model uses these class scores in a binary cross-entropy loss function to compute the loss during training as:

$$\mathscr{L} = -\sum_{i=1}^{L}\left[y_i \log(s_i) + (1 - y_i)\log(1 - s_i)\right] \tag{1}$$

where $y_i \in \{0, 1\}$ is the ground-truth label for c_i.

3. Methodology

In this section, we define our HTC approach named HPTD. First, we describe the process used to obtain prompts for each level of the class hierarchy and to incorporate this information into the classifier model. We then provide a description of our model architecture as illustrated in Fig. 1. Finally, we describe our proposed adaptions to the DPT method which allows more space to be used by text tokens in the input sequence.

3.1 Hierarchy-aware prompts

We follow the same approach as HPT to construct prompts for each of the levels in the hierarchy to incorporate the hierarchical class structure during the fine-tuning process. Fig. 2 shows an example of the high-level procedure used to initialise the embeddings of the hierarchical prompts. We construct a graph that represents the hierarchical class structure of the associated task and initialise the embeddings associated with each node in the graph as the average of the embeddings of their class name tokens. Furthermore, we randomly initialise the embeddings of K virtual nodes $P_1 \cdots P_K$ and connect each node $P_i \forall i \in \{1, \dots, K\}$ with each of the nodes in the i-th level of the graph so that each virtual node can capture the information of its associated level. This graph structure is passed through a GAT which aggregates the information between the classes in each level to the virtual nodes. Finally, the embeddings of the virtual nodes are used to initialise the embeddings of the first prompt at each level ($\mathbf{p}_{1,1} \cdots \mathbf{p}_{K,1}$).

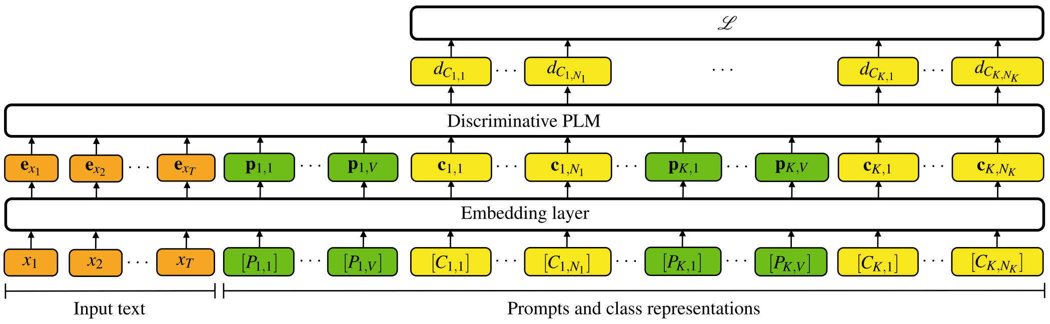


Figure 1. The HPTD architecture during training. HPTD modifies the text input sequence of length T (orange) by appending a template that comprises V prompts for each of the K levels in the class hierarchy (green), followed by a representation of each class in the associated level (yellow). The tokens are passed through the discriminative PLM to obtain the output for the class tokens which are used to drive the training process.

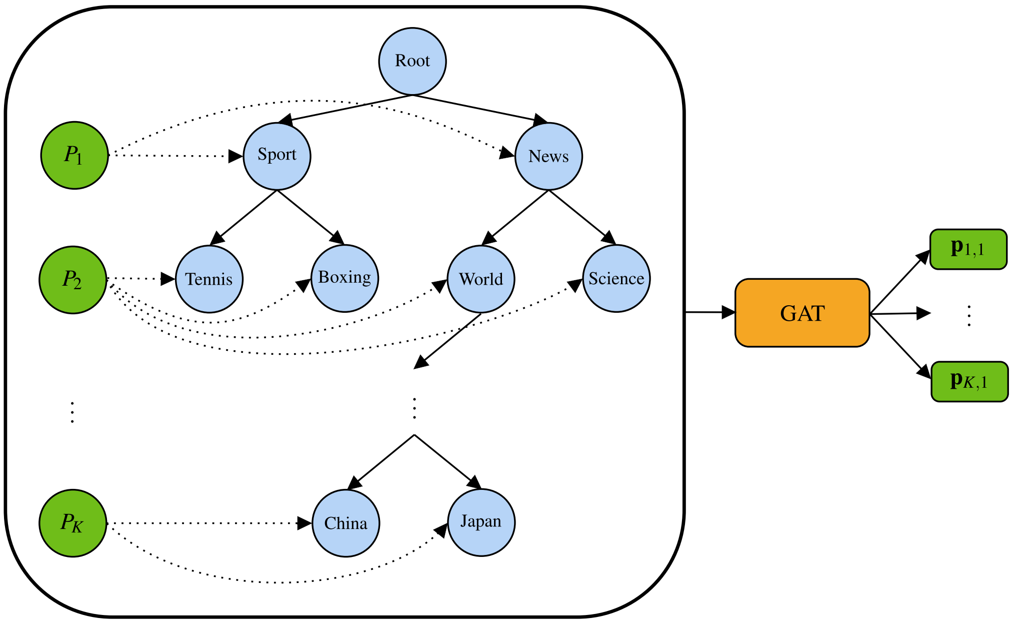


Figure 2. High-level procedure used to initialise the hierarchical prompt embeddings. We construct the hierarchical class graph and attach virtual nodes which form the prompts associated with each level. The graph is passed through a GAT which obtains the embeddings for the prompts associated with each level.

