# HiMTL: Hierarchy-aware Multi-Task Learning for Hierarchical Text Classification

Anonymous ACL submission

## Abstract

Hierarchical relationships between labels can be used to control the information flow in text classification models. However, while these models are required to distinguish nuances between closely related fine-grained labels, their weight updates are also influenced by unrelated branches of the hierarchy. In this paper, we show that systematically splitting the hierarchy into multiple sub-hierarchies, thus training multiple localized hierarchy-aware classification layers on top of a shared text encoder, can improve classification scores on simple and complex hierarchies. HiMTL is not a new model but an architectural extension that can be applied to different state-of-the-art hierarchical classification models.

## 1 Introduction

Hierarchical text classification is a special case of multi-label text classification where the labels underlie hierarchical relations. In this hierarchy, which can be either a tree or a directed acyclic graph (DAG), at least one path exists from the most coarse-grained true label to the most fine-grained true label. This structure can be used to control the global and local information flow in a classification model instead of treating all labels as independent as in a traditional flat classification layer. Here, *global* information refers to information about the hierarchical structure as a whole, whereas *local* information refers to information restricted to, for example, a specific level of the hierarchy (Silla and Freitas, 2011). While some architectures have shown that the combining of local and global predictions can be beneficial (Wehrmann et al., 2018; Qi and Chelmis, 2023; Liu et al., 2024), several other architectures have achieved state-of-the-art performance based purely on global information. Global approaches can be as simple as post-processing the outputs of a flat classifier based on logical constraints (Giunchiglia and Lukasiewicz, 2020), but more commonly solve the problem by placing a structure encoder between a text encoder and a classification layer (Zhou et al., 2020; Chen et al., 2021; Zhu et al., 2023).

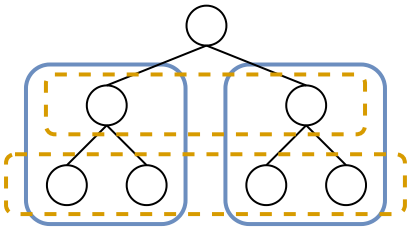

Figure 1: Previous methods split the hierarchy horizontally and train hierarchy-agnostic classifiers on each level (dashed orange boxes). HiMTL splits the hierarchy vertically at the root instead and trains hierarchy-aware classifiers on each branch (blue boxes).

What the aforementioned architectures combining local and global predictions have in common is that they split the hierarchy horizontally and train one predictor for each layer of the hierarchy. The resulting classifiers are not hierarchy-aware and need to distinguish between labels located on different branches of the hierarchy, but also between labels sharing the same parent node, which are less discriminative. We hypothesize that, instead of splitting the hierarchy horizontally, it can be beneficial to split it vertically and train one predictor on each branch starting from the root node. In other words, the hierarchy is split into multiple, more local sub-hierarchies, where each classifier only learns to discriminate between closely related labels without its weights being influenced by disjoint branches. These parallel classifiers can still make use of global information with respect to the specific sub-hierarchies they are trained on, and the losses of all classifiers eventually propagate back to a shared text encoder. The two different ways of splitting the hierarchy are compared in Figure 1.

In the following, we formulate this approach as a multi-task learning problem and propose different

ways to construct these parallel hierarchy-aware classifiers and combine their outputs into a single model prediction. Our implementation is available on GitHub.[1]

## 2 Related Work

Early neural-network-based hierarchical classification models extend feed-forward networks and LSTMs (Hochreiter and Schmidhuber, 1997) with per-level local predictions, using the hidden state from the preceding level. The final layer makes a global prediction for all labels in the hierarchy, which is then weighed with the local predictions (Wehrmann et al., 2018). A simplified architecture trains disjoint classifiers for each level of the hierarchy and produces a global prediction via a linear operation on the concatenated local predictions (Liu et al., 2024). Local predictions can also be made level by level as a sequence-to-sequence task with the T5 encoder-decoder architecture (Raffel et al., 2020) while constraining the decoder outputs to valid sequences according to the global structure of the hierarchy (Torba et al., 2024).

