Enhancing Text-Based Hierarchical Multilabel Classification for Mobile Applications via Contrastive Learning

Jiawei Guo
Tencent
Shenzhen, Guangdong, China
javedguo@tencent.com

Yang Xiao
Xidian University
Xi'an, Shaanxi, China
yxiao@xidian.edu.cn

Weipeng Huang*
Shenzhen Institute of Information Technology
Shenzhen, Guangdong, China
weipenghuang@sziit.edu.cn

Guangyuan Piao
Independent Researcher
Dublin, County Dublin, Ireland
parklize@gmail.com

Abstract

A hierarchical labeling system for mobile applications (apps) benefits a wide range of downstream businesses that integrate the labeling with their proprietary user data, to improve user modeling. Such a label hierarchy can define more granular labels that capture detailed app features beyond the limitations of traditional broad app categories. In this paper, we address the problem of hierarchical multilabel classification for apps by using their textual information such as names and descriptions. We present: 1) HMCN (Hierarchical Multilabel Classification Network) for handling the classification from two perspectives: the first focuses on a multilabel classification without hierarchical constraints, while the second predicts labels sequentially at each hierarchical level considering such constraints; 2) HMCL (Hierarchical Multilabel Contrastive Learning), a scheme that is capable of learning more distinguishable app representations to enhance the performance of HMCN. Empirical results on our Tencent App Store dataset and two public datasets demonstrate that our approach performs well compared with state-of-the-art methods. The approach has been deployed at Tencent and the multilabel classification outputs for apps have helped a downstream task—credit risk management of users—improve its performance by 10.70% with regard to the Kolmogorov-Smirnov metric, for over one year.

CCS Concepts

• **Computing methodologies → Neural networks**; **Supervised learning by classification**.

Keywords

Hierarchical Multilabel Classification, Contrastive Learning, App Classification

*Corresponding Author.

This work is licensed under a Creative Commons Attribution 4.0 International License.
KDD '25, Toronto, ON, Canada
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737216

ACM Reference Format:
Jiawei Guo, Yang Xiao, Weipeng Huang, and Guangyuan Piao. 2025. Enhancing Text-Based Hierarchical Multilabel Classification for Mobile Applications via Contrastive Learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3711896.3737216

1 Introduction

Tencent App Store[1] is one of the most widely used app stores in China. It has 200 million monthly active users with over 30,000 applications (apps). These apps are organized in a hierarchical structure that can be considered a topic taxonomy. Behind the scenes, apps are categorized into one or more topics or labels in the taxonomy, which consists of three levels. The top level has broad topics such as "Finance", "Video" or "Game" and the lower levels have granular topics as shown in Fig. 1. Organizing items into such a taxonomy is a common practice in industry, and it can be used in many downstream tasks such as providing personalized recommendations for users based on interacted items and associated topics or creating Ad campaigns[2] [3, 11, 17, 38]. Such a hierarchical structure enables downstream business applications to capture more detailed and comprehensive aspects of apps. As an example, businesses that run credit risk management can apply the classification results to strengthen their user profiling and modeling, thereby improving their risk management models.

The task of classifying apps into correct topics in the taxonomy can be treated as a hierarchical multilabel classification problem. In our case, an app consists of three types of textual information: 1) the app name, 2) the app description, and 3) the editorial comments summarizing the key functionalities of the app. Therefore, encoding these textual features into a vector representation or embedding of the app is an important step, which is then followed by building a classification model based on those app embeddings.

Intuitively, we expect the similarity score of app embeddings with similar topics to be higher than that of unrelated ones in the embedding space. This may be observed in the right heatmap in Fig. 2. For example, the similarity between two online video apps (`IQiYi` and `Tencent Video`) is much higher than the similarity between unrelated apps like `Taobao` (a shopping app) and `Tencent`

[1] https://sj.qq.com/
[2] https://developers.google.com/privacy-sandbox/private-advertising/topics

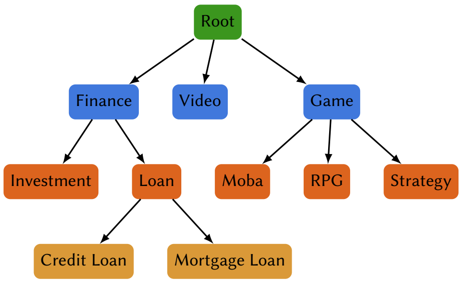

Figure 1: An example of label hierarchy. Moba: multiplayer online battle arena games; RPG: role-playing games.

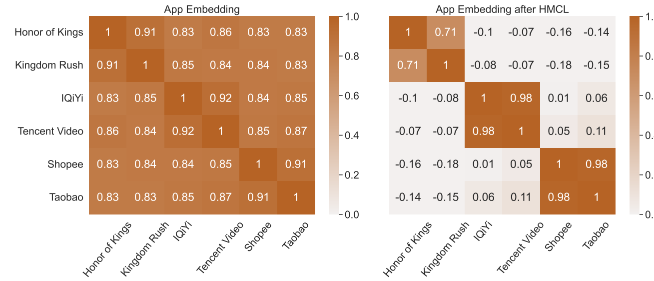

Figure 2: Example of app embedding (cosine) similarities after HMCL (Hierarchical Multilabel Contrastive Learning)[3].

Video. We can use a straightforward approach such as a pretrained BERT [9] or one of its variants such as RoFormer [30] to derive the corresponding app embeddings based on the concatenated text of these three types of textual information. However, as shown in the left heatmap in Fig. 2, the similarity scores between app embeddings are not distinguishable using such a straightforward approach, which in turn results in non-optimal classification performance.

Based on the above observations, we propose a Hierarchical Multilabel Classification Network (HMCN) with a pretrained encoder using a Hierarchical Multilabel Contrastive Learning (HMCL) scheme, for classifying apps in the Tencent App Store. More specifically, our contributions are highlighted as follows:

(1) We propose a Hierarchical Multilabel Contrastive Learning (HMCL) scheme in Section 4, which results in better app embeddings as shown in the right side of Fig. 2.
(2) We present the HMCN (Section 3.3), a hierarchical multilabel classification network incorporating two classification angles: a global one treating all labels in the hierarchy equally, and a local one maintaining different embeddings for predicting the label assignments at different levels. The experiments in Section 5 on our Tencent App Store dataset as well as on two public datasets show that our approach provides the best classification performance on the app dataset, and competitive performance on the public ones.
(3) The HMCN together with HMCL has been deployed at Tencent for over one year and the classification results have been used for assessing credit risks of users. Compared to the previously deployed approach, incorporating app labels from our approach shows 10.70% improvement on the key evaluation metric, Kolmogorov-Smirnov (KS) value, of the downstream task (Section 5.5).

In fact, our solution is more suited to small business teams since it is far less demanding of the computational resources and deployment resources compared to, e.g., large language models [24, 41].

2 Related Work

Hierarchical multilabel classification methods can be broadly categorized into global and local approaches [29]. The global approaches degenerate the hierarchical multilabel classification task into a multilabel classification task [12], which are simple to implement but are often prone to underfitting. These methods mostly center around developing encoders or decoders that encapsulate the hierarchical constraints [23, 26, 37, 42]. In contrast, local methods pass the information from parent to children, and hence predict labels for each level of the hierarchy from top to bottom [2, 4, 16, 19, 28]. The HMCN [35, 36] has shown its effectiveness by combining both global and local approaches. Our work falls into this category and adapts the HMCN to accommodate the multi-field, text-based description of apps. Motivated by suboptimal app embeddings obtained via a pretrained BERT (or its variant), as illustrated in Fig. 2, we introduce hierarchical multilabel contrastive learning for pretraining the text encoder for the HMCN.