3.2 Model architecture

Suppose we have a hierarchical class structure with K levels, and N_i is the number of classes in level $i \in \{1, \ldots, K\}$ of the hierarchy. Furthermore, we have a text token sequence $\mathbf{x}$, class tokens $[C_{i,j}]$ which represent classes $j \in \{1, \ldots, N_i\}$ at levels i, and prompt tokens $[P_{i,v}]$ which represent prompts $v \in \{1, \ldots, V\}$, where V is the number of prompts per level, at levels i. HPTD modifies the input sequence to transform the HTC task, so it resembles the RTD task which predicts each of the class tokens ($[C_{i,j}]$) as `original` or `replaced`. Therefore, the input sequence which is fed to the discriminative PLM is constructed as follows:

$$\mathbf{x}\,[P_{1,1}] \cdots [P_{1,V}]\,[C_{1,1}] \cdots [C_{1,N_1}] \cdots [P_{K,1}] \cdots [P_{K,V}]\,[C_{K,1}] \cdots [C_{K,N_K}] \tag{2}$$

We omit the `[CLS]`, `[SEP]`, and `[PAD]` tokens for sake of clarity. The `[PAD]` tokens are added before the prompt and class tokens for sequences that use less than 512 tokens, so the prompt and class tokens always have the same position in the input sequence. The PLM converts each token to its vector representation through the embedding layer to obtain:

$$\mathbf{X}, \mathbf{p}_{1,1}, \ldots, \mathbf{p}_{1,V}, \mathbf{c}_{1,1}, \ldots, \mathbf{c}_{1,N_1}, \ldots, \mathbf{p}_{K,1}, \ldots, \mathbf{p}_{K,V}, \mathbf{c}_{K,1}, \ldots, \mathbf{c}_{K,N_K} \tag{3}$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ are the embeddings of the input text tokens, $\mathbf{p}_{i,v}$ is the prompt embedding for the v-th prompt at level i, and $\mathbf{c}_{i,j}$ is the class embedding for the j-th class at level i. The embeddings of the class tokens are initialised with the average of their class name token embeddings, while the embeddings of the first prompt token at each level ($\mathbf{p}_{1,1} \cdots \mathbf{p}_{K,1}$) are initialised with the virtual node embeddings from the procedure described above. These embeddings are passed through the discriminative PLM which obtains the output for each class token as:

$$d_{C_{1,1}}, \ldots, d_{C_{1,N_1}}, \ldots, d_{C_{K,1}}, \ldots, d_{C_{K,N_K}} \tag{4}$$

where $d_{C_{i,j}}$ is the output score of the j-th class at level i.

The class score that represents the confidence that the text document belongs to the j-th class at level i is calculated as $s_{i,j} = 1 - \sigma(d_{C_{i,j}})$. The rational behind this formulation of the class scores is based on the fact that discriminative PLMs' objective is to assign small scores to `original` tokens and large scores to `replaced` tokens (Yao *et al.* 2022). The model uses these class scores to compute the classification loss using a binary cross-entropy loss function as follows:

$$\mathscr{L} = -\sum_{i=1}^{K}\sum_{j=1}^{N_i}\left[y_{i,j}\log(s_{i,j}) + (1 - y_{i,j})\log(1 - s_{i,j})\right] \tag{5}$$

where $y_{i,j} \in \{0, 1\}$ is the ground-truth label for the j-th class in level i.

During inference, the class scores are used to predict the multi-hot vector $\mathbf{Y}' = [y'_{1,1}, \ldots, y'_{K,N_K}]$ which represents the class set associated with the text document and is calculated as:

$$y'_{i,j} = \begin{cases} 1, & s_{i,j} \geq \gamma \\ 0, & s_{i,j} < \gamma \end{cases} \tag{6}$$

where γ is a threshold that determines whether the class is assigned to the document or not.

3.3 Adaptions to DPT

As mentioned previously, the major shortcoming of the DPT approach for text classification tasks is that it transforms the input sequence to include the tokens associated with each of the class names in the label set. This significantly reduces the available space for text tokens in the input sequence, given the limited token sequence lengths in PLMs. This renders the approach infeasible for text classification problems with a large number of classes because there is little or no space left for the input text tokens.

However, the only limitation that prevents input sequences from being longer than the model's predefined token sequence length are the position embeddings assigned to tokens. These position embeddings are required, since transformer architectures do not naturally capture positional information (Vaswani *et al.* 2017). Each word is assigned a position ID based on its position in the input sequence. PLMs typically use a maximum sequence length of 512, so position IDs are assigned from 0 to 511 sequentially for each of the tokens in the input sequence. These position IDs are mapped to position embeddings through the position embedding matrix, which is learnt during training and has a corresponding length of 512. Therefore, the PLM does not accept position IDs larger than 511, as it does not have a corresponding entry in the position embedding matrix.

Table 1. Comparison of token space available for input text tokens using the DPT and HPTD approaches for different illustrative hierarchical class structures. The column ‘Additional tokens’ shows the improvement of the HPTD approach in terms of usable tokens for input text

		DPT		HPTD		
Levels	Classes	Tokens	% Tokens	Tokens	% Tokens	Additional tokens
2	50	460	89.84	508	99.21	+ 48
2	200	310	60.54	508	99.21	+ 198
2	800	0	0	508	99.21	+ 798
8	50	454	88.67	496	96.88	+ 42
8	200	304	59.38	496	96.88	+ 192
8	800	0	0	496	96.88	+ 792

To address this limitation of DPT, we assign the same position IDs to class representations in the same level and use a single learnable vector representation for each class.