Several recent state-of-the-art models skip local predictions and process aggregate global information using a structure encoder. HiAGM (Zhou et al., 2020) encodes the hierarchy in a graph convolutional network (GCN) (Kipf and Welling, 2017) which receives a text encoding produced by GloVe (Pennington et al., 2014) or BERT (Devlin et al., 2019) as its input. HiMatch (Chen et al., 2021) employs two GCNs of the same structure, one for input text encodings and one for label description encodings, and minimizes the distances between their outputs in a shared latent space. Few-shot settings benefit from this approach. HiTIN (Zhu et al., 2023) reduces the amount of trainable parameters by replacing the GCN with a coding tree minimizing the structure entropy (Li and Pan, 2016) of the hierarchy.

The previous three architectures use recursive regularization (Gopal and Yang, 2013)

$$L^R(\mathbf{W}) = \sum_{p \in Y} \sum_{q \in \text{child}(p)} \frac{1}{2} \| w_p^2 - w_q^2 \| \quad (1)$$

to minimize the distance between node embeddings of adjacent labels from a label set $Y$.

A hierarchical violation occurs if the probability of a label is predicted to be greater than the probability of its parent. This can be used as a logical constraint to post-process model outputs using a simple maximum criterion, and explicitly integrating this post-processing into the binary cross-entropy function lets model with flat classification layers converge to better local optima if the hierarchy is large (Giunchiglia and Lukasiewicz, 2020).

Gradient Routing (Cloud et al., 2024) provides an alternative take on multi-task learning. Each training batch is task-specific and masks the gradient such that only specific parts of the neural network layers, i.e., interpretable routes through the network, are updated.

## 3 Methodology

A label hierarchy is a directed acyclic graph (DAG) $G(V, E)$ where the vertices (nodes) $V$ represent the labels, and the edges $E$ model the relationships between parent and child labels. If each child can only be reached by a single parent, $G$ is a tree. In practice, parent nodes are typically more general and easier to classify than their children.

Given training samples $(x, \boldsymbol{y})$ with texts $x$ and gold labels $\boldsymbol{y} \in \{0, 1\}^{|V|}$, we denote the output of a text encoder as a vector $\hat{\boldsymbol{x}} \in \mathbb{R}^d$. The structure encoder is a function $f_{\theta_G} : \mathbb{R}^d \to [0, 1]^{|V|} : \hat{\boldsymbol{x}} \mapsto \hat{\boldsymbol{y}}$ assigning probabilities $P(y_i = 1 \mid \mathrm{x} = x; \theta_G)$ to all $|V|$ labels, where $\theta_G$ are the trainable parameters of the encoding of $G$.

For instance, if the structure encoder is a graph convolutional network (GCN), then

$$f_{\theta_G} = \sigma\left(\overleftarrow{E} V \boldsymbol{W}_1 + \overrightarrow{E} V \boldsymbol{W}_2\right) \quad (2)$$

where $\overleftarrow{E}$ and $\overrightarrow{E}$ are in- and out-edges of $G$, $\boldsymbol{W}_i$ are weight matrices in $\theta_G$, and $\sigma$ is the sigmoid activation function transforming logits into probabilities.

### 3.1 Hierarchy-aware Multi-Task Learning

Given a hierarchy $G(V, E)$ with $n$ out-edges from the root node, we split $G$ into $n$ subgraphs, or sub-hierarchies, $G^{(i)}(V^{(i)}, E^{(i)})$ where $V^{(i)} \subseteq V$ and $E^{(i)} \subseteq E$. If $G$ is a tree, then

$$\bigcap_{i=1}^{n} V^{(i)} = \bigcap_{i=1}^{n} E^{(i)} = \emptyset \quad (3)$$

The adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|V|}$ of $G$ is split into $n$ smaller adjacency matrices $\boldsymbol{A}^{(i)} \in \mathbb{R}^{|V^{(i)}|}$. If $G$ is

[1] The repository will be added after the anonymity period.

a perfectly balanced tree, the number of parameters in $\theta_G$ thus decreases linearly from $O\left(|V|^2\right)$ to

$$O\left(n\left(\frac{|V|}{n}\right)^2\right) = O\left(\frac{|V|^2}{n}\right) \quad (4)$$

However, if $G$ is an unbalanced DAG with many overlapping branches, the number of parameters might in fact increase.