Contrastive learning (CL) [6, 14, 27] is a methodology for learning a representation space that allows similar data to get closer together while the pushing dissimilar data further apart. This approach enhances a model's ability to distinguish between relevant and irrelevant features, ultimately improving its performance on downstream or end-to-end tasks such as classification and clustering. However, the classical CL has focused primarily on unsupervised learning and multiclass classification [5, 13, 15, 18, 25]. For multilabel classification in recent studies [32, 39], the supervised loss function is modified by introducing weights, derived from the similarity of the label vectors of samples, to adjust the loss for sample pairs. Wang et al. [34] develop a hierarchy encoder and a text encoder to respectively encode hierarchical labels and input text. It generates a positive peer for a data item by removing the unimportant words from the data item itself. Zhu et al. [43] also employ a structural text encoder to encode hierarchical labels and input text. The work differs from [34] in that it projects the label and text vectors into the same space; hence, they define the positive sample pairs as the corresponding label and text vectors. Although Zhang et al. [40] adopt the term "hierarchical multilabel contrastive learning", their approach actually deals with the hierarchies where only one label is assigned to a data point under an active parent label. In contrast, our case assumes that a data point can be assigned multiple active labels under one active parent label.

Our proposal follows the conventional technical path for CL to focus on the relationships between data samples, rather than the relationships between labels and samples [34, 43], taking the hierarchical information into account. Also, our proposed HMCL occurs during the pretraining phase, prior to training HMCN. It is

[3] Honor of Kings and Kingdom Rush are mobile games but with significantly different play styles; IQiYi and Tencent Video are two online video apps; Shopee and Taobao are two online shopping apps.

decoupled from the classification model training and can be used with other classification models.

3 Model and Implementations

We begin by introducing the necessary notation. Next, we present our data encoding strategy, followed by a detailed discussion of the HMCN architecture and HMCL. The overall architecture is illustrated in Fig. 3.

3.1 Notation

Using Fig. 1 as a motivating example, we first define a hierarchy H using the notation of directed graph such that $H = (V, E)$, where V is the label set and E denotes the set of parent-to-child relationships. Precisely, for a set of m ordered labels, $V = \{v_1, \dots, v_m\}$ and $E = \{(u, v) : u = parent(v), \forall u, v \in V\}$. This clarifies the top-down paths. In our task, one parent label can have multiple children while one child label can have only one parent. We denote the dataset by $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ where $\mathbf{X} = \{x_1, \dots, x_n\}$ represents the features and $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ are the assigned labels to the observations. Considering any index i, $\mathbf{y}_i \in \{0, 1\}^m$ where 1 indicates that the corresponding label has been assigned to x_i while 0 refers to the opposite. Moreover, we denote the assignment of label v for x_i by y_{iv}. We emphasize that there could possibly be multiple 1s or all 0s in any given $\mathbf{y}$. A label v for any x cannot be 1 once its parent label u for this x is 0. Finally, let us write that x_i has an active label v if $y_{iv} = 1$, while x_i has an inactive label v if $y_{iv} = 0$.

3.2 Data Encoding

In our scenario, apps are represented by the text data from the App Store. We utilize three fields to construct the app embedding: app name, description, and editorial comments. It follows that we could feed all the information into a text encoder, where we apply RoFormer, to transform the information to embeddings. However, concatenating all fields could lead to prohibitively long input text that the encoder has to truncate, resulting in potential information loss. Also, certain fields might be empty due to practical reasons, and their importance should be down-weighted.

To address the above issue, we set a special token to each field with a similar usage of `[CLS]` in the BERT models [9]. This token will be inserted in front of each field and the fields will be independent inputs passed to the encoder. We extract the embedding for the corresponding token from the three fields and merge them into a 2D embedding. In particular, we set the tokens `[N]` for app name, `[D]` for description, and `[C]` for editorial comments, respectively. We input the app name as a sentence x_N into RoFormer and obtain the 2D embeddings $H_N \in \mathbb{R}^{s \times d}$ where s is the length of sequence (without pooling). As the special token `[N]` is engineered to always stay in position 0, we can extract an 1D embedding $\mathbf{h}_N \in \mathbb{R}^d$ by fetching the first row from H_N, such that $\mathbf{h}_N = H_N[0] = \text{RoFormer}(x_N)[0]$. Similarly, we achieve the embedding $\mathbf{h}_N, \mathbf{h}_D, \mathbf{h}_C$ for app name, description, and editorial comments respectively by

$$\mathbf{h}_N = H_N[0]; \quad \mathbf{h}_D = H_D[0]; \quad \mathbf{h}_C = H_C[0]\,. \tag{1}$$

We further obtain the embedding $\mathbf{h}^*$ for x by

$$\mathbf{h}^* = \text{CONCAT}([\mathbf{h}_N, \mathbf{h}_D, \mathbf{h}_C];\ dim = 0) \in \mathbb{R}^{3 \times d} \tag{2}$$

where $\text{CONCAT}(\cdot)$ concatenates the inputs along the x-axis when $dim = 0$; otherwise, it concatenates them along the y-axis when $dim = 1$. This formula can then be extended to cases with additional fields. To focus on the most important features, we apply a self-attention to acquire the embedding at the root level, $\mathbf{h}_0$, by

$$\mathbf{h}_0 = \text{Encoder}(x) = \text{MultiHeadAttn}(\mathbf{h}^*, \mathbf{h}^*, \mathbf{h}^*) \tag{3}$$

where $\text{MultiHeadAttn}(\cdot, \cdot, \cdot)$ refers to the multihead attention [31]. The three arguments correspond to the *query*, *key*, and *value* in the attention mechanism. When all three arguments are the same, it becomes a self-attention computation.

3.3 Hierarchical Multilabel Classification Network

We follow the well-known work [35] to split our classification model into two classification manners, namely the global and local manners. In the global manner, the constraints of hierarchical multilabel classification are ignored, and all labels are treated equally as in a simple multilabel classification. The local manner maintains different embeddings for predicting the label assignments at different levels of the hierarchy. More importantly, the information of the embedding at a higher level (closer to the root node) will be passed on to the embedding at the current level. This design ensures that the information of the hierarchy can be utilized. Finally, these two types of predictions are merged to produce the final predictions.

To alleviate the violation of label assignment (i.e., a data point with label v is assigned a value of 1 while its parent u for that data is assigned 0), a path regularization term is added to the loss function.