3.3.1 Position IDs

Given the structure of the input sequence formed by our approach, the position ID assignments may be exploited to avoid the limitation of DPT. Since the objective of the model is to determine whether the class representations after a level-wise prompt are `original` or `replaced`, we argue that the positional information of each class in the same level may be reused. Therefore, we assign the same position IDs to all of the class tokens in the same level because these classes have no inherent ordering. This allows HPTD to scale to HTC tasks with much larger hierarchical class structures while maintaining many more input text tokens than the DPT approach. Table 1 shows a comparative illustration of the models’ encoding efficiencies by using this approach of position ID assignments in HPTD. We assume that the sequence length is limited to 512 tokens and the hierarchical prompts and class representations are included in the input sequence as described in Section 3.2, where the number of prompts per level (V) is 1. The columns ‘Levels’ and ‘Classes’ give the number of levels and classes in the class structure, while ‘Tokens’ is the number of tokens that can be used for input sequence text and ‘% Tokens’ is the percentage of the maximum input sequence length (512) that is available for the text tokens to occupy. ‘Additional tokens’ is the number of additional tokens that is passed to the model in HPTD over the 512 token sequence length of DPT. From this table, we can see that our proposed approach enables many more input text tokens to form part of the sequence that is passed to the PLM. However, as shown in the ‘Additional tokens’ column, this leads to a larger number of total tokens forming part of the input sequence. Therefore, our approach becomes computationally expensive for tasks with a very large number of classes due to the quadratic computational complexity scaling of the attention mechanism in transformer-based PLMs which computes the attention scores between each pair of tokens. This implies that our approach may become infeasible for tasks with extremely large hierarchical class structures due to limited computational resources.

3.3.2 Soft class representations

DPT uses the tokens of the class name descriptions to represent each of the classes in the input sequence. Therefore, for class names that comprise multiple tokens, each of the tokens are added to the input sequence and the first of these tokens is used to obtain the class score. We improve

Table 2. Characteristics of the benchmark HTC datasets. The columns 'Levels' and 'Classes' give the number of levels and classes in the class structure. 'Avg. Classes' is the average number of classes per document, while 'Train', 'Dev', and 'Test' are the number of instances in each of the dataset splits

Dataset	Levels	Classes	Avg. Classes	Train	Dev	Test
WOS	2	141	2.0	30,070	7,518	9,397
RCV1-V2	4	103	3.24	20,833	2,316	781,265
NYT	8	166	7.6	23,345	5,834	7,292

Table 3. The average per-level branching factor of the hierarchy in each benchmark dataset, which is calculated as the average number of child nodes for the nodes at a particular level. The number of nodes per level is given in parentheses

Dataset	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 8
WOS	19.14 (7)	0.0 (134)	–	–	–	–	–	–
RCV1-V2	13.75 (4)	0.78 (55)	0.02 (43)	0.0 (1)	–	–	–	–
NYT	6.75 (4)	1.89 (27)	0.92 (51)	0.36 (47)	0.71 (17)	0.5 (12)	0.33 (6)	0.0 (2)

this approach by creating a single learnable vector representation for each class token ($[C_{i,j}]$) and initialise its embedding ($\mathbf{c}_{i,j}$) with the average of its token embeddings. This further increases the number of input text tokens that can form part of the input sequence compared to the DPT approach as each class only requires one token.

4. Experiments

4.1 Datasets

To evaluate the performance of the proposed method, we perform experiments on the three most popular HTC benchmark datasets: Web-Of-Science (WOS) (Kowsari *et al.* 2017), RCV1-V2 (Lewis *et al.* 2004), and NYTimes (NYT) (Sandhaus 2008). For each of these datasets, we use the same preprocessing steps and dataset splits as in previous works (Zhou *et al.* 2020; Deng *et al.* 2021; Chen *et al.* 2021; Jiang *et al.* 2022; Wang *et al.* 2022a; Huang *et al.* 2022,b). Tables 2 and 3 show the summary statistics and hierarchical properties of these datasets.

4.2 Threshold selection

The commonly used approach for multi-label text classification tasks is to select a predefined and static threshold value $\gamma = 0.5$. We compare this approach to three variations where γ is tuned during the training process using the development sets of the associated datasets. The variations are as follows:

1. A single tuned threshold γ for all classes in the class hierarchy.
2. A threshold $\gamma_k \, \forall \, k \in [1, K]$ for each level.
3. A threshold $\gamma_l \, \forall \, l \in [1, L]$ for each class.

We use a bootstrapping technique to determine the thresholds for each of these approaches. We randomly sample 10% of instances from the development set 100 times and calculate the Macro-F1 score for each sample for each of the possible thresholds. We considered the set

Table 4. Performance comparisons of the HPTD approach using the three commonly used benchmark datasets. Standard deviations for the proposed methods are given in parentheses

Model	WOS		RCV1-V2		NYT	
Model	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
HiMatch	86.20	80.53	84.73	64.11	–	–
HGCLR	87.11	81.20	86.49	68.31	78.86	67.96
PAAMHiA-T5	<u>90.36</u>	81.64	87.22	70.02	77.52	65.97
HBGL	87.36	82.00	87.23	**71.07**	80.47	70.19
HPT	87.16	81.93	**87.26**	69.53	80.42	70.42
HPTD-ELECTRA	87.45 (0.16)	81.67 (0.12)	86.30 (0.17)	68.12 (0.97)	80.54 (0.05)	70.66 (0.14)
HPTD-DeBERTaV3	**87.85** (0.13)	**82.13** (0.34)	86.25 (0.19)	66.85 (1.33)	**81.45** (0.14)	**72.40** (0.41)

{0.2, 0.3, 0.4, 0.5, 0.6, 0.7} as possible values for the thresholds. The threshold for which the highest average Macro-F1 score is achieved over the 100 samples is chosen as the single threshold (variation 1), or the threshold for the associated level (variation 2) or class (variation 3).