Instead of a single structure encoder $f_{\theta_G}$, we now train $n$ distinct models $f_{\theta_{G^{(i)}}} : \mathbb{R}^d \rightarrow [0,1]^{|V^{(i)}|}$, each with its own loss

$$L^{(i)} = \text{BCE}\left(\hat{\boldsymbol{y}}^{(i)}, \boldsymbol{y}^{(i)}\right) + \lambda_1 L^2 + \lambda_2 L^R \quad (5)$$

where BCE is the binary cross-entropy function commonly used for multi-label classification, $L^2$ is $L^2$ regularization to minimize model weights, $L^R$ is recursive regularization (Gopal and Yang, 2013) as defined in Section 2 to minimize the distance between the weights of adjacent nodes, and $\lambda_i$ are regularization hyperparameters.

We interpret $L^{(i)}$ as $n$ distinct tasks and minimize the final objective function $\mathcal{L} = \sum_{i=1}^{n} L^{(i)}$. The gradient of all $n$ hierarchy-aware classification layers propagates back to the shared text encoder, such that global information shared by different branches of the hierarchy is not lost. In a sense, the text encoder can thus be interpreted as the root node. The resulting model architecture is visualized in Figure 2.

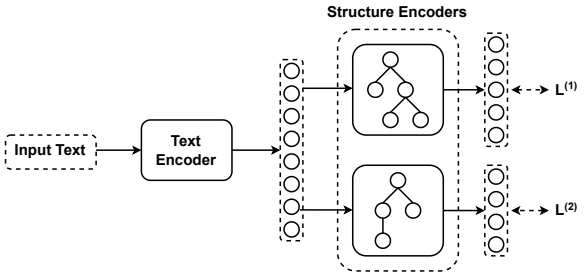


Figure 2: The proposed architecture for a hierarchy with two branches. A text encoder transforms a text into a vector which is then transformed by two separate structure encoders into label predictions. Two separate losses propagate back to the text encoder.

### 3.2 Inference

During inference, the $n$ model outputs $\hat{\boldsymbol{y}}_i$ need to be combined into a single prediction $\hat{\boldsymbol{y}}$. Two possible ways to do so are **masking** and **projecting**, which are shown in Figure 3.

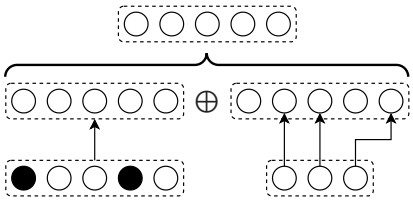

Figure 3: Undesired labels can be zeroed (left) or removed from the layer outputs and projected back to the original dimension during inference (right). The resulting vectors are combined into a single prediction using an element-wise sum or maximum.

For **masking**, each classifier $f_{\theta_{G^{(i)}}}$ produces a vector $\hat{\boldsymbol{y}}_i \in [0,1]^{|V|}$ instead of the desired dimension $|V^{(i)}|$. The gold labels $\boldsymbol{y}$ are multiplied element-wise with a mask $\mathbb{1}_{V^{(i)}}$ such that, during training, the corresponding classifier never sees a positive example of any label $v \in V_i$ it is not supposed to predict. We also apply this mask to $\hat{\boldsymbol{y}}$ to exclude these labels from the gradient computation entirely. However, the large number of practically unused parameters affects the regularization terms.

**Projection** is in line with the definitions in Section 3.1. The model stores $n$ mappings $\phi^{(i)} : \mathbb{N}^{|V^{(i)}|} \rightarrow \mathbb{N}^{|V|}$ to keep track of the corresponding label indices in the model outputs and in the ground truth. The model outputs $\hat{\boldsymbol{y}}_i \in [0,1]^{|V^{(i)}|}$ are then projected to a sparse vector of dimension $|V|$.

In both methods, these sparse predictions can be combined with an element-wise sum if the hierarchy is a tree, or with an element-wise maximum to account for overlapping branches in a DAG. Rather than the maximum, the arithmetic mean like in ensemble learning appears more intuitive but less straightforward to implement, since non-overlapping labels need to be ignored in the computation.