3.3.1 Local Manner. Let $\mathbf{z}$ represent the likelihoods of $\mathbf{y}$ where a single element $z_v = p(y_v = 1|x)$. In this manner, the model outputs the estimated likelihoods of the label assignment $\{\hat{\mathbf{z}}^{(1)}, \dots, \hat{\mathbf{z}}^{(L)}\}$, level by level, where $\hat{\mathbf{z}}^{(\ell)}$ denotes the likelihoods of the predictions at level ℓ. Hence, in this local manner, we generate the likelihood estimates $\hat{\mathbf{z}}_{local}$ for x by concatenating the local likelihoods:

$$\hat{\mathbf{z}}_{local} = \text{CONCAT}([\hat{\mathbf{z}}^{(1)}, \dots, \hat{\mathbf{z}}^{(L)}];\ dim = 1)\,. \tag{4}$$

We now describe generation of local predictions at non-root levels. For any non-root level ℓ, the embedding $\mathbf{h}_{\ell-1}$ from the last level is gathered to construct the embedding $\mathbf{h}_\ell$ along with the meta embedding. We thus achieve the local embedding by

$$\mathbf{h}_\ell = \begin{cases} \text{MLP}(\mathbf{h}_0) & \ell = 1 \\ \text{MultiHeadAttn}\,(\mathbf{h}_0, \mathbf{h}_{\ell-1}, \mathbf{h}_{\ell-1}) & \ell = 2, \dots, L \end{cases}. \tag{5}$$

The MLP at the first level can be regarded as a learnable "prior" for the encoded embedding $\mathbf{h}_0$ and can be thought of as certain information from the root. For other levels, we acquire the local embedding through cross-attention at the subsequent levels. We replace the concatenation method used in literature [35] to merge the embeddings by cross-attentions. This approach can identify significant features within local embeddings in relation to their parent embeddings, during the information transfer. In effect, we found that the empirical performance of using multihead cross-attentions and concatenations for information passing is extremely close when testing on our app data. However, the attention-based approach avoids generating prohibitively long embeddings that may cause memory overflow.

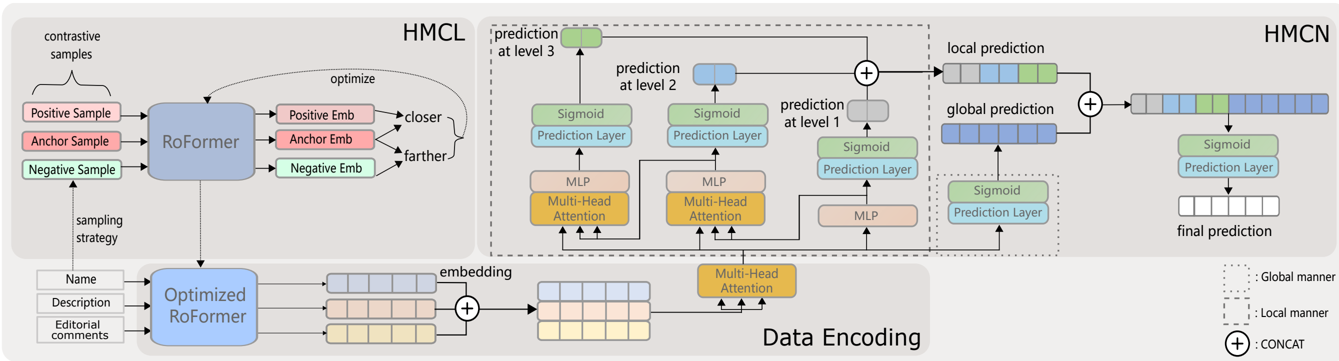

Figure 3: The HMCL + HMCN architecture. In HMCN, the left dotted block refers to the local manner, and the right dotted block refers to global one. The Prediction Layer refers to the last MLP layer for prediction, and $\oplus$ indicates a CONCAT operation.

Provided the local embedding, the local prediction is then

$$\hat{\mathbf{z}}^{(\ell)} = \mathrm{Sigmoid}(\mathrm{MLP}_\ell(\mathbf{h}_\ell)), \quad \forall \ell = 1, \ldots, L \tag{6}$$

where the Sigmoid layer is widely adopted as the multilabel classification layer. Furthermore, the MLPs for computing the prediction logits also have to be localized for better fitting capacity. Following that we can construct the local embedding $\hat{\mathbf{z}}_l$ using Eq. (4).

3.3.2 Global Manner. The global manner is straightforward to implement. Let MLP_g denote the global MLP. We obtain the global prediction $\hat{\mathbf{z}}_{global}$ by

$$\hat{\mathbf{z}}_{global} = \mathrm{Sigmoid}(\mathrm{MLP}_g(\mathbf{h}_0))\,. \tag{7}$$

3.3.3 Prediction Integration. One may merge the local and global predictions using a weighted mixture. In our task, the two predictions are combined through an MLP, i.e.,

$$\hat{\mathbf{z}} = \mathrm{MLP}(\mathrm{CONCAT}([\hat{\mathbf{z}}_{local}, \hat{\mathbf{z}}_{global}])) \tag{8}$$

based on Eqs. (4) and (7). This approach can also be regarded as a strategy for acquiring the ensemble of the two predictions.

3.3.4 Path Regularization. To mitigate the problem of path violation, we penalize the cases that $p(y_v = 1|x) > p(y_u = 1|x)$ where u is the parent label of v, for any single x. Given any data point, the likelihood of assigning a value of 1 to the parent label should be at least as high as the likelihood of assigning 1 to any of its child labels. We adopt a simple hinge loss to define the regularization term R as

$$R(x) = \sum_{u,v \in V} \mathbb{1}\{u = parent(v)\} \max(0, \hat{z}_v - \hat{z}_u)\,, \tag{9}$$

such that the sub-term $\hat{z}_v - \hat{z}_u$ can only contribute to the gradient computations when $\hat{z}_v > \hat{z}_u$, i.e., $p(y_v = 1|x) > p(y_u = 1|x)$. In such a case, the sub-term will be minimized.

3.3.5 Optimization. We are now able to detail the optimization implementations. We select the focal loss (FL) [21] as the main loss function, such that

$$\mathrm{FL}(\hat{\mathbf{z}}, \mathbf{y}) = \sum_{v \in V} -\alpha \left[y_v (1 - \hat{z}_v)^\gamma \log \hat{z}_v + (1 - y_v) \hat{z}_v^\gamma \log(1 - \hat{z}_v) \right]$$

where α and γ provide greater flexibility in handling minority labels[4], through assigning greater weights to the data points that are hard to learn. Hence, the final loss function is

$$\mathcal{L} = \sum_{(x,\mathbf{y}) \in \mathcal{D}} \mathrm{FL}(\hat{\mathbf{z}}, \mathbf{y}) + \lambda R(x) \tag{10}$$

given λ a coefficient for weighting the regularization term. When the value of λ is relatively small, the model tends to learn violated paths that favor the statistical characteristics of data over the hierarchical structure [35]. Conversely, with a relatively large value of λ, the model tends to assign smaller probabilities to deeper labels, which might affect the convergence process.

4 Hierarchical Multilabel Contrastive Learning

The HMCL process is applied prior to training the classification model HMCN to obtain the pretrained starting point. A good starting point of the model weights could benefit the subsequent training tasks and help to learn a more generalized model [10]. One key component to training a successful pretrained model in contrastive learning is constructing effective positive and negative sample pairs Even though constructing positive and negative sample pairs in multiclass classifications is considered straightforward, the task becomes far more challenging in the multilabel classification scenarios [7, 32, 39], in particular when hierarchical structures are imposed as constraints [40]. In this work, we propose three negative sampling strategies: ALL, LEVEL, and SIBLING, which will later be examined in Section 5.2. In the following discussion, we will detail each of them.

The priority of HMCL lies in the sampling strategy rather than the conventional contrastive learning loss function design. The positive sampling procedure is consistent, while the choice of negative sampling strategies will be the key aspect to our empirical success. Clearly, a pair (x, x') can be a negative pair even if they shared certain labels (but the labels for them are not identical). They are probably more often regarded as a positive pair if they have more shared labels. Meanwhile, they could still be a negative pair as long as they have a difference in the label assignment. With randomness, the sampling approach is equivalent to imposing weights that

[4] Minority labels refer to the labels that contain only a minor collection of data instances.

