Well Begun is Half Done: An Implicitly Augmented Generative Framework with Distribution Modification for Hierarchical Text Classification

Huawen Feng, Jingsong Yan, Junlong Liu, Junhao Zheng, Qianli Ma[†]
School of Computer Science and Engineering,
South China University of Technology, China
541119578@qq.com, qianlima@scut.edu.cn

Abstract

Hierarchical Text Classification (HTC) is a challenging task which aims to extract the labels in a tree structure corresponding to a given text. Discriminative methods usually incorporate the hierarchical structure information into the encoding process, while generative methods decode the features according to it. However, the data distribution varies widely among different categories of samples, but current methods ignore the data imbalance, making the predictions biased and susceptible to error propagation. In this paper, we propose an **IM**plicitly **A**ugmented **G**enerativ**E** framework with distribution modification for hierarchical text classification (**IMAGE**). Specifically, we translate the distributions of original samples along various directions through implicit augmentation to get more diverse data. Furthermore, given the scarcity of the samples of tail classes, we adjust their distributions by transferring knowledge from other classes in label space. In this way, the generative framework learns a better beginning of the feature sequence without a prediction bias and avoids being misled by its wrong predictions for head classes. Experimental results show that **IMAGE** obtains competitive results compared with state-of-the-art methods and prove its superiority on unbalanced data.

Keywords: Hierarchical text classification, implicit augmentation, distribution modification

1. Introduction

Hierarchical Text Classification (HTC), a subtask of multi-label text classification, is an essential technique in scenarios requiring structural information like fine-grained entity typing (Xu and Barbosa, 2018), news classification (Lewis et al., 2004) and so on. In particular, the labels in HTC are hierarchical and usually represented as a tree from coarse-grained to fine-grained labels. However, since each label path starts at the root node and ends at intermediate or leaf nodes, the nodes closer to the root node appear much more frequently than those far away, making the label distribution extremely imbalanced.

This challenging task requires extracting the hierarchical structure information and overcoming the biases caused by imbalance. Early methods for HTC adopt Tree-LSTM (Zhou et al., 2020) as the encoder to get the text features level by level. Such an encoder is similar to the structure of the label tree, but directly using it to encode the text will prohibit the interaction between the semantics of text and labels. Specifically, Tree-LSTM just encodes text level by level, and then the classifier predicts the probabilities of different levels of labels. Consequently, the features of labels and the interaction between text and labels are neglected. Hence, connecting the hierarchical structure information and the text representation has become the main issue in the field.

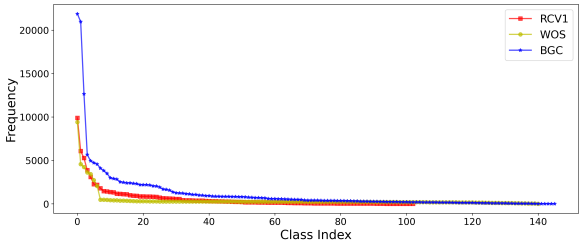


Figure 1: The diagram of the imbalance of HTC. Only a few labels (head classes) frequently appear, while others (medium and tail classes) rarely appear. The global frequency of labels shows a long-tailed distribution.

With the rapid development of Graph Convolutional Networks (Kipf and Welling, 2016; Defferrard et al., 2016) and Transformers (Vaswani et al., 2017), many methods have started to utilize various graph models to get the representation of the label hierarchy and made some progress. For instance, HiMatch (Chen et al., 2021) uses two GCNs to encode the label tree in a top-down approach as well as a bottom-up approach, respectively. HGCLR (Wang et al., 2022a) uses Graphormer, a special Transformer designed for graphs, to obtain the label hierarchy in the dataset. These discriminative methods successfully incorporate the hierarchical structure information into the text features to some

[†]Corresponding author

LREC-COLING 2024, pages 17433–17443

extent. Nevertheless, their dependence on auxiliary graph models requires more computational resources and increases time cost. In contrast, some methods do not rely on external tools any more. HBGL (Jiang et al., 2022), for example, replaces the graph modules with trainable label embeddings, which makes BERT work as a graph encoder. However, the label hierarchy here just acts as prior knowledge that facilitates the acquirement of sophisticated features. The enhanced representation is finally fed into classifiers and processed separately. Consequently, the final classification is still several mutually independent processes of matching text with each label.

Given that the auto-regressive model has a serialized structure, recent studies apply the generative pre-trained models to HTC. Compared to the traditional discriminative approaches, these generative ones will consider all the previous predictions while predicting a label sequence. For instance, Seq2Tree (Yu et al., 2022) proposes a generation framework with a constrained decoding strategy that effectively maintains the labels' consistency in prediction. And HPT (Wang et al., 2022b) utilizes a hierarchy-aware prompt tuning method and classifies layer by layer. Unfortunately, the prediction is usually biased because of imbalanced data. The labels closer to the root appear much more frequently but only account for a small proportion, and the global frequency of labels shows a long-tailed distribution (as shown in Figure 1). According to the previous work of long-tailed recognition (Li et al., 2022; Park et al., 2022; Alshammari et al., 2022), these head classes are always allocated with large weights, which distorts the label space. Actually, almost all the decoding strategies follow a fixed top-down order, hence there's a strong preference for the head classes from the beginning. What's worse, the phenomenon's impact will be amplified further due to error propagation in the sequence prediction. Once a wrong prediction for head classes occurs in the beginning, all the subsequent predictions will be influenced.