4.3 Implementation details

We implemented the HPTD approach using PyTorch, PyTorch Lightning, and Hugging Face. For fair comparisons to previous work that utilises BERT architectures (Chen *et al.* 2021; Jiang *et al.* 2022; Wang *et al.* 2022a, b), we use `electra-base-discriminator` and `deberta-v3-base` which has the same number of model parameters as the `bert-base-uncased` model used by these approaches. However, it should be noted that DeBERTaV3 uses a vocabulary size of 128K as opposed to BERT and ELECTRA that only use 30K. We used the Adam optimiser (Kingma and Ba 2015) and performed hyperparameter tuning on the learning rate and batch size with possible values of {8e-6, 1e-5, 1.5e-5, 2e-5} and {16, 32}, respectively. We set the number of prompts per level (V) to 4. We trained our models on a single V100 GPU with training times ranging between 1 and 2 hours depending on the dataset and hyperparameter selection.

We trained the model on the training set for a maximum of 20 epochs and stopped training when the Macro-F1 score on the development set did not increase for 5 epochs. We chose the model and hyperparameter combination which obtained the highest Macro-F1 score on the development set to evaluate the performance of our models on the test set.

4.4 Main results

In Table 4, we present the results of our approach compared to the most recent HTC approaches using the three commonly used benchmark datasets. It should be noted that the PAAMHiA-T5 approach (Huang *et al.* 2022) uses an underlying PLM with twice the number of model parameters as the other approaches. We report the average performance from three runs with different random seeds while using a fixed threshold (γ) of 0.5 because previous approaches did not tune this threshold. We use ELECTRA and DeBERTaV3 as the discriminative PLMs in our approach and refer to the two models as HPTD-ELECTRA and HPTD-DeBERTaV3, respectively.

The results show that HPTD-DeBERTaV3 outperforms HPTD-ELECTRA on the WOS and NYT datasets and achieves comparable performance on the RCV1-V2 dataset. We hypothesise that HPTD-DeBERTaV3 generally outperforms HPTD-ELECTRA due to DeBERTaV3's

Table 5. Results on the benchmark datasets when removing components of the HPTD-ELECTRA model

	WOS		RCV1-V2		NYT	
Ablation models	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
HPTD-ELECTRA	**88.11**	**81.87**	87.12	**68.45**	**80.56**	**71.99**
-*r.m.* additional prompts	88.02	81.73	**87.38**	66.87	80.29	71.52
-*r.m.* hierarchical prompts	87.60	81.24	87.19	68.27	80.42	71.57
-*r.m.* prompt initialisation	87.77	81.64	86.93	68.00	80.46	71.89

improved performance on the RTD task. These improvements can be attributed to the attention mechanism that calculates the attention weights between tokens through separate matrices on their contents and relative positions, and the gradient-disentangled embedding sharing method which prevents the 'tug-of-war' dynamics where the generator and discriminator pull token embeddings in different directions during training.

HPTD-DeBERTaV3 outperforms the current state-of-the-art approaches (bar models with a significantly larger number of model parameters) on the WOS and NYT datasets. Moreover, HPTD-DeBERTaV3 significantly outperforms previous approaches on the NYT dataset with F1 score improvements of 0.98 (Micro) and 1.98 (Macro) percentage points. The NYT dataset has the most complex hierarchical class structure with the deepest hierarchy, which shows that our approach can leverage more complex hierarchical class structures effectively to improve classification performance. However, our approach did not improve on the HBGL and HPT approaches when evaluated on the RCV1-V2 dataset, which has 20,833 train and 781,265 test instances. We hypothesise that our approach struggles to consistently classify classes that are extremely scarce in the training data compared to previous approaches and provide further analysis in Section 4.6.

Table 4 also shows the standard deviation over the three runs for each dataset to give an indication of the stability of our two models and provide a baseline for future research.

The results show that even though HPTD-DeBERTaV3 generally outperforms HPTD-ELECTRA, using ELECTRA as the discriminative PLM in our approach improves the stability over multiple runs with different random seeds. Furthermore, the results show that the Macro-F1 scores are generally more inconsistent than Micro-F1 scores across multiple runs.

Furthermore, to determine whether the difference in performance between the HPTD-ELECTRA and HPTD-DeBERTaV3 models is statistically significant, we used a T-test with a significance level of 0.05. We found that the differences in performance are not statistically significant on the WOS dataset (with $P = 0.126$ and $P = 0.221$ for the Micro-F1 and Macro-F1 scores, respectively) but are statistically significant on the NYT dataset (with $P = 0.007$ and $P = 0.009$ for the Micro-F1 and Macro-F1 scores, respectively). For the RCV1-V2 dataset, the difference was statistically significant for the Macro-F1 ($P = 0.026$) scores, but not the Micro-F1 scores ($P = 0.163$).

4.5 Ablation study

To test the effectiveness of our approach, we remove some components of the model and observe the impact that it has on performance. Table 5 shows the ablation results for the HPTD-ELECTRA model on the development sets of the benchmark datasets. We discuss the ablation study on the NYT dataset as it has the deepest and most complex class hierarchy.

By removing the additional prompts, that is, only using one prompt per level, the Micro-F1 and Macro-F1 scores decrease by 0.27 and 0.47, respectively. This shows that even though the additional prompts leave less space for input text tokens, they can improve performance by providing better context of each level before the associated class representations.

The removal of the hierarchical prompts, that is, placing all the prompts before the class representations, also reduces the performance of the model in terms of Micro-F1 and Macro-F1 scores. Therefore, we believe that the placement of the prompts before the class representations of its associated level allow the model to leverage the hierarchical class structure more effectively.

Finally, by removing the initialisation step of the first prompt at each level, that is, randomly initialising all of the prompts, the Micro-F1 and Macro-F1 decrease with 0.12 and 0.10, respectively.

The ablation studies on the WOS and RCV1-V2 datasets generally show similarly decreasing performance when removing the various components of HPTD.