Algorithm 1: Sampling positive instances for x_i at level ℓ

Initialize $\mathcal{X}_{i\ell}^+ \leftarrow \emptyset$;
for *each anchor label* v *in* $V_{i\ell}^+$ **do**
    Sample x uniformly from $\mathcal{X}_v$ and add it to the set $\mathcal{X}_{i\ell}^+$
return $\mathcal{X}_{i\ell}$

Algorithm 2: Negative sampling for x_i at level ℓ

Initialize $\mathcal{X}_{i\ell}^- \leftarrow \emptyset$;
for *each anchor label* v *in* $V_{i\ell}^+$ **do**
    Sample a negative label u from $V_{\neg v}$;
    Sample an instance x which has active u but inactive v ;
    Add x to $\mathcal{X}_{i\ell}^-$;
return $\mathcal{X}_{i\ell}^-$

reflect the degree of label overlap in contrastive pairs, taking into account their hierarchical relationships.

4.1 Positive Sampling

We adopt the approach of [40], sampling contrastive data points level by level. Following this principle, the sampling process for each data point at a given level is performed in a label-wise manner.

Let $V_{i\ell}^+$ denote the positive label set for x_i, which contains the active labels for x_i at the ℓ-th level. Conversely, let $V_{i\ell}^-$ denote the corresponding negative label set, containing all the inactive labels for x_i at level ℓ. While it constructs the label sets given one datum x, we also construct sub-sample sets given on every label. We thus denote the set of sub-samples whose label assignment of label v is active by $\mathcal{X}_v$. Now, let $\mathcal{X}_{i\ell}^+$ be the positive sample set for x_i at level ℓ. We sample one single instance from every positive label of x_i, as shown in Algorithm 1.

4.2 Negative Sampling Strategies

Our solutions focus on constructing effective negative label level-wise that propagate to the samples associated with these labels. The pair construction process is generally decomposed for each data point and executed level by level. At each level, negative sampling is performed label by label. For a given label v, referred to as the anchor label in this context, negative labels are sampled relative to v. Assuming that v is at level ℓ, we focus on constructing the following two sets:

(1) $V_{\neg v}$ the negative sample set of the anchor label v;
(2) $\mathcal{X}_{i\ell}^-$ the subset of negative samples of x_i at the ℓ-th level.

Algorithm 2 depicts the procedure of this strategy. For each data point x_i, we sample negative samples at every level of the hierarchy. However, a single data point may have multiple active labels at any given level. Thus, the algorithm iterates through each active label, treating it as an anchor label v. For each anchor label v, we construct the corresponding negative label set $V_{\neg v}$ for v. Specifically, with respect to the anchor label v, the process involves two key steps: 1) sampling a negative label from $V_{\neg v}$, and 2) sampling an instance associated with this negative label. Crucially, the sampled instance must not have the anchor label v active, ensuring that the negative sample is distinct and meaningful in the context of CL. Finally, we construct the level-wise negative sample sets $\{\mathcal{X}_{i1}, \ldots, \mathcal{X}_{iL}\}$, which are instrumental in computing the final contrastive loss. Instead of sampling directly from the entire negative sub-space, sampling the negative labels in the first place ensures that the minority labels can be equally treated and involved in the contrastive learning.

The nature of sampling strategy is to determine the negative sample set that constrains the sampling space, which is in particular directed by $V_{\neg v}$. Consequently, the methodology used to construct $V_{\neg v}$ directly shapes the resulting negative sampling strategy. Next, we explore and analyze three distinct negative sampling strategies: All, Level, and Sibling.

4.2.1 Negative Sampling Strategy: All. To explain this scenario, let us first fix the anchor label for x_i to v. In the All strategy, and the negative label set $V_{\neg v}$ is constructed to include all labels across all levels, excluding the ancestors and successors of v. This implies that labels at higher or lower levels in the hierarchy, relative to v, are also eligible to be sampled as negative labels for v. To illustrate the process, we consider the example where the anchor label is set to "Game". In this case, the negative labels must exclude "Game" itself, as well as all its ancestors and successors (e.g., "Game-Moba", "Game-Strategy", etc.). This strategy promotes broad contrastive comparisons by drawing negative samples from a wide range of labels. Unfortunately, it turns out that this strategy disregards the hierarchical structure of the labels.

Furthermore, this approach is prone to sampling more negative labels from the lower levels of the hierarchy, as the number of labels grows exponentially when it comes closer to the leaf nodes. As a result, the negative data samples may be biased towards instances associated with labels that have a large number of leaf descendants, which diminishes the effectiveness of the contrastive comparisons.

4.2.2 Negative Sampling Strategy: Level. This strategy improves All to enhance the negative sample comparison by focusing the negative sample set that stays in the same level of that for the anchor label. Assume an anchor label for x_i at level ℓ is v. Let $V^{(\ell)}$ denote the set of labels at the level ℓ of the hierarchy. The negative set $V_{\neg u}$ is actually the set of labels at the same level except v itself: $V_{\neg v} = V^{(\ell)} \setminus \{v\}$. The construction of $V_{\neg v}$ leverages hierarchical structure by restricting negative samples to active labels at the same level as the anchor label v, explicitly excluding v itself. Consequently, the labels that are ancestors or successors of v in the hierarchy are inherently eliminated from the negative label pool. Hence, this strategy can avoid the situation in All strategy, where negative label samples are biased toward those with active labels at lower levels. Given any anchor label of x, we can limit the selection of negative samples to those at the same level, thereby constructing more balanced negative samples given every active label of x.

4.2.3 Negative Sampling Strategy: Sibling. This approach is inspired by [40]. The negative sample space for the anchor label v is restricted to its siblings for each level. Precisely, we have $V_{\neg v}$ to be all the siblings of v. The siblings of u have already excluded u by definition. In summary, the Sibling strategy attempts to better distinguish the data points with respect to the labels sharing the same parent node. The downside is that it restricts data comparison

Table 1: Comparison of contrastive labels generated by three negative sampling strategies, using the label hierarchy in Fig. 1 for an example data point with labels {"Finance", "Finance-Investment"}.

Level	Anchor label	Strategy	Contrastive Label Set
First	Finance	All	{Video, Game, Game-Moba, Game-RPG, Game-Strategy}
		Level	{Video, Game}
		Sibling	{Video, Game}
Second	Finance-Investment	All	{Finance, Finance-Loan, Finance-Loan-Credit Loan, Finance-Loan-Mortgage Loan, Video, Game, Game-Moba, Game-RPG, Game-Strategy}
		Level	{Finance-Loan, Game-Moba, Game-RPG, Game-Strategy}
		Sibling	{Finance-Loan}

to non-overlapping negative samples, potentially missing out on certain significant comparisons.

4.2.4 Comparison of the Negative Sampling Strategies. The main distinction between the three strategies is the scope of sampling negative samples based on anchor labels, which further determines different optimization directions. Noteworthy, for each anchor sample, our approach constructs effective negative labels for every level, propagating to the samples that contain these labels. We aim to largely avoid cases where negative labels are concentrated in certain labels, as this could hinder the generation of more balanced contrastive samples. At a higher level, the All strategy samples negative labels from *all* levels, the Level strategy samples from the *current* level of the anchor label, and the Sibling strategy samples only from the *sibling* node labels at the anchor label's current level. The All strategy is prone to sampling anchor labels from the lower levels, which slightly diverges from our goal.