To sum up, most discriminative models are over-reliant on external graph modules, which consumes too much computational resources and time. Meanwhile, their classifiers work independently and overlook the connections between predictions. Additionally, there are some generative methods. They usually ignore the data imbalance so that their predictions are easily biased and susceptible to error propagation.

The problems mentioned above motivated us to propose an **IM**plicitly **A**ugmented **G**enerativ**E** framework with distribution modification for hierarchical text classification (**IMAGE**). Almost all existing discriminative methods directly divide HTC into several separate multi-label classification tasks according to the hierarchy. Given that, we adopt the generative framework to predict a label sequence. Moreover, in order to overcome the adverse effect of long-tailed label distribution, we introduce an implicit augmentation method. Specifically, the distributions of original samples are translated along various directions, which increases the data diversity. Considering that the statistics of tail classes are limited because of the scarcity, we utilize a distribution modification strategy to adjust their distributions. In addition, we employ random shuffling while training the model to prevent the serialized predictions from being dominated by biased head classes. As a result, no longer limited by the data imbalance and fixed predicting order, the model can get a better representation and predict balanced label sequences flexibly. In a word, we enhance the model's decoding process (predicting process) to have a better beginning of the feature sequence, eliminating the predicting bias and misleading preference for the head classes. Just as our title says, "**Well begun is half done.**"

In summary, the contributions of this paper are as follows:

- Through a detailed analysis of the existing methods, we point out the problems in previous frameworks of HTC.
- We propose an implicitly augmented generative framework with distribution modification. To the best of our knowledge, it is the first time to solve the problem of long-tailed label distribution for HTC.
- We conduct experiments on the three benchmark datasets. Compared with those strong baselines, the results demonstrate the effectiveness of implicit augmentation with distribution modification and random shuffling in improving sequence prediction.

2. Related Work

2.1. Long-tailed Recognition

The long-tailed data poses great challenges for classification tasks on how to deal with the class imbalance (Kang et al., 2019). The head classes only account for a small proportion of all classes, but the samples belonging to them make up a large proportion of the whole dataset. Such a lopsided distribution usually results in a prediction bias towards head classes, which affects the global classification.

Most existing methods are based on various rebalancing strategies, like re-weighting the loss (Lin et al., 2017), re-sampling the data (Chawla et al., 2002), adjusting the training process according to

confusion matrix (Kozerawski et al., 2020), transferring knowledge from head to tail classes (Cui et al., 2018), and so on. Unfortunately, these approaches are inapplicable to multi-label classification because they have to handle each class separately.

2.2. Implicit Data Augmentation

Data augmentation is a data-space solution to the problem of limited data (Shorten and Khoshgoftaar, 2019), which can be divided into explicit and implicit augmentation. Based on the original data and external tools, explicit augmentation generates real and observable data (hard samples) with strong interpretability. For instance, AAAS (Feng and Ma, 2022) generates metaphorical sentences by masking and predicting the words in original samples. In contrast, without the constraint of data authenticity, implicit augmentation (Wang et al., 2019) replaces the hard samples with virtual ones. Not being the regular images or text comprehensible to humans, these samples may be just a group of tensors semantically similar or different to the original samples.

Implicit data augmentation is more suitable for HTC because the correspondence between samples and labels in multi-label classification is too complex to generate real samples. Fortunately, rather than distinguishing which words determine the labels and which do not, implicit data augmentation adopts tensors with semantic information instead of observable data, making augmentation much easier. However, owing to the dependence on the data distribution statistics, the effectiveness of implicit augmentation will be sharply reduced for tail classes (Chen et al., 2022).

2.3. Distribution Modification

The data distribution is usually imbalanced in real-world scenarios, significantly influencing the models' performance. The current study focuses on how to correct the biased distribution. PBSA (Feng et al., 2023), for example, adopt a self-supervised method to generate the expected distribution and use Kullback-Leibler to optimize the data distribution. Nevertheless, generating supervision information requires a lot of extra computation. The time cost can be catastrophic when the data volume is enormous.

3. Methodology

3.1. Task Formulation

Given a document D consisting of n words $D = [x_1, x_2, ..., x_n]$, HTC aims to predict all the matching labels $\overline{Y}$ in the label tree $\mathbb{H} = (Y, E)$, where $Y = [y_1, y_2, ..., y_{N_c}]$ is the label set (the nodes of $\mathbb{H}$) and E is the edge set (the edges of $\mathbb{H}$). Specifically, $\mathbb{H}$ is constructed from coarse-grained to fine-grained labels and the target labels $\overline{Y}$ can be regarded as one or more label paths of the tree. Each starts at the root node and ends at the intermediate or leaf nodes.