4.6 Level-wise results

Fig. 3 presents the Micro-F1 and Macro-F1 scores for the classes at each level for the three benchmark datasets. These figures show that the performance at a certain level is generally directly correlated with the average number of training instances per class at that level. We observe that the Micro-F1 and Macro-F1 scores on the WOS dataset significantly decrease from level 1 to level 2 as the average number of training instances per class reduces. A similar trend is seen in the four-level hierarchical class structure of the RCV1-V2 dataset, but interestingly the HPTD-ELECTRA model outperforms HPTD-DeBERTaV3 for the level 2 and 3 classes, while HPTD-DeBERTaV3 performs significantly better for level 4 which only has a single class. From these figures, we can see that our approaches struggle to accurately classify the lower-level classes of the RCV1-V2 dataset, especially in terms of Macro-F1 scores. This indicates that our approaches may not be able to consistently classify the classes with very few training instances compared to previous approaches. The level-wise results on the NYT dataset also show that the performance generally decreases for the levels with fewer training instances, apart from level 8 which has the lowest Micro-F1 and Macro-F1 scores even though it has a higher average number of training instances per class than levels 3 to 7. However, level 8 of the NYT dataset only has two classes which likely leads to greater variance in performance compared to levels with more classes.

4.7 Threshold selection results

Table 6 presents the results of the HPTD-ELECTRA and HPTD-DeBERTaV3 models for each of the threshold selection approaches.

The results show that a fixed threshold of 0.5 consistently achieves the highest Micro-F1 scores across the three datasets for both of our models. We believe that the other threshold selection approaches may overfit the development datasets, whereas the static 0.5 threshold generalises better to classes with many instances in the test set.

Furthermore, the per-class threshold selection approach generally obtains the best Macro-F1 scores across the three datasets. We hypothesise that this is because a tuned threshold for each class allows better performance on classes which generally require a lower or higher threshold. Therefore, a fixed threshold of 0.5 may be suitable for HTC tasks that require majority classes to be classified accurately, whereas a per-class tuned threshold may be beneficial for HTC tasks that require better accuracy across all classes.

4.8 Low-resource results

We simulate a low-resource setting to assess the robustness of our approach for HTC tasks when fewer training instances are available. Following Wang *et al.* (2022b), we randomly sample 10%

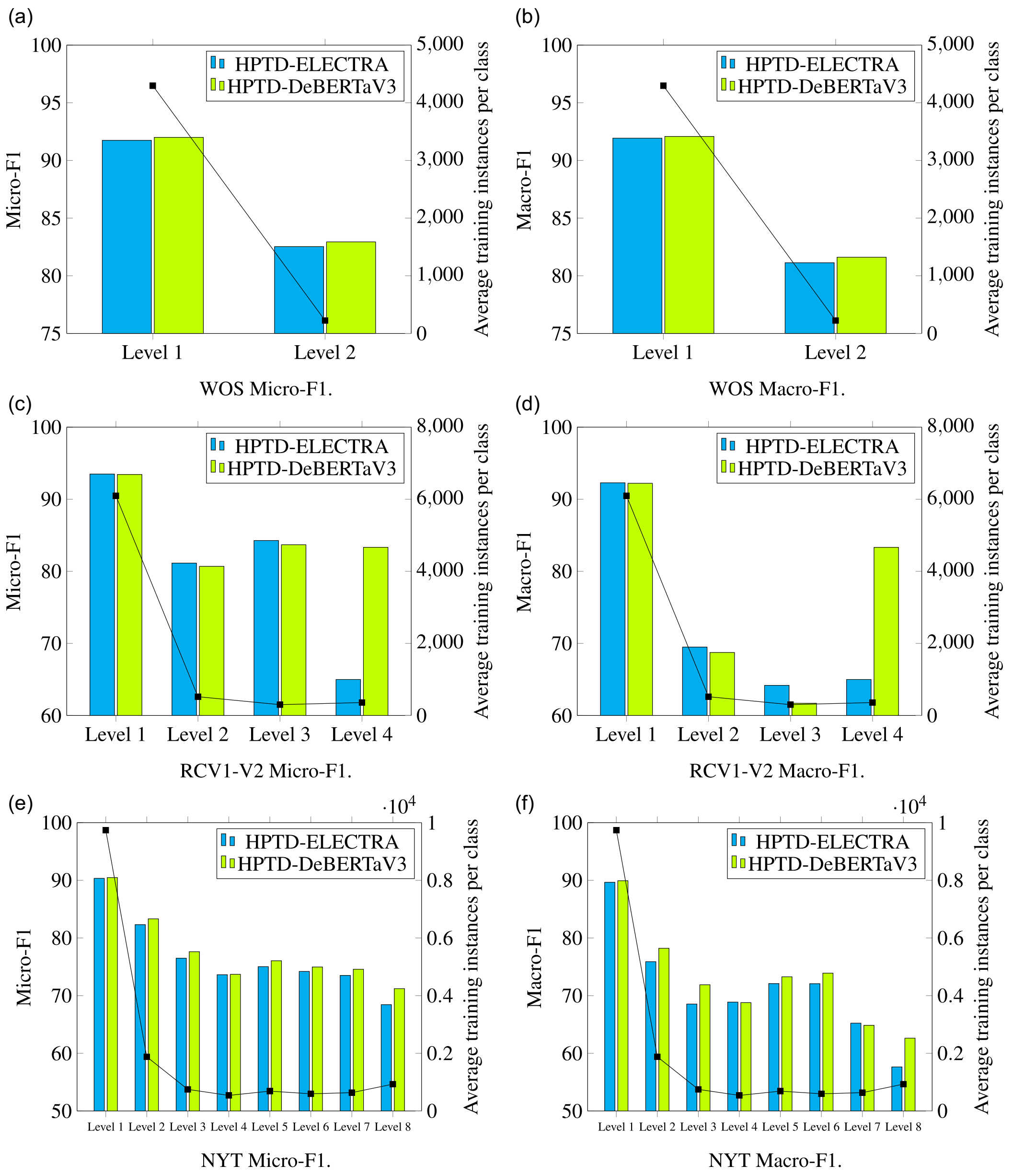


Figure 3. Level-wise performance results of our approaches on the three benchmark datasets. The bar plots present the F1 scores (left y-axis) for the different approaches at each level of the hierarchy, while the line plot shows the average training instances for the classes at a particular level (right y-axis).

of the training data to form the training set and keep the other settings the same. Our sampling method involves drawing random samples from the training set in a non-stratified manner. Table 7 shows the performance results of the HPTD models under the low-resource scenario. Again, the F1 scores are computed as the average over three runs using different random seeds.