Imagine that we would like to sample negative peers for an app with labels {"Finance", "Finance-Investment"}. Using the hierarchy in Fig. 1 as an example, we illustrate the possible sampling results from different negative sampling strategies (see Table 1). First, we sample the negative labels from the first level. Let the anchor label at the first level be "Finance". When using the All strategy to select the negative label, our contrastive set is {"Video", "Game", "Game-Moba", "Game-RPG", "Game-Strategy"}. We exclude "Finance-Loan", "Finance-Investment", "Finance-Loan-Credit Loan", "Finance-Loan-Mortgage Loan" because their first-level labels are also "Finance". In other words, they are all the descendant nodes of the anchor label "Finance". Notably, samples from "Game" or "Game-XX" share the same first-level label, "Game". Consequently, the probability of sampling the "Game" label at the first level is four times higher than that of sampling "Video", which introduces bias into the optimization process. The Level and Sibling strategies address this issue. The Level strategy returns {"Video", "Game"} as the set of contrastive labels, since these two labels are both first-level labels. In the case of the Sibling strategy, the contrastive label set is also {"Video", "Game"} due to the fact that they share the same parent label "Root".

We then sample the negative labels from the second level. Let the anchor label at the second level be "Finance-Investment" for our discussion. When using the All strategy, since the label "Finance-Investment" has no descendant nodes, the set of contrastive labels is {"Finance", "Finance-Loan", "Finance-Loan-Credit Loan", "Finance-Loan-Mortgage Loan", "Video", "Game", "Game-Moba", "Game-RPG", "Game-Strategy"}. In the case of the Level strategy, the set of contrastive labels is {"Finance-Loan", "Game-Moba", "Game-RPG", "Game-Strategy"}, since these labels are all second-level labels. Finally, when using the Sibling strategy, the set of contrastive labels contains only one element "Finance-Loan", because the label "Finance-Investment" has only one sibling node. The Level strategy empirically outperforms the Sibling strategy since it contrasts each anchor label with a broader distribution of negative sample labels at each level, thereby enhancing the discriminative capabilities.

4.3 Contrastive Loss

Let $\mathrm{Proj}(x)$ denote a neural network that employs our encoder to transform input x into $\mathbf{h}$ and a non-linear layer that projects the embedding $\mathbf{h}$ into a subspace, as outlined in the practitioner's guide in [5, 13, 15]. For simplicity, we write $\mathbf{s} = \mathrm{Proj}(x)$. We emphasize that a normalization layer is always appended to the end of the network, allowing us to perform easy cosine similarity computations. Most losses in contrastive learning are built on the Softmax function. For instance, the InfoNCE [25], a typical loss and a root for many variants, approximates the probability of correctly identifying the positive sample pairs via a Softmax function over the similarities between the positive and negative pairs.

However, our scenario is evidently more complex when defining the positive and negative pairs, since there are multiple levels for determining if a sample pair is positive or negative. What is even more challenging is that the same pair can be regarded as both cases under different anchor labels. Employing Softmax to model the probabilities of identifying the positive pairs may lead to a cumbersome loss formula and may demand a difficult code implementation. We therefore propose a more elegant solution, which was found to perform empirically in our task. Let us denote by $\Pr(x, x^+)$ and $\Pr(x, x^-)$ the probabilities of a positive pair (x, x^+) and of a negative pair (x, x^-), respectively. Inspired by multilabel classification, where labels are represented as binary vectors and binary cross-entropy is used as the loss function, we obtain

$$\Pr(x, x^+) = \mathrm{Sigmoid}(\mathbf{s}^\top \mathbf{s}^+ / \alpha); \tag{11}$$

$$\Pr(x, x^-) = 1 - \mathrm{Sigmoid}(\mathbf{s}^\top \mathbf{s}^- / \alpha) \tag{12}$$

Table 2: The statistics of datasets.

Dataset	Level	Label	Training	Validation	Test
app	3	177	30,628	3,798	14,286
RCV1	4	103	20,833	2,316	781,265
WOS	2	141	30,070	7,518	9,397

given α the scaling factor of the input value for Sigmoid. Recall that $\mathcal{X}^{+}_{i\ell}$ and $\mathcal{X}^{-}_{i\ell}$ are respectively the positive and negative sample set for x_i at level ℓ of the label hierarchy, generated through the sampling strategies. Assuming that $I(\mathcal{B})$ returns the indices of data in the batch $\mathcal{B}$, the batch-wise contrastive loss $\mathcal{L}_{cl}(\mathcal{B})$ is

$$\mathcal{L}_{cl}(\mathcal{B}) = \frac{1}{|\mathcal{B}|L} \sum_{i \in I(\mathcal{B})} \sum_{\ell=1}^{L} \frac{1}{|V^{+}_{i\ell}|} \left(\sum_{x^{+} \in \mathcal{X}^{+}_{i\ell}} \log \Pr(x_i, x^{+}) + \sum_{x^{-} \in \mathcal{X}^{-}_{i\ell}} \log \Pr(x_i, x^{-}) \right). \quad (13)$$

With this loss, one can effectively pretrain the classification model.

5 Experiments

In this section, we first present the experimental setup. Subsequently, we compare the three HMCL negative sampling strategies, and discuss the experimental findings. Finally, we discuss the deployment of the HMCN in combination with the HMCL and its impact on a realworld downstream task.

5.1 Experimental Settings

Our primary task was to train a model using the app data[5]. However, to examine the generalization ability of our approach, we also conducted experiments on two public datasets: the RCV1 dataset [20] and the WOS dataset [19]. The RCV1 data contains titles and abstracts (main text). It is worth noting that WOS is a hierarchical multiclass dataset; thus, there exists only one field of text in the data. We followed the strategy in [42] to split both datasets. The statistics of the datasets are summarized in Table 2.

Following a general practice [1, 42, 43], we adopted micro-F1 and macro-F1 as evaluation metrics. The micro-F1 computes the F1 score over the entire dataset whereas the macro-F1 is the interclass average F1 score. We compared our work with the following models on the RCV1 and WOS datasets: HiAGM [42], HTCInfoMax [8], and HiMatch [4]: These models applied a structural encoder to encode hierarchical labels and enhance model performance by matching the semantic similarity between label vectors and text vectors. HILL [43] and HGCLR [34] are two approaches that employ contrastive learning to augment the representation capabilities of the base text encoders. Due to the limit of space, we leave our implementation details in the Appendix.

5.2 Comparing the Negative Sampling Strategies

In this section, we compare the negative sampling strategies for the HMCL: All, Level, and Sibling. Table 3 illustrates the comparison of the three strategies on the test sets of the app and the RCV1 datasets, first presenting the performance of a single HMCN, followed by the performance of the HMCN with HMCL based on each sampling strategy. Overall, all three strategies outperform the independent HMCN significantly. This indicates that HMCL enhances the HMCN performance, regardless of which negative sampling strategy is used. Of the three negative sampling strategies, Level achieves the best performance, followed by Sibling and All.

[5] Part of the public data examples can be found in the Tencent App Store

Table 3: Comparison of the negative sampling strategies.