3.2. Generative Backbone

As explained in Section 1, the generative model has a serialized structure predicting new labels based on all the previous predictions. Given that, we adopt the generative framework, and here we take BART (Lewis et al., 2019) as an example. BART's encoder first encodes the input document D to get a vector H^E:

$$H^E = Encoder([x_1, x_2, ..., x_n]) \tag{1}$$

The encoder's output H^E is then fed into BART's decoder with the previous predictions $Y_{<t} = [y_1, y_2, ..., y_m]$ to get $H^D = [h_1, h_2, ..., h_T]$, where h_t is the last hidden state of the current time step t and T is the length of each predicted sequence.

$$h_t = Decoder(H^E; Y_{<t}) \tag{2}$$

Further, the probability distribution g_t is calculated by:

$$\begin{aligned} z_t &= W h_t \\ g_t &= softmax(z_t) \\ W &= [w_1, w_2, ..., w_{N_c}] \end{aligned} \tag{3}$$

where $W \in \mathbb{R}^{N_c \times d}$ is the learnable parameters of a linear layer (classifier), and w_k is the weight of classification head for class k.

Finally, we can obtain the classification loss:

$$L_{cls} = -\frac{1}{Q}\frac{1}{T}\sum_{q=1}^{Q}\sum_{t=1}^{T}\log g_{(q,t)} \tag{4}$$

where Q is the number of samples and $g_{(q,t)}$ indicates the probability distribution of q-th sample at step t. Approximatively, the whole process can be regarded as a single-label classification task of $Q \times T$ samples.

3.3. Random Shuffling

The decoding strategies of traditional generative methods usually follow a fixed top-down order (e.g. BFS, DFS and so on). Nevertheless, due to the long-tailed data distribution, there's a strong preference for the head classes from the beginning. The labels in a higher hierarchy have a larger sample size, attracting more models' attention in prediction. Consequently, the model tends to predict a structurally matched label rather than a semantically matched one, especially in confusing cases.

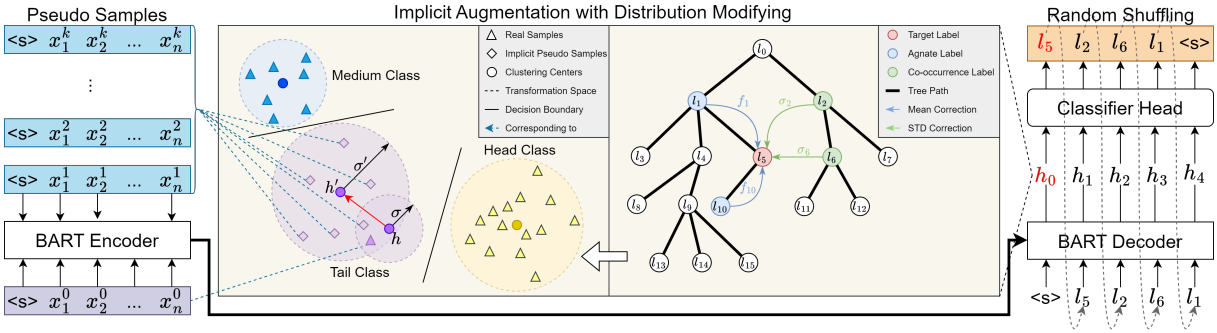


Figure 2: The diagram of our method. We propose implicit augmentation with distribution modification to enhance the generative model: (a) first, we get the feature sequence (decoder's last hidden states) of input sample. (b) Then we calculate their original distributions and modify them by agnate nodes and co-occurring labels. (c) Finally, we augment original features under the modified distributions (the number of augmented features is discussed in Section 3.6), which can be regarded as the process of sampling data points in the transformed feature space. Merely, these data points are pseudo samples not really existing.

Given the above predicting bias, we use random shuffling instead of common decoding strategies. The only difference between random shuffling and traditional decoding orders is that "random" initializes the label sequence order randomly before the start of the training, serving as a one-time operation. Specifically, the predicting order is no longer restricted, so the model is encouraged to predict any target label without following fixed orders. As shown in Figure 2, the target order should have been $[l_1, l_2, l_5, l_6]$ and $[l_1, l_5, l_2, l_6]$ under the strategy of BFS and DFS, respectively. However, after applying random shuffling, predicting any combination of the target labels is acceptable (e.g. $[l_5, l_2, l_6, l_1]$, $[l_2, l_5, l_1, l_6]$, $[l_6, l_1, l_2, l_5]$, $[l_1, l_6, l_5, l_2]$ and so on are all correct), which means we focus more on the completeness and accuracy of the output sequence than on its order. In this way, without the order constraint, the model can predict the most appropriate label of the current time step.

Random shuffling captures the co-occurrence information of target labels. For a text sample in HTC, we define the length of its corresponding labels as T, and there are $T!$ different permutations, which means that we have $T!$ different text-labels pairs for training, including but not limited to BFS and DFS pairs. Therefore, Random shuffling helps the model capture the hierarchical information between labels and the co-occurrence information between labels. The trick can be seen as data augmentation.

