Hierarchical Text Classification with Reinforced Label Assignment

Yuning Mao[1], **Jingjing Tian**[2], **Jiawei Han**[1], **Xiang Ren**[3]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
[2]Department of Computer Science, Peking University, Beijing, China
[3]Department of Computer Science, University of