Table 6. Performance comparisons of the different threshold selection approaches. ‘0.5’ uses a fixed 0.5 threshold for each class, while ‘Single’ uses a single tuned threshold for each class. ‘Level’ and ‘Class’ use a tuned threshold for each level and class, respectively

		WOS		RCV1-V2		NYT	
Model	Threshold	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
HPTD-ELECTRA	0.5	**87.45**	81.67	**86.30**	68.12	**80.54**	70.66
	Single	87.18	81.75	85.83	68.47	80.37	70.73
	Level	87.18	81.75	85.84	68.52	80.42	70.74
	Class	87.28	**81.93**	86.24	**68.57**	80.50	**70.83**
HPTD-DeBERTaV3	0.5	**87.85**	82.13	**86.25**	66.85	**81.45**	72.40
	Single	87.76	82.45	85.48	67.55	81.19	72.23
	Level	87.81	**82.46**	85.78	67.55	81.26	72.33
	Class	87.49	82.29	86.09	**67.84**	81.40	**72.56**

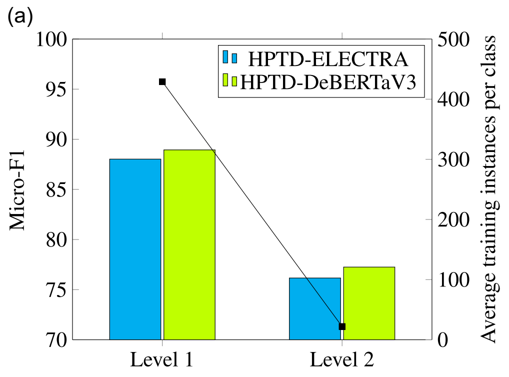
Table 7. Performance results of the HPTD models under a low-resource scenario where only 10% of the training data is used. For comparison, the achieved results when all training data is used are shown in parentheses

	WOS		RCV1-V2		NYT	
Model	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
HPTD-ELECTRA	82.34 (87.45)	74.75 (81.67)	**80.98** (86.30)	**49.07** (68.12)	75.00 (80.54)	61.28 (70.66)
HPTD-DeBERTaV3	**83.47** (87.85)	**75.74** (82.13)	79.42 (86.25)	45.16 (66.85)	**76.18** (81.45)	**63.38** (72.40)

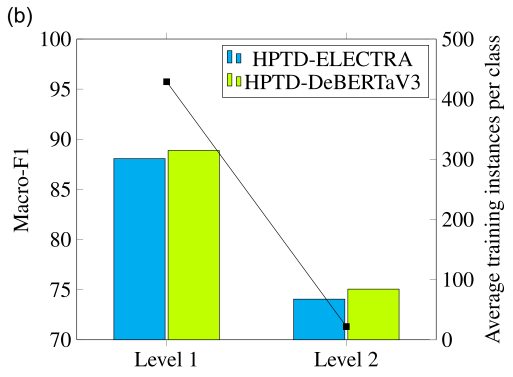
HPTD-DeBERTaV3 outperforms HPTD-ELECTRA in the low-resource scenario on the WOS and NYT datasets. We hypothesise that is due to the same reasons as mentioned in Section 4.4. However, HPTD-ELECTRA outperforms HPTD-DeBERTaV3 by 3.91 for the Macro-F1 score on the RCV1-V2 dataset, showing that it may be a more suitable approach for low-resource settings in certain cases.

For both of our models, the Micro-F1 scores do not decrease as significantly as the Macro-F1 scores when using less training data. This is likely due to the non-stratified random sampling approach used which may sample very few instances for particular classes.

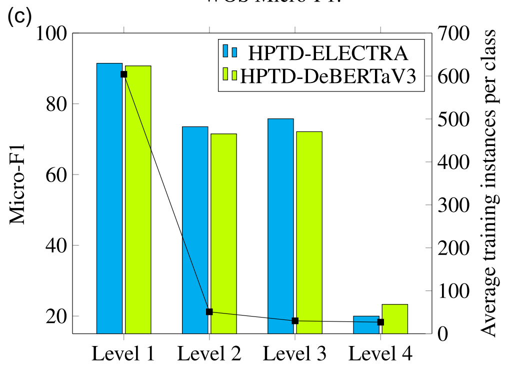
Fig. 4 presents the Micro-F1 and Macro-F1 scores for the classes at each level for the three benchmark datasets in the low-resource scenario. The results show that the classification performance generally decreases for the lower levels of the class hierarchy which have fewer average training instances per class, as in the full training data scenario. Furthermore, comparing these figures to the full training data level-wise results shown in Fig. 3, we observed that the Micro-F1 performance generally decreases consistently across the different levels of the class hierarchy. However, the first-level Macro-F1 scores on the RCV1-V2 and NYT datasets remain high in the low-resource setting, while the lower-level classes have a significant decrease in Macro-F1 scores when moving from the full training data to the low-resource scenario. Furthermore, the large difference in performance at the fourth level of the RCV1-V2 dataset in the full training data scenario was resolved in the low-resource scenario, since the HPTD-DeBERTaV3 model has too few training instances to effectively classify the single class at the fourth level.