3.4. Implicit Augmentation

Instead of generating real and observable samples, implicit augmentation aims to produce new features loaded with semantic information. Given the feature $h_{(q,t)}$, the q-th sample's last hidden state at step t (the decoder's output which is then fed into the classifier head), we can randomly sample along $\mathcal{N}(0, \sigma_k)$ to generate new features with different semantic transformations. Given that, the augmented new feature obeys a Gaussian distribution which can be formulated as:

$$\begin{aligned} \widetilde{h}^{p}_{(q,t)} &\sim \mathcal{N}(h_{(q,t)}, \sigma_k) \\ k &= Y(q,t) \end{aligned} \tag{5}$$

where $\widetilde{h}^{p}_{(q,t)}$ is the p-th augmented feature whose class k (determined by q and t) is the same as the original feature $h_{(q,t)}$. σ_k is the covariance matrix of the class k, which is calculated based on k's feature centre $\boldsymbol{h_k}$:

$$\begin{aligned} \boldsymbol{h_k} &= \frac{1}{|S_k|} \sum_{(q,t)\in S_k} h_{(q,t)} \\ \sigma_k(a,b) &= \frac{1}{|S_k|-1} \sum_{(q,t)\in S_k} (h_{(q,t)}(a) - \boldsymbol{h_k}(a)) \\ &\quad (h_{(q,t)}(b) - \boldsymbol{h_k}(b)) \end{aligned} \tag{6}$$

where S^k is the set of features belonging to class k. $\sigma_k(a,b)$ indicates the value of position (a,b) in the covariance matrix of k and $h_{(q,t)}(a)$ is the value of a-th dimension of $h_{(q,t)}$. The meanings of $\boldsymbol{h_k}(a)$, $\boldsymbol{h_k}(b)$, and $h_{(q,t)}(b)$ are similar to $h_{(q,t)}(a)$. On account of the computation cost, the parameters of distributions in Equation 6 are calculated by online estimation.

Figuratively, the implicit augmentation can be regarded as locating pseudo sample points in the feature space formulated as $\mathcal{N}(h_{(q,t)}, \sigma_k)$. However, the statistical information on tail classes is limited due to the scarcity of samples. In other words, their sampling space is determined by quite a few data

points, which severely undermine the effectiveness of implicit augmentation on tail classes. Therefore, we propose a distribution modification approach to address this issue by adjusting their feature space based on other classes.

3.5. Distribution Modification

In traditional long-tailed recognition, most methods transfer the knowledge from similar classes to the target class based on the confusion matrix. Unfortunately, HTC is a multi-label classification task where the confusion matrix is inapplicable. Given that, we use the label hierarchy to modify $h_{(q,t)}$ and σ_k. Specifically, we use agnate (ancestors and descendants) and co-occurring labels to adjust them, respectively. Intuitively, the agnate labels share certain partial features according to the inheritance relationship, so we use them to adjust the expected value. As Figure 2 shows, for instance, the agnate labels of l_5 are l_1 and l_{10}. They have intra-class similarities because of the homology based on the tree structure. Noteworthily, the previous study (Chen et al., 2021) finds that the generative framework easily confuses the target label with its sibling nodes (e.g. the siblings of l_5 are l_3 and l_4), so these misleading nodes are not taken into consideration while modifying distributions. On the other hand, the part of their features similar to the target label has already been contained in their common ancestors. Based on that, we can get the modification of $h_{(q,t)}$:

$$\begin{aligned} \triangle h_{(q,t)} &= \sum_{i \in S_k^{agn}} w_{ik} \boldsymbol{h_i} \\ w_{ik} &= Norm(\frac{1}{\log |r_i - r_k| + 1}) \end{aligned} \tag{7}$$

where S_k^{agn} is the set of agnate labels of k and r_i is k's level and w_{ik} weights them by their relative distance in the tree.

As for the adjustment of σ, inspired by hypernetworks (Ha et al., 2016), we use the frequency of co-occurrence to supervise a set of parameters. The co-occurring labels capture the inter-class semantic directions of the training data. Specifically, the more frequently any two classes co-occur, the more consistent the sampling directions are. Mathematically, a positive correlation exists between their weights $\hat{w}_{ik}$ and the frequency of co-occurrence. The modification and loss function can be given by:

$$\begin{aligned} \triangle \sigma_k &= \sum_{i \in S_k^{co}} w_{ik} \sigma_k \\ L_\sigma &= \sum_{k=1}^{N_c} \frac{1}{|S_k^{co}|} \sum_{i \in S_k^{co}} (w_{ik} - \log X_{ik})^2 \end{aligned} \tag{8}$$

where S_k^{co} is the set of co-occurring labels of k and X_{ik} is the frequency of the co-occurrence of i and k.

Ultimately, the distribution of features is modified and defined as:

$$\widetilde{h}^p_{(q,t)} \sim \mathcal{N}(h_{(q,t)} + \alpha \triangle h_{(q,t)}, \beta(\sigma_k + \triangle \sigma_k)) \tag{9}$$

where α and β are decaying factors to control the degree of modification.

3.6. Upper Bound of the Loss Function

A naive method to implement the implicit augmentation is to explicitly augment each original feature $h_{(q,t)}$ for P times. Following Equation 9, we can generate P samples for each sample. The training loss is:

$$\begin{aligned} L_{aug} &= -\frac{1}{P}\frac{1}{Q}\frac{1}{T}\sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{t=1}^{T}\log g^p_{(q,t)} \\ &= -\frac{1}{P}\frac{1}{Q}\frac{1}{T}\sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{t=1}^{T}\log \frac{e^{w_k \widetilde{h}^p_{(q,t)}}}{\sum_{j=1}^{N_C} e^{w_j \widetilde{h}^p_{(q,t)}}} \end{aligned} \tag{10}$$

where $g^p_{(q,t)}$ is the probability distribution of p-th augmented feature for q-th sample at time step t and w_k is the weight for class k (the class of $g^p_{(q,t)}$ where $k = Y(q,t)$) in the whole weight matrix W of the classifier. P is the number of augmented features. As we mentioned in Section 2.2, certain directions in the feature space correspond to meaningful semantic transformations. However, searching for these semantic directions is difficult, especially for large-scale problems. So, we approximate the procedure by sampling random vectors in semantic space. Theoretically, the value of P (the augmentation times or the number of augmented features) should be as large as possible. It is similar to the times of flipping the coin in a coin-tossing simulation (As the number of times increases, the probability distribution gets closer to the theoretical value). Nevertheless, the time cost will be catastrophic when P is enormous (we can not generate so many augmented features and train the model based on them).