## 4 Experiments

We extend two state-of-the-art models, HiAGM (Zhou et al., 2020) and HiTIN (Zhu et al., 2023), with our multi-task learning architecture. We use the same framework that was originally developed for HiAGM and extended independently for HiMatch (Chen et al., 2021) and HiTIN. Each experiment uses BERT (Devlin et al., 2019) as the text encoder and runs once on a single Nvidia A40 with 48 GB GDDR6 memory and 32 GB RAM. We further use the same training parameters used by the authors of HiAGM and HITIN: Models are

| Model | WOS | | RCV1-v2 | | NYT | | Average | |
|---|---|---|---|---|---|---|---|---|
| | Micro-$F_H$ | Macro-$F_H$ | Micro-$F_H$ | Macro-$F_H$ | Micro-$F_H$ | Macro-$F_H$ | Micro-$F_H$ | Macro-$F_H$ |
| HiAGM (Zhou et al., 2020) | 87.52 | 81.56 | 86.81 | 68.74 | **79.64** | 69.24 | 84.66 | 73.18 |
| HiAGM+MTL (masking) | **87.61** | **81.87** | **87.13** | 69.82 | 79.40 | 69.06 | **84.71** | **73.58** |
| HiAGM+MTL (projection) | 86.89 | 80.90 | 86.93 | **70.22** | 79.42 | **69.59** | 84.41 | 73.57 |
| HiTIN (Zhu et al., 2023) | 87.39 | 81.73 | 88.68 | **73.00** | **79.75** | 69.21 | **85.27** | **74.65** |
| HiTIN+MTL (masking) | **87.83** | **81.99** | 88.38 | 72.35 | 79.59 | 69.02 | **85.27** | 74.45 |
| HiTIN+MTL (projection) | 87.39 | 81.49 | **88.71** | 72.78 | 79.66 | **69.67** | 85.25 | **74.65** |

Table 1: Micro- and macro-averaged hierarchical F-scores on three different datasets achieved by two baseline models and their HiMTL extensions. In each column, the best score in each category is marked in bold.

optimized with AdamW with a linearly decreasing initial learning rate of 2e-5 for 100 epochs, with early stopping after no improvement for 50 epochs. $L^2$ and recursive regularization are used with $\lambda_1 = 0.01$ and $\lambda_2 =$ 1e-6, respectively. HiAGM uses a node dimension $d = 300$ for its GCNs, whereas HiTIN uses coding trees of depth 2 and node dimension $d = 768$. The model outputs are combined using an element-wise maximum during inference.

These models are evaluated on three English hierarchical text classification datasets with label hierarchies of increasing complexity: Web of Science (Kowsari et al., 2017), Reuters Corpus Volume I-v2 (Lewis et al., 2004), and the New York Times Annotated Corpus (Sandhaus, 2008). Note that all three hierarchies are split into 4 branches at the root note, and since the WOS hierarchy has a depth of 2, each of its four predictors is essentially a flat classifier with an additional root label.

Micro and macro F-scores are reported for each experiment. In these metric computations, not only the most granular label is considered, but all predecessors. This variation of the metric is sometimes referred to as the hierarchical $F_1$-measure $F_H$ (Kosmopoulos et al., 2015) and it is less punishing if the model fails to predict the correct granular labels but correctly predicts the coarse labels. For clarity, we use the notation $F_H$. The results are reported in Table 1. Note that we are not able to reproduce the HiAGM and HiTIN results reported by Zhu et al. (2023) but usually achieve better scores with their provided codes and configurations.

HiMTL leads to small improvements especially when applying it to HiAGM. Here, both HiMTL implementations achieve better average Macro-$F_H$ scores while the masking approach achieves the best average Micro-$F_H$ score and works best on the WOS dataset with a simple hierarchy. The latter also applies to HiTIN, although the average scores do not indicate that it benefits from HiMTL overall. On NYT, the dataset with the most complex hierarchy, HiMTL always decreases the Micro-$F_H$ but the projection implementation increases the Macro-$F_H$. The training time increases by approximately 30% on all datasets.

## 5 Conclusion

We proposed an architectural extension for hierarchical text classification models by splitting label hierarchies into smaller, more localized sub-hierarchies, and training parallel hierarchy-aware classifiers in a multi-task learning setting. We further compared two different implementations to merge the model outputs into a single prediction: masking or projecting the output vectors. While this approach leads to improvements on a dataset with a simple label hierarchy and to a better macro average on a dataset with a complex hierarchy, its overall impact is small.