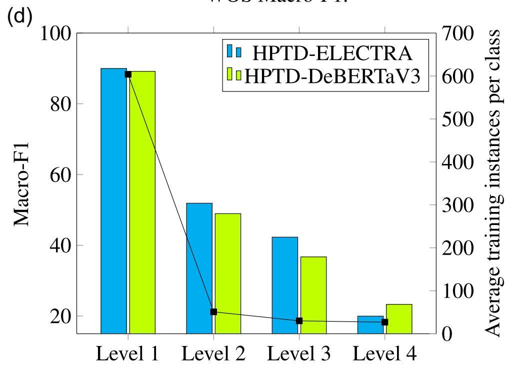
WOS Micro-F1.



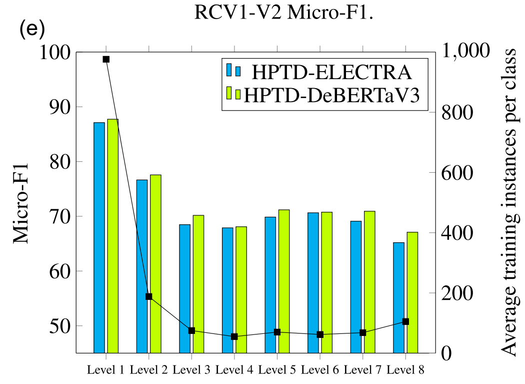
WOS Macro-F1.



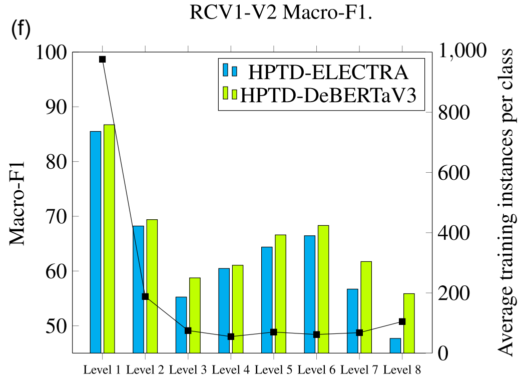
RCV1-V2 Micro-F1.



RCV1-V2 Macro-F1.



NYT Micro-F1.



NYT Macro-F1.

Figure 4. Level-wise performance results of our approaches on the three benchmark datasets in the low-resource scenario. The bar plots present the F1 scores (left *y*-axis) for the different approaches at each level of the hierarchy, while the line plot shows the average training instances for the classes at a particular level (right *y*-axis).

5. Conclusion

In this paper, we proposed the HPTD method which effectively classifies text documents into hierarchical class structures. HPTD transforms the input token sequence to include hierarchy-aware prompts followed by class representations for the classes at each level in the hierarchy. Therefore, HPTD injects the HTC task into the RTD pre-training objective used by discriminative language models and leverages the model more effectively by reducing the gap between the pre-training and fine-tuning phases. Our proposed improvements to the standard approach of prompt tuning discriminative language models allows us to scale hierarchical classification tasks to much larger class hierarchies. We performed extensive experiments which show that our approach is robust and outperforms current state-of-the-art approaches on two out of three benchmark datasets. Finally, analysis of different approaches to tune selection thresholds showed that a fixed threshold of 0.5 consistently obtains the best Micro-F1 scores, whereas tuning a selection threshold per class generally achieves the best Macro-F1 results. For future research, we will investigate the performance of various graph neural network architectures to encode the class hierarchy. Additionally, we will evaluate state-of-the-art HTC approaches using performance metrics that account for the hierarchical class structure, rather than relying solely on the Micro-F1 and Macro-F1 metrics which 'flatten' the hierarchy.

6. Limitations

Our approach leverages the RTD pre-training task that discriminative PLMs use during pre-training. Therefore, HPTD is not generalisable to PLMs which use different pre-training tasks. Furthermore, discriminative PLMs are constrained to a maximum sequence length. Although our approach improves the shortcoming of previous prompt tuning approaches for discriminative PLMs, the number of additional tokens that are added to the end of the input token sequence is proportional to the number of levels and classes in the class hierarchy. Therefore, our approach may be infeasible for HTC tasks with exceptionally deep hierarchical class structures. Finally, these added tokens lead to greater computational complexity such that HPTD may become exceedingly computationally expensive for HTC tasks with an extremely large number of classes.

References

Banerjee S., **Akkaya C.**, **Perez-Sorrosal F. and Tsioutsiouliklis K.** (2019). Hierarchical transfer learning for multi-label text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, *Florence, Italy*. Association for Computational Linguistics, pp. 6295–6300.

Chen H., **Ma Q.**, **Lin Z. and Yan J.** (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*, Online. Association for Computational Linguistics, pp. 4370–4379.

Chen X., **Zhang N.**, **Xie X.**, **Deng S.**, **Yao Y.**, **Tan C.**, **Huang F.**, **Si L. and Chen H.** (2022). Knowprompt: knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference, WWW'22*, *New York, NY, USA. Association for Computing Machinery*, pp. 2778–2788.

Clark K., **Luong M.**, **Le Q.V. and Manning C.D.** (2020). ELECTRA: pre-training text encoders as discriminators rather than generators. In *Proceedings of the 8th International Conference on Learning Representations*, *Addis Ababa, Ethiopia*. OpenReview.net.

Deng Z., **Peng H.**, **He D.**, **Li J. and Yu P.** (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*, Online. Association for Computational Linguistics, pp. 3259–3265.