Considering that, we simplify the computation by deducing the upper bound of L_{aug} while P increases towards infinity. The original loss can be written as:

$$\begin{aligned} \underset{P\to\infty}{L_{aug}} &= \frac{1}{Q}\frac{1}{T}\sum_{q=1}^{Q}\sum_{t=1}^{T} G \\ G &= \mathbb{E}_{\widetilde{h}^p_{(q,t)}}[\log \sum_{j=1}^{N_C} e^U] \\ U &= \psi_k^j \widetilde{h}^p_{(q,t)}, \quad \psi_k^j = w_j - w_k \end{aligned} \tag{11}$$

The logarithm function is a convex function so that we can get the following inequality based on

Jensen's inequality: $\mathbb{E}[f(X)] \leq f(\mathbb{E}[X])$

$$G \leq \log \mathbb{E}_{\widetilde{h}^p_{(q,t)}}[\sum_{j=1}^{N_C} e^U] = \log \sum_{j=1}^{N_C} \mathbb{E}_{\widetilde{h}^p_{(q,t)}}[e^U] \quad (12)$$

According to Equation 9, we can infer that $\psi_k^j \widetilde{h}^p_{(q,t)}$ also obeys a Gaussian distribution which can be formulated as:

$$\begin{aligned} U \sim & \mathcal{N}(\psi_k^j(h_{(q,t)} + \alpha \triangle h_{(q,t)}), \Sigma_{jk}) \\ \Sigma_{jk} = & \beta \psi_k^j (\sigma_k + \triangle \sigma_k) {\psi_k^j}^T \end{aligned} \quad (13)$$

Then we can leverage Moment Generating Function of Gaussian distributions:

$$\mathrm{E}\left[e^{tX}\right] = e^{t\mu + \frac{1}{2}\sigma^2 t^2}, \quad X \sim \mathcal{N}\left(\mu, \sigma^2\right) \quad (14)$$

to further simplify G. We substitute Equation 14 with Equation 12 and obtain:

$$\log \sum_{j=1}^{N_C} \mathbb{E}_{\widetilde{h}^p_{(q,t)}}[e^U] = \log \sum_{j=1}^{N_C} e^V \quad (15)$$

where $V = \psi_k^j(h_{(q,t)} + \alpha \triangle h_{(q,t)}) + \frac{1}{2}\beta \psi_k^j (\sigma_k + \triangle \sigma_k) {\psi_k^j}^T$. Then the upper bound is:

$$\overline{L_{aug}} = \frac{1}{Q}\frac{1}{T}\sum_{q=1}^{Q}\sum_{t=1}^{T} \log \sum_{j=1}^{N_C} e^V = -\frac{1}{Q}\frac{1}{T}\sum_{q=1}^{Q}\sum_{t=1}^{T} \log(\frac{e^{w_k(h_{(q,t)} + \alpha \triangle h_{(q,t)})}}{\sum_{j=1}^{N_C} e^{w_j(h_{(q,t)} + \alpha \triangle h_{(q,t)}) + \frac{1}{2}\beta \psi_k^j (\sigma_k + \triangle \sigma_k) {\psi_k^j}^T}}) \quad (16)$$

which is the final loss to train the **IM**plicitly **A**ugmented **G**enerativ**E** framework with distribution modification (**IMAGE**). The whole training process is listed in Algorithm 1. We first use L_{cls} to get a trained generative backbone (feature network and classifier). Then we use $\overline{L_{aug}}$ and L_σ to train it to solve the predicting bias caused by long-tailed distribution.

Algorithm 1: Training Process

Input: train set D_{train}, validation set D_{val}, pre-trained generative model $\boldsymbol{M_0}$, pre-training epochs E_1, IMAGE-training epochs E_2.
for $i = 1, , 2..., E_1$ **do**
  Train the model $\boldsymbol{M_0}$ on D_{train} by L_{cls}.
  Update the parameters of $\boldsymbol{M_0}$.
  Get F1-score on D_{val}.
end
Get the best model $\boldsymbol{M_1}$ for highest F1-score.
for $i = 1, , 2..., E_2$ **do**
  Train the model $\boldsymbol{M_1}$ on D_{train} by $\overline{L_{aug}}$ and L_σ.
  Update θ.
  Get F1-score on D_{val}.
end
Get the best model $\boldsymbol{M_2}$ for highest F1-score.
return $\boldsymbol{M_2}$

4. Experiments

We conduct extensive experiments to verify the effectiveness of our proposed model IMAGE and analyze it with ablation studies and visualization results. In this section, we attempt to answer the following questions: **RQ1:** Does IMAGE perform better than existing methods? **RQ2:** Are the implicit augmentation and distribution modification the key factors affecting the performance? **RQ3:** Is "Well Begun is Half Done" right?