There are several other architectures and methods that could be evaluated in a multi-task learning setting, such as HiMatch (Chen et al., 2021), the post-processing of model outputs to remove hierarchical violations (Giunchiglia and Lukasiewicz, 2020), or structure encoders such as Gated Attention Networks (Veličković et al., 2018). Furthermore, there are several recent datasets for large-scale hierarchical text classification with complex label hierarchies such as the legal corpus EURLEX57K (Chalkidis et al., 2019) or its larger, multilingual variant MultiEURLEX (Chalkidis et al., 2021). However, the implementation was restricted by the framework we re-used, which was developed in the pre-Transformers era without easy extensibility in mind.

For the near future, we thus plan to re-implement this framework and some of the state-of-the-art architectures based on standardized libraries such as PyTorch Geometric (Fey and Lenssen, 2019) and NetworkX (Hagberg et al., 2008) to provide modern tools for easier modification, research, and deployment of hierarchical classification models.

## Limitations

For comparability with previous state-of-the-art methods, we used a framework that was originally published in 2020 without considering the latest developments such as Transformers. As a consequence, the extensibility of the code base was limited without making changes to the implementations of these SOTA models, whose reported results we were not able to reproduce. The same framework provides pre-processing codes for the three used datasets. We did not evaluate our implementation on other datasets used by previous SOTA methods, as it is not clear how these were pre-processed. We aim to solve this issue in future work with a more customizable framework and re-implementations of these methods based on standardized libraries.

Due to time constraints and limited computational resources, we were not yet able to run all experiments multiple times and thus do not report means, standard deviations, or the statistical significance of the results.

## Acknowledgments

Acknowledgments will be added after the anonymity period.

## References

Ilias Chalkidis, Emmanouil Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. Large-scale multi-label text classification on EU legislation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6314–6322, Florence, Italy. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, and Ion Androutsopoulos. 2021. MultiEURLEX - a multi-lingual and multi-label legal document classification dataset for zero-shot cross-lingual transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6974–6996, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Alex Cloud, Jacob Goldman-Wetzler, Evžen Wybitul, Joseph Miller, and Alexander Matt Turner. 2024. Gradient routing: Masking gradients to localize computation in neural networks. *Preprint*, arXiv:2410.04332.

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.

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

Eleonora Giunchiglia and Thomas Lukasiewicz. 2020. Coherent hierarchical multi-label classification networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9662–9673. Curran Associates, Inc.

Siddharth Gopal and Yiming Yang. 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, page 257–265, New York, NY, USA. Association for Computing Machinery.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Aris Kosmopoulos, Ioannis Partalas, Eric Gaussier, Georgios Paliouras, and Ion Androutsopoulos. 2015. Evaluation Measures for Hierarchical Classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery*, 29(3):820–865.

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.

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

Angsheng Li and Yicheng Pan. 2016. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*, 62(6):3290–3339.

Haoyang Liu, Xuegang Hu, Shengxing Bai, and Yaojin Lin. 2024. Hierarchical Multi-Granular Decision Networks for Hierarchical Classification.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Wenting Qi and Charalampos Chelmis. 2023. Hybrid Loss for Hierarchical Multi–label Classification Network. In *2023 IEEE International Conference on Big Data (BigData)*, pages 819–828.

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

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Carlos N Silla and Alex A Freitas. 2011. A survey of hierarchical classification across different application domains. *Data mining and knowledge discovery*, 22:31–72.

Fatos Torba, Christophe Gravier, Charlotte Laclau, Abderrhammen Kammoun, and Julien Subercaze. 2024. A study on hierarchical text classification as a seq2seq task. In *Advances in Information Retrieval*, pages 287–296, Cham. Springer Nature Switzerland.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*.

Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical multi-label classification networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5075–5084. PMLR.

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.

He Zhu, Chong Zhang, Junjie Huang, Junran Wu, and Ke Xu. 2023. HiTIN: Hierarchy-aware tree isomorphism network for hierarchical text classification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7809–7821, Toronto, Canada. Association for Computational Linguistics.