Devlin J., **Chang M.-W.**, **Lee K. and Toutanova K.** (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Long and Short Papers.*Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, *Minneapolis, MN, USA*. Association for Computational Linguistics, vol **1**, pp. 4171–4186,

Dodge J., **Ilharco G.**, **Schwartz R.**, **Farhadi A.**, **Hajishirzi H. and Smith N.** (2020). Fine-tuning pretrained language models: weight initializations, data orders, and early stopping. arXiv preprint arXiv: 2002.06305.

du Toit J. and Dunaiski M. (2023). Hierarchical text classification using language models with global label-wise attention mechanisms. In *Artificial Intelligence Research*. Springer Nature Switzerland, pp. 267–284.

Dumais S. and Chen H. (2000). Hierarchical classification of web content. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '00*. New York, NY, USA: Association for Computing Machinery, pp. 256–263.

Gargiulo F., Silvestri S., Ciampi M. and De Pietro G. (2019). Deep neural network for hierarchical extreme multi-label text classification. *Applied Soft Computing* **79**, 125–138.

Gopal S. and Yang Y. (2013). Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13*. New York, NY, USA: Association for Computing Machinery, pp. 257–265.

Hambardzumyan K., Khachatrian H. and May J. (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)*, Online. Association for Computational Linguistics, pp. 4921–4933.

He P., Gao J. and Chen W. (2021a). DeBERTaV3: improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing.

He P., Liu X., Gao J. and Chen W. (2021b). DeBERTa: decoding-enhanced BERT with disentangled attention. In *9th International Conference on Learning Representations, Online*. OpenReview.net.

Huang W., Liu C., Xiao B., Zhao Y., Pan Z., Zhang Z., Yang X. and Liu G. (2022). Exploring label hierarchy in a generative way for hierarchical text classification. In *Proceedings of the 29th International Conference on Computational Linguistics, Gyeongju, Republic of Korea*. International Committee on Computational Linguistics, pp. 1116–1127.

Jiang T., Wang D., Sun L., Chen Z., Zhuang F. and Yang Q. (2022). Exploiting global and local hierarchies for hierarchical text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates*. Association for Computational Linguistics, pp. 4030–4039.

Kingma D. and Ba J. (2015). Adam: a method for stochastic optimization. In *3rd International Conference on Learning Representations, San Diego, CA, USA*.

Koller D. and Sahami M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning, ICML'97, San Francisco, CA, USA*. Morgan Kaufmann Publishers Inc, pp. 170–178.

Kowsari K., Brown D.E., Heidarysafa M., Jafari Meimandi K., Gerber M.S. and Barnes L.E. (2017). Hdltex: hierarchical deep learning for text classification. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 364–371.

Lewis D.D., Yang Y., Rose T.G. and Li F. (2004). RCV1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research* **5**, 361–397.

Mao Y., Tian J., Han J. and Ren X. (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), Hong Kong, China*. Association for Computational Linguistics, pp. 445–455.

Nentidis A., Krithara A., Paliouras G., Krallinger M., Sanchez L.G., Lima S., Farre E., Loukachevitch N., Davydova V. and Tutubalina E. (2024). Bioasq at clef2024: the twelfth edition of the large-scale biomedical semantic indexing and question answering challenge. In **Goharian N., Tonellotto N., He Y., Lipani A., McDonald G., Macdonald C. and Ounis I.** (eds), *Advances in Information Retrieval*. Cham: Springer Nature Switzerland, pp. 490–497.

Peng H., Li J., Wang S., Wang L., Gong Q., Yang R., Li B., Yu P.S. and He L. (2021). Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification. *IEEE Transactions on Knowledge and Data Engineering* **33**(6), 2505–2519.

Qin G. and Eisner J. (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*, Online. Association for Computational Linguistics, pp. 5203–5212.

Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W. and Liu P.J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* **21**(1).

Sandhaus E. (2008). The New York Times annotated corpus. Technical report, Linguistic Data Consortium, Philadelphia.

Schick T. and Schütze H. (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*, Online. Association for Computational Linguistics, pp. 255–269.

Shimura K., Li J. and Fukumoto F. (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*. Association for Computational Linguistics, pp. 811–816.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L. and Polosukhin I. (2017). Attention is all you need. In **Guyon I., Luxburg U. V., Bengio S., Wallach H., Fergus R., Vishwanathan S. and Garnett R.** (eds), *Advances in Neural Information Processing Systems*, vol. **30**. Curran Associates, Inc.

Veličković P., Cucurull G., Casanova A., Romero A., Liò P. and Bengio Y. (2018). Graph attention networks. In *International Conference on Learning Representations*. Accepted as poster.

Wang Z., Wang P., Huang L., Sun X. and Wang H. (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), Dublin, Ireland*. Association for Computational Linguistics, pp. 7109–7119.

Wang Z., Wang P., Liu T., Lin B., Cao Y., Sui Z. and Wang H. (2022b). HPT: hierarchy-aware prompt tuning for hierarchical text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, *Abu Dhabi, United Arab Emirates*. Association for Computational Linguistics, pp. 3740–3751.

Wu J., Xiong W. and Wang W. Y. (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*, *Hong Kong, China*. Association for Computational Linguistics, pp. 4354–4364.

Yao Y., Dong B., Zhang A., Zhang Z., Xie R., Liu Z., Lin L., Sun M. and Wang J. (2022). Prompt tuning for discriminative pre-trained language models. In *Findings of the Association for Computational Linguistics*, Dublin, Ireland, Association for Computational Linguistics, pp. 3468–3473.

Zhou J., Ma C., Long D., Xu G., Ding N., Zhang H., Xie P. and Liu G. (2020). Hierarchy-aware global model for hierarchical text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online, Association for Computational Linguistics, pp. 1106–1117.

Cite this article: du Toit J and Dunaiski M. Prompt tuning discriminative language models for hierarchical text classification. *Natural Language Processing* https://doi.org/10.1017/nlp.2024.51