	RCV1		app	
	Micro-F1	**Macro-F1**	**Micro-F1**	**Macro-F1**
HMCN	87.52 ± 0.18	70.39 ± 0.10	79.67 ± 0.16	47.79 ± 0.19
All	87.13 ± 0.36	71.14 ± 0.34	80.16 ± 0.67	48.02 ± 1.19
Level	**87.92 ± 0.08**	**71.36 ± 0.40**	**80.75 ± 0.05**	**48.62 ± 0.39**
Sibling	87.67 ± 0.04	70.52 ± 0.34	80.34 ± 0.33	48.32 ± 0.75

Although the All strategy is a close performer to the best-performing Level strategy in terms of macro-F1 on the RCV1 dataset, Level consistently outperforms it across all results. This coincides with our analysis in Section 4.1 that the contrastive comparisons in Level are more effective than that in All. In Level, the selected negative labels for sampling are less biased toward leaf nodes, increasing the chances of picking samples associated exclusively with non-leaf labels. It enriches the diversity in the contrastive comparisons. The restrictions on constructing the negative samples in Level is the most balanced. The Sibling strategy outperforms All on all metrics except macro-F1 on the RCV1 dataset. Compared with the app dataset, which has evenly distributed labels, RCV1 exhibits a more imbalanced label distribution. We observe that Sibling underperforms on under-represented labels in comparison to All, consequently leading to a lower macro-F1 score. Given the superior performance compared to All and Sibling, in the rest of the experiments, we only report the results using the Level strategy. That is, HMCL will be limited to "HMCL with the Level strategy" unless otherwise noted.

5.3 Results for the App Data

We implemented the BERT and RoFormer model using a global multilabel classification approach. The two best performing state-of-the-art (SOTA) approaches on the two public datasets, HILL and HGCLR, were also implemented for comparison. BERT was retained in the implementations, since the HGCLR and HILL are tightly integrated with it.

Table 4 shows the results for the app data. First, we observe that the BERT and RoFormer solutions obtain a close performance despite that RoFormer is better at the micro-F1. The RoFormer-based HMCN demonstrates a significant improvement on both metrics. This could be attributed to the local manner of handling classification, and the information transferred from the previous level is helpful. In addition, we observe that the HGCLR and HILL achieve a close performance. They both outperformed the HMCN with respect to the micro-F1, and HILL outperforms the HMCN in regard to macro-F1. However, the HMCL enables the HMCN to outperform them on both metrics, making our final approach the best-performing one for the data. This indicates that, the improved quality of app embeddings obtained via HMCL can effectively enhance the performance of HMCN. A detailed analysis of the app embedding quality with and without HMCL regarding uniformity and alignment [33] can be found in Appendix B.

5.4 Results for the Public Data

We report our test set results for the public datasets in Table 5. As the HGCLR and HILL employ only the field of abstract, we concatenated the title to the abstract when training these two models. The results of using this modification for these two models both show improvements in the two metrics. For our own models, the means and standard deviations are presented. On the RCV1 dataset, the HMCN with HMCL outperforms all the SOTA solutions. Interestingly, the performance of the single HMCN (with BERT as the base text encoder) is close to that of the SOTA approaches. As expected, the HMCL is able to lift the HMCN to achieve increases in both metrics. The results imply that the HMCL effectively helps the instances with active minority labels gain better recognition regarding their representations.

The HILL performs the best for the WOS dataset. We notice that the WOS dataset contains only one field, namely the main text, whereas the RCV1 dataset contains article titles and main text, and the app data contains more fields. With only one field in the data, the HMCN might not be able to exploit its full potential since it is designed to cope with data containing multiple fields. Also, WOS is a hierarchical *multiclass* dataset which might further hinder the HMCN from performing, as the HMCN is designed for a *mutlilabel* scenario. In particular, the cross-attention for information transfer between levels might be less effective when using only one field. However, it shows that the HMCL is consistently capable of enhancing the performance of the HMCN. This also evidently supports the effectiveness of our contrastive learning procedures.

5.5 Deployment and Impact

5.5.1 App classification. As illustrated in illustrated in Fig. 4, we first gather all the app information from the App Store, including the app name, description, editorial comments, etc. Next, we sample a subset of apps to train and evaluate the model. This subset will go through several iterations of human labeling to ensure that the resulting ground truth labels are of high quality for training. After training, the model is deployed to infer labels for all apps. To account for new and obsolete apps in the App Store, we repeat the inference process monthly to update the labels for all apps. Moreover, we regularly fine-tune HMCN with newly labeled data

Table 4: Empirical results on the app dataset.

	Micro-F1	**Macro-F1**
BERT	78.28 ± 0.06	46.59 ± 0.15
RoFormer	79.19 ± 0.09	46.57 ± 0.81
HGCLR	80.62	47.78
HILL	80.33	47.92
HMCN	79.67 ± 0.16	47.79 ± 0.19
HMCN & HMCL	**80.75 ± 0.05**	**48.62 ± 0.39**

Table 5: Experimental results on the public datasets.

	RCV1		**WOS**	
	Micro-F1	**Macro-F1**	**Micro-F1**	**Macro-F1**
BERT	86.26	67.35	86.26	80.58
HiAGM*	85.58	67.93	86.04	80.19
HTCInfoMax*	85.53	67.09	86.30	79.97
HiMatch*	86.33	68.66	86.70	81.06
HGCLR*	86.49	68.31	87.11	81.20
HILL*	87.31	70.12	**87.28**	**81.77**
HGCLR (title & abstract)	87.22	69.89	-	-
HILL (title & abstract)	87.70	70.96	-	-
HMCN	87.52 ± 0.18	70.39 ± 0.10	86.45 ± 0.08	80.91 ± 0.30
HMCN & HMCL	**87.92 ± 0.08**	**71.36 ± 0.40**	86.90 ± 0.15	81.07 ± 0.13

[1] The results for methods marked with * were collected directly from the paper for HILL [43].

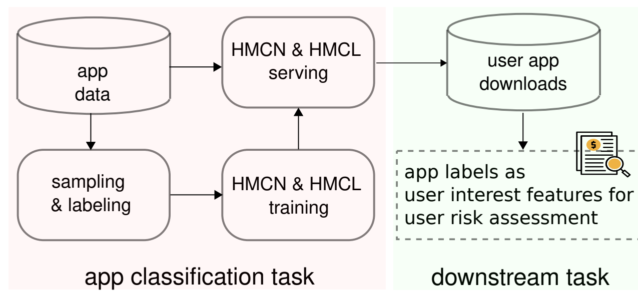

Figure 4: The deployment of the HMCN and HMCL, and the role of its app classifications in the downstream task of user risk assessment. The app labels assigned by the HMCN are used to extract user interests from the app download history of users, which are then merged with additional features for training and inference of the risk assessment model.

to enhance the model. Given the hyperparameters discussed in the Appendix, the training process of the HMCL and HMCN would respectively take around 52 hours and 3 hours to complete. These inferred labels serve as important features that are input into the downstream task.

5.5.2 Downstream task. We focus our discussion on a particular downstream task where the business objective is to assess the risk of users becoming victims of telecommunications fraud. Telecommunications fraud refers to the criminal act of defrauding victims into providing confidential information by using false information or disguised identities through communication means such as telephone, text messages, and the internet. The purpose of this downstream task is to identify in advance the risk of users potentially becoming victims based on user characteristics and to remind relevant organizations to take protective measures. The downstream business placeholder could incorporate our classification results to enhance their modeling on user interests. These user interest features are then merged with other features to train a dedicated machine learning model for assessing user risk.