4.1. Datasets

To evaluate the effectiveness of our model, we conduct experiments on three widely used datasets for HTC, including Web Of Science (WOS) (Kowsari et al., 2017), RCV1-V2 (Lewis et al., 2004), and BlurbGenreCollection(BGC). The detailed dataset statistics are listed in Table 1, where Depth is the max depth of the label tree, and Avg(T) indicates the average number of labels for each sample. To make a fair comparison, we use the same train, validation, and test splits as those in previous work.

Dataset	N_c	Depth	Avg(T)	Train	Validation	Test
WOS	141	2	2	30,070	7,518	9,397
RCV1-V2	103	4	3.24	20,833	2,316	781,265
BGC	146	4	3.01	58,715	14,785	18,394

Table 1: Detailed dataset statistics.

4.2. Experimental settings

We adopt BART-Base, a typical generative model, as the backbone of our experiments. The maximum length of the WOS, RCV1-V2, and BGC output sequences are set to 6, 20, and 16, respectively. And each sequence starts from a token "<s>", ends at a token "</s>" and then is padded by several tokens "<pad>". The number of total epochs is set to 150, where E_1 is set to 30 and E_2 is set to 120. We set the batch size to 10 and optimized the model with AdamW with a learning rate of 2e-5. During the training stage, we train the model with the train set and evaluate it with the validation set at each epoch. As for the validation and inference stage, we use greedy search to predict label sequences. All of our experiments are conducted on Tesla P40.

We compare our models with current strong baselines, including: **TextRNN** (Liu et al., 2016): Use the multi-task learning framework based on RNN to share information while modelling the text.

Model	WOS		RCV1-V2		BGC	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
TextRNN	77.94	69.65	-	-	-	-
TextCNN	82.00	76.18	76.60	43.00	-	-
TextRCNN	83.55	76.99	81.57	59.25	-	-
HiAGM	85.82	80.28	83.96	63.35	77.22	57.91
BERT+HiAGM	86.04	80.19	85.58	67.93	-	-
HTCInfoMax	85.58	80.05	83.51	62.71	-	-
BERT+HTCInfoMax	86.30	79.97	85.53	67.09	-	-
HiMatch	86.20	80.53	84.73	64.11	76.57	58.34
BERT+HiMatch	86.70	81.06	86.33	68.66	78.89	63.19
HGCLR	87.11	81.20	86.49	68.31	-	-
HPT	87.16	81.93	87.26	69.53	-	-
SGM-T5	85.83	80.79	84.39	65.09	77.84	60.91
Seq2Tree	87.20	82.50	87.20	70.01	79.72	63.96
PAAM-HiA-T5	90.36	81.64	87.22	70.02	-	-
IMAGE	87.76	82.38	87.69	70.77	81.16	67.37

Table 2: Experimental results of on three benchmarks. The best result is in red, the second is in green, and the third is in blue. The table is divided into three parts, and from top to bottom they are: discriminative methods, generative methods, and our method.

TextCNN (Chen, 2015): Introduce Parallel CNN and Deep CNN to capture local semantic features. **TextRCNN**: Apply a recurrent structure to capture contextual information so that less noise will be introduced compared to traditional window-based neural networks. **HiAGM** (Zhou et al., 2020): Formulate the hierarchy as a directed graph and utilize hierarchy-aware encoders (Tree-LSTM) to model label dependencies. **HiMatch** (Chen et al., 2021): Regard HTC as a semantic matching problem and model the matching relationship from coarse-grained labels to fine-grained ones. **HGCLR** (Wang et al., 2022a): Apply Graphormer, a graph encoder, to embed the hierarchy into the text feature instead of modelling them separately. **HPT** (Wang et al., 2022b): Utilize prompt tuning methods to handle HTC from a multi-label MLM view. **SGM-T5** (Yang et al., 2018): View HTC as a sequence generation problem and use T5, a generative model, as the backbone. **Seq2Tree** (Yu et al., 2022): Propose a constrained decoding strategy to tackle label inconsistency problems based on the generative framework. **PAAM-HiA-T5** (Huang et al., 2022): Use path-adaptive attention mechanism to lead the model to adaptively focus on the path where the currently generated label is located, shielding the noise from other paths.

Among them, **TextRNN**, **TextCNN**, **TextRCNN**, **HiAGM**, **HiMatch**, **HGCLR**, and **HPT** are discriminative methods, while **SGM-T5**, **Seq2Tree**, and **PAAM-HiA-T5** are based on genrative frameworks.

4.3. RQ1: Does IMAGE perform better than existing methods?

Table 2 shows the experimental results of IMAGE compared with other baselines on three datasets, and the overall results indicate its effectiveness. We can find that the performance of IMAGE is good on all datasets and it almost surpasses all the existing methods, especially on the more complex datasets - RCV1-V2 and BGC. Nevertheless, on the simpler task - WOS, Micro-F1 and Macro-F1 of IMAGE may not be the best but are still quite competitive compared with state-of-the-art methods.

Notably, the performance of generative models is generally better than discriminative methods. To prove that the improvement does not come from the larger size, we list the parameters of some typical backbones in Table 3, including BERT+GCN, BERT+Graphormer, T5 and BART. Our method, by contrast, achieves a good performance with the least parameters.