To deploy our features online for the downstream task, a dedicated test dataset was prepared and held exclusively by the downstream business team. Features that can improve their evaluation metric beyond a certain threshold on this dataset were approved for integration into their online deployment. Once deployed, the marginal feature contribution would no longer be measured. The positive-to-negative sample ratio was around 1:10, with positive samples referring to the users that could encounter fraud.

Concretely, the effectiveness of user interest features is evaluated using the KS value of the downstream credit risk management model. The KS value is computed as follows. The downstream model predicts on their user data and generates the scores of these samples being positive. We define the cumulative distribution for identifying the positive and negative samples by $\mathrm{CDF}_p(t)$ and $\mathrm{CDF}_n(t)$ respectively, where t is the threshold value and

$$\mathrm{CDF}_p(t) = \frac{\text{No. of positive samples given score } < t}{\text{No. of positive samples}} \tag{14}$$

$$\mathrm{CDF}_n(t) = \frac{\text{No. of negative samples given score } < t}{\text{No. of negative samples}} . \tag{15}$$

The KS value measures the largest separation between the two CDFs, such that

$$\mathrm{KS} = \max_t \left|\mathrm{CDF}_p(t) - \mathrm{CDF}_n(t)\right| . \tag{16}$$

The higher KS indicates better separation between the two distributions. The business team processed the scores by first sorting them and then dividing them into 11 bins. The threshold values for each bin were defined by their upper bounds, excluding the first bin. The metric is used to measure a model's ability to distinguish between positive and negative samples, particularly suited for binary classification problems. It assesses the model's discriminative power by comparing the maximum difference between the cumulative distribution functions of positive and negative samples [22]. Integrating our app classification results leads to an improvement of **10.7%** with respect to the KS value on the test set. It implies that the model can be more accurate (+10.7%) in identifying users who might probably become victims, enabling relevant organizations to take preventive measures in advance. As a result, app labels classified using HMCN and HMCL have been successfully deployed online and integrated into the feature set used by the downstream task. The deployment has been in place for over one year.

6 Conclusion

In this paper, we proposed a systematic design for hierarchical multilabel classification of app labels on the Tencent App Store. Our approach adapts the HMCN to accommodate the multi-field, text-based description of apps. Additionally, we introduced the HMCL for pretraining the text encoder, discussed three negative sampling strategies, and examined the effectiveness of leveraging contrastive learning. With high-quality hierarchical multilabel classification results that are generated by the HMCN with HMCL, downstream business applications such as user risk assessment can identify the categories of focus and the extent of exploration, which in turn can improve their key performance metrics. Both offline experiments and the 10.70% performance boost in the downstream task demonstrate the efficacy of the HMCN with the text encoder pretrained through the HMCL. Since the textual information such as app names, descriptions, and editorial comments is generally available in other app stores, we believe our approach can be applied to many other app stores as well. For future work, additional downstream business applications will be explored to leverage the hierarchical multi-labels of apps.

Acknowledgments

We are grateful to the anonymous reviewers for the extremely constructive feedback, which has helped significantly improve this paper.

This project was initiated when Weipeng Huang was with Tencent. Weipeng Huang has been partly supported by Doctoral Research Initiation Program of Shenzhen Institute of Information Technology (Grant SZIIT2024KJ001) and Guangdong Research Center for Intelligent Computing and Systems (Grant PT2024C001).

References

[1] Rami Aly, Steffen Remus, and Chris Biemann. 2019. Hierarchical multi-label classification of text with capsule networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. 323–330.

[2] Siddhartha Banerjee, Cem Akkaya, Francisco Perez-Sorrosal, and Kostas Tsioutsiouliklis. 2019. Hierarchical transfer learning for multi-label text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 6295–6300.

[3] Ali Cevahir and Koji Murakami. 2016. Large-scale Multi-class and Hierarchical Product Categorization for an E-commerce Giant. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 525–535.

[4] 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)*. 4370–4379.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[6] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Vol. 1. IEEE, 539–546.

[7] Son D Dao, Ethan Zhao, Dinh Phung, and Jianfei Cai. 2021. Multi-label image classification with contrastive learning. *arXiv preprint arXiv:2107.11626* (2021).

[8] Zhongfen Deng, Hao Peng, Dongxiao He, Jianxin Li, and Philip S Yu. 2021. HTCInfoMax: A global model for hierarchical text classification via information maximization. *arXiv preprint arXiv:2104.05220* (2021).

[9] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research* 11, 19 (2010), 625–660. http://jmlr.org/papers/v11/erhan10a.html

[11] Rafael S Gonçalves, Matthew Horridge, Rui Li, Yu Liu, Mark A Musen, Csongor I Nyulas, Evelyn Obamos, Dhananjay Shrouty, and David Temple. 2019. Use of owl and semantic web technologies at pinterest. In *The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18*. Springer, 418–435.

[12] 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*. 257–265.

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems* 33 (2020), 21271–21284.

[14] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 9)*, Yee Whye Teh and Mike Titterington (Eds.). PMLR, Chia Laguna Resort, Sardinia, Italy, 297–304. https://proceedings.mlr.press/v9/gutmann10a.html

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[16] Wei Huang, Enhong Chen, Qi Liu, Yuying Chen, Zai Huang, Yang Liu, Zhou Zhao, Dan Zhang, and Shijin Wang. 2019. Hierarchical multi-label text classification: An attention-based recurrent network approach. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1051–1060.

[17] Weipeng Huang, Guangyuan Piao, Raul Moreno, and Neil J. Hurley. 2020. Partially Observable Markov Decision Process Modelling for Assessing Hierarchies. In *Asian Conference on Machine Learning*. PMLR, 641–656.
[18] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems* 33 (2020), 18661–18673.
[19] 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)*. IEEE, 364–371.
[20] 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 (2004), 361–397.
[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. 2017. Focal Loss for Dense Object Detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Los Alamitos, CA, USA, 2999–3007. doi:10.1109/ICCV.2017.324
[22] MA Liu, Jennifer Lewis Priestley Ph D, et al. 2018. A comparison of machine learning algorithms for prediction of past due service in commercial credit. (2018).
[23] Yuning Mao, Jingjing Tian, Jiawei Han, and Xiang Ren. 2019. Hierarchical text classification with reinforced label assignment. *arXiv preprint arXiv:1908.10419* (2019).
[24] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196* (2024).
[25] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).
[26] Kervy Rivas Rojas, Gina Bustamante, Arturo Oncevay, and Marco A Sobrevilla Cabezudo. 2020. Efficient strategies for hierarchical text classification: External knowledge and auxiliary tasks. *arXiv preprint arXiv:2005.02473* (2020).
[27] Ruslan Salakhutdinov and Geoff Hinton. 2007. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial intelligence and statistics*. PMLR, 412–419.
[28] Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. 2018. HFT-CNN: Learning hierarchical category structure for multi-label short text categorization. In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 811–816.
[29] Carlos N Silla and Alex A Freitas. 2011. A survey of hierarchical classification across different application domains. *Data mining and knowledge discovery* 22 (2011), 31–72.
[30] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing* 568 (2024), 127063.
[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).
[32] Ran Wang, Xinyu Dai, et al. 2022. Contrastive learning-enhanced nearest neighbor mechanism for multi-label text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 672–679.
[33] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*. PMLR, 9929–9939.
[34] Zihan Wang, Peiyi Wang, Lianzhe Huang, Xin Sun, and Houfeng Wang. 2022. Incorporating hierarchy into text encoder: a contrastive learning approach for hierarchical text classification. *arXiv preprint arXiv:2203.03825* (2022).
[35] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical multi-label classification networks. In *International conference on machine learning*. PMLR, 5075–5084.
[36] Linli Xu, Sijie Teng, Ruoyu Zhao, Junliang Guo, Chi Xiao, Deqiang Jiang, and Bo Ren. 2021. Hierarchical multi-label text classification with horizontal and vertical category correlations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2459–2468.
[37] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in neural information processing systems* 32 (2019).
[38] Fattane Zarrinkalam, Stefano Faralli, Guangyuan Piao, Ebrahim Bagheri, et al. 2020. Extracting, mining and predicting users' interests from social media. *Foundations and Trends® in Information Retrieval* 14, 5 (2020), 445–617.
[39] Pingyue Zhang and Mengyue Wu. 2024. Multi-Label Supervised Contrastive Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 15 (Mar. 2024), 16786–16793. doi:10.1609/aaai.v38i15.29619
[40] Shu Zhang, Ran Xu, Caiming Xiong, and Chetan Ramaiah. 2022. Use all the labels: A hierarchical multi-label contrastive learning framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16660–16669.
[41] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).
[42] 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*. 1106–1117.
[43] He Zhu, Junran Wu, Ruomei Liu, Yue Hou, Ze Yuan, Shangzhe Li, Yicheng Pan, and Ke Xu. 2024. HILL: Hierarchy-aware Information Lossless Contrastive Learning for Hierarchical Text Classification. *arXiv preprint arXiv:2403.17307* (2024).

A Implementation Details

We now specify the implementation of the contrastive learning. As discussed in Section 4, we sample positive and negative instances for anchor samples at each level. To ensure sufficient contrast among samples, we repeated this process multiple times. For the app dataset, we respectively performed this sampling process 10, 20, and 50 times for level 1, 2, and 3. For the RCV1 dataset, we respectively performed the sampling process 10, 20, and 50 times for level 1, 2, and 3. Since there is only one label at level 4, we would not conduct sampling at this level. At last, for the WOS dataset, we respectively repeated the sampling process 5 and 20 times, for level 1 and 2.

We learn that the number of samples per label, at higher hierarchical levels, is less than that at lower levels. Thus, the sampling multiples are correspondingly higher. After repeating the sampling for each anchor sample multiple times, we obtain a sufficient number of contrastive samples. We used the Adam optimizer with a mini-batch size of 8 and a learning rate of 1e-5 for all configurations. The learning rate decayed based on the number of training batches, reducing by a factor of 0.8 every 4,000 batches. Additionally, we set the scaling parameter α in the contrastive loss function to 0.1. As mentioned above, we repeated the same sampling strategy for each batch multiple times, we indeed increased the number of comparisons. Consequently, we observed that 1 epoch for the HMCL is always sufficient to perform.

With regard to the base text encoder, we applied RoFormer[6] for the app data, while BERT[7] was employed for both RCV1 and WOS datasets. The text encoder trained through the HMCL was used in the followed classification tasks. We applied the Adam optimizer with a batch size of 8 for all datasets. For all classification settings, we trained the model for 20 epochs and decayed the learning rate by a factor of 0.8 every two epochs. Regarding the app data, we set the initial learning rate to 5e-3. For the RCV1 and WOS datasets, we set the initial learning rate to 1e-4. The classification experiments for the public datasets were repeated 5 times under each configuration.

The hyperparameters for the focal loss were fixed to the default values $\alpha = 0.25, \gamma = 2$. The scaling factor for the path regularization λ was fixed to a simple choice of 1 under all configurations. When gauging the labels from the prediction logits, we applied the threshold of 0.5 to decide if a label v is considered active for x_i. That is, given any data index i and label v, we defined

$$y_{iv} = \begin{cases} 1 & z_{iv} \geq 0.5 \\ 0 & otherwise \end{cases} .$$

[6] https://huggingface.co/junnyu/roformer_chinese_base
[7] https://huggingface.co/google-bert/bert-base-cased

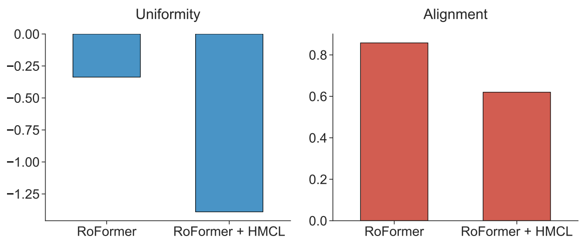

Figure 5: Improvement of app embedding quality with HMCL (lower values indicate better performance for both metrics)

Our experiments were conducted on a server equipped with the Xeon(R) Platinum 8372HC CPU with 3.40GHz, 8 NVIDIA A10 Tensor Core GPUs, and 360GB of RAM. The floating-point precision was set to BF16 (Brain Floating Point 16 bits).

B Uniformity and Alignment of App Embeddings with and without the HMCL

Here, we examine two important properties, *uniformity* and *alignment*, to assess the quality of app embeddings without and with HMCL. Uniformity measures how well the embeddings are spread out over the representation space, while alignment measures how close embeddings of positive pairs are in the representation space. They are formally defined as follows.

The *uniformity* property favors embeddings that are roughly uniformly distributed on the unit hypersphere, preserving as much information of the data as possible [33]. For randomly sampled pairs (x, y) with normalized embeddings $\mathbf{s}_x, \mathbf{s}_y$, the uniformity loss is formulated as:

$$\mathcal{L}_{\text{uniform}} = \log \mathbb{E}_{(x,\mathrm{y})\sim p_{\text{data}}} \left[\exp\left\{ \tau \left(\mathbf{s}_x^\top \mathbf{s}_y - 1 \right) \right\} \right] \tag{17}$$

where $\tau > 0$ is the temperature hyperparameter, p_{data} is the data distribution, and $\mathbf{s}_x^\top \mathbf{s}_y - 1 = -(1 - \mathbf{s}_x^\top \mathbf{s}_y)$ represents the negative cosine distance between $\mathbf{s}_x$ and $\mathbf{s}_y$.

The *alignment* property prefers that two samples forming a positive pair should have embeddings that are close each other, and thus be (mostly) invariant to irrelevant noise factors [33]. For positive pairs (x, x^+) with normalized embeddings $\mathbf{s}, \mathbf{s}^+$, the alignment loss is defined as:

$$\mathcal{L}_{\text{align}} = \sum_{l=1}^{L} \mathbb{E}_{(x,x^+)\sim p_{\text{pos}}^{(l)}} \left[1 - \mathbf{s}^\top \mathbf{s}^+ \right] \tag{18}$$

where $\mathbf{s}^\top \mathbf{s}^+$ represents cosine similarity and $1 - \mathbf{s}^\top \mathbf{s}^+$ denotes the corresponding cosine distance. Apart from that, $p_{\text{pos}}^{(l)}$ denotes the distribution of positive pairs at level l, where the two samples x and x^+ share at least one common label at that level.

For both metrics, lower values indicate better performance. As shown in Fig. 5, uniformity improves (decreases from -0.337 to -1.390), and alignment also improves (decreases from 0.858 to 0.602), indicating enhanced app embedding quality with HMCL.