Model	Size
BERT+GCN	148M
BERT+Graphormer	156M
T5	220M
BART	140M

Table 3: The size of different backbones.

4.4. RQ2: Are the implicit augmentation and distribution modification the key factors affecting the performance?

The results of the ablation study are shown in Table 4. Compared with vanilla BART, our method

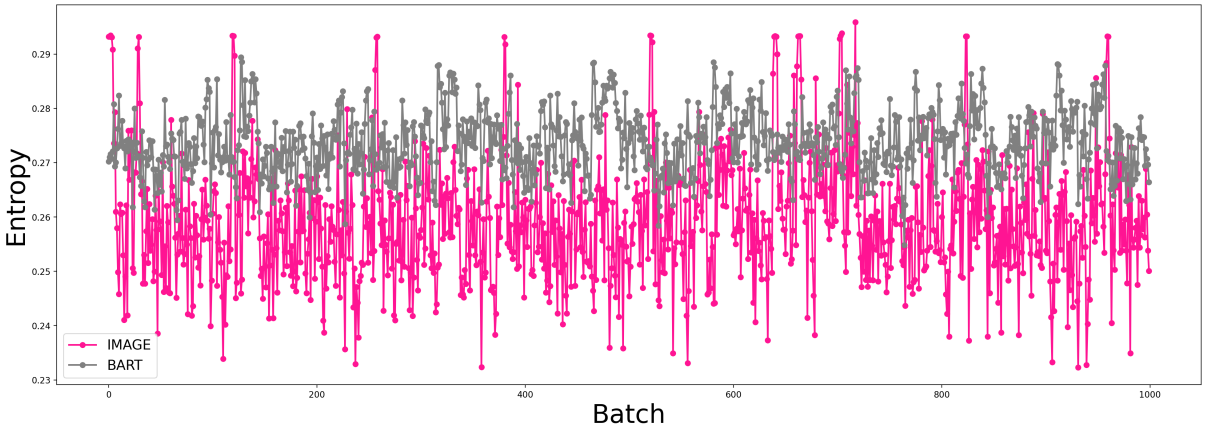


Figure 3: The average entropy of the beginning's output score of 1,000 mini-batches sampled randomly.

significantly improves the performance of generative models. The results drop when removing the Distribution Modification, which proves its effectiveness and necessity. Meanwhile, BART without the whole Implicit Augmentation module obtains a similar result. Therefore, we conclude that adjusting distribution is the critical factor for implicit augmentation on long-tailed data, and eliminating it will affect the augmentation effect. Moreover, we can find that Random Shuffling also plays an essential role in the training. It enforces the model to make the most proper predictions without the limitation of fixed orders, which helps the decoder focus more on the output's accuracy and completeness to select a semantically matched label rather than a structurally matched one, making the predicting process immune to the bias caused by the strong preference for the head classes. Overall, the modules and tricks mentioned above work together and alleviate the adverse impacts of data imbalance and fixed orders.

Model	Micro-F1	Macro-F1
IMAGE	87.69	70.77
-w/o Distribution Modification	87.43	69.26
-w/o Implicit Augmentation	87.06	69.22
-w/o Random Shuffling	86.67	69.35
BART (Backbone)	86.51	68.15

Table 4: The results of the ablation study on the benchmark dataset - RCV1-V2.

4.5. RQ3: Is "Well Begun is Half Done" right?

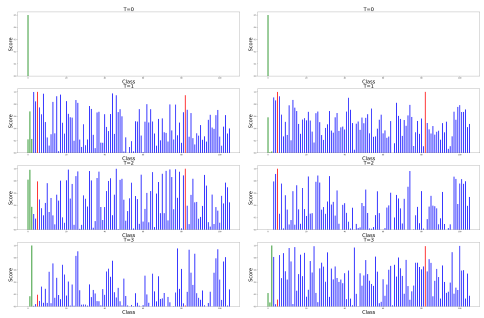
We observe the final output score and plot histograms to verify how IMAGE achieves the anticipation. We make a comparison with the typical generative framework - BART, which uses the traditional decoding strategy in previous work. As shown in Figure 4, the predictions of the traditional generative method are strongly associated with the time step. Due to the prediction bias, it tends to predict a wrong head class rather than a correct tail class at the beginning. What is worse, the error will further affect the following predictions, which results in the deviation of the whole output sequence from the target. However, through Random Shuffling and Implicit Augmentation with Distribution Modification, IMAGE has a much better feature sequence. Each target class is allocated with a high score, even if it is a tail class. In other words, the model can predict a semantically matched label rather than a structurally matched one. In this way, the model learns a better beginning of the feature sequence without a prediction bias and avoids being misled by its wrong predictions for head classes. Therefore, we can say "**Well begun is half done.**"

Figure 3 proves its correctness from a global view. We randomly sample 1,000 mini-batches and calculate the average entropy of their beginning output score. It is obvious that IMAGE's output has less average entropy. We speculate that IMAGE focuses more on the target labels, so the output distribution is relatively concentrated. On the contrary, the traditional generative framework has a strong preference for all head classes, hence the relatively dispersed output distribution.

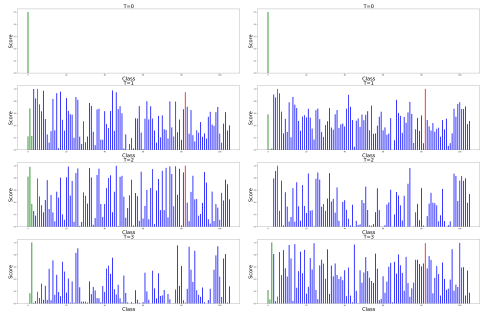
5. Conclusion

This paper summarizes existing HTC methods, indicating that almost all of them ignore biased predictions caused by the long-tailed distribution. We propose IMAGE, an implicitly augmented generative framework, and conduct massive experiments on the three benchmark datasets. The remarkable performance demonstrates the effectiveness of our method for getting better feature sequences, especially their beginnings. We expect our work will

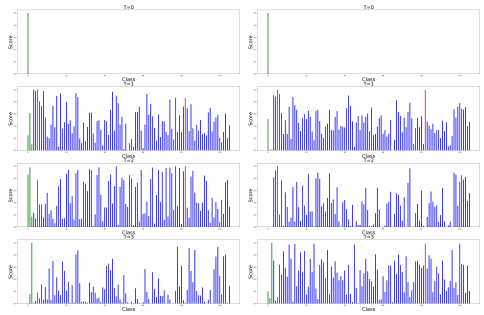
direct more scholarly attention to the representation learning of generative frameworks.



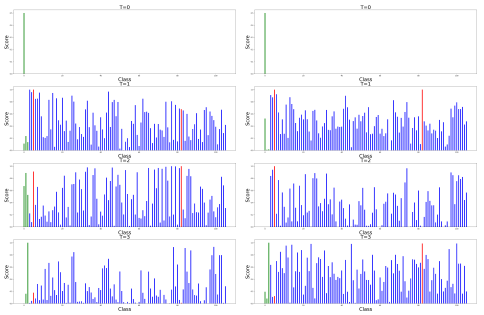
(a) BART-1 (b) IMAGE-1



(c) BART-2 (d) IMAGE-2



(e) BART-3 (f) IMAGE-3



(g) BART-4 (h) IMAGE-4

Figure 4: The output of BART (the left graph) compared with IMAGE (the right graph). Class 0, 1 and 2 (green bars) corresponds to <s>, <pad>, and </s> which are ignorable in the global predictions.

6. Bibliographical References

Shaden Alshammari, Yu-Xiong Wang, Deva Ramanan, and Shu Kong. 2022. Long-tailed recognition via weight balancing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6897–6907.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16:321–357.

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.

Xiaohua Chen, Yucan Zhou, Dayan Wu, Wanqian Zhang, Yu Zhou, Bo Li, and Weiping Wang. 2022. Imagine by reasoning: A reasoning-based implicit semantic data augmentation for long-tailed classification. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pages 356–364.

Yahui Chen. 2015. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo.

Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. 2018. Large scale fine-grained categorization and domain-specific transfer learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4109–4118.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29.

Huawen Feng, Zhenxi Lin, and Qianli Ma. 2023. Perturbation-based self-supervised attention for attention bias in text classification.

Huawen Feng and Qianli Ma. 2022. It's better to teach fishing than giving a fish: An auto-augmented structure-aware generative model for metaphor detection. In Findings of the Association for Computational Linguistics: EMNLP 2022, pages 656–667, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. arXiv preprint arXiv:1609.09106.

Wei Huang, Chen Liu, Bo Xiao, Yihua Zhao, Zhaoming Pan, Zhimin Zhang, Xinyun Yang, and Guiquan Liu. 2022. Exploring label hierarchy in a generative way for hierarchical text classification. In Proceedings of the 29th International Conference on Computational Linguistics, pages 1116–1127.

Ting Jiang, Deqing Wang, Leilei Sun, Zhongzhi Chen, Fuzhen Zhuang, and Qinghong Yang. 2022. Exploiting global and local hierarchies for hierarchical text classification. arXiv preprint arXiv:2205.02613.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. 2019. Decoupling representation and classifier for long-tailed recognition. arXiv preprint arXiv:1910.09217.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

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.

Jedrzej Kozerawski, Victor Fragoso, Nikolaos Karianakis, Gaurav Mittal, Matthew Turk, and Mei Chen. 2020. Blt: Balancing long-tailed datasets with adversarially-perturbed images. In Proceedings of the Asian Conference on Computer Vision.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

Tianhong Li, Peng Cao, Yuan Yuan, Lijie Fan, Yuzhe Yang, Rogerio S Feris, Piotr Indyk, and Dina Katabi. 2022. Targeted supervised contrastive learning for long-tailed recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6918–6928.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision, pages 2980–2988.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101.

Seulki Park, Youngkyu Hong, Byeongho Heo, Sangdoo Yun, and Jin Young Choi. 2022. The majority can help the minority: Context-rich minority oversampling for long-tailed classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6887–6896.

Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. Journal of big data, 6(1):1–48.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems, 30.

Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Gao Huang, and Cheng Wu. 2019. Implicit semantic data augmentation for deep networks. Advances in Neural Information Processing Systems, 32.

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. arXiv preprint arXiv:2203.03825.

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.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. arXiv preprint arXiv:1803.03378.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. Sgm: sequence generation model for multi-label classification. arXiv preprint arXiv:1806.04822.

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.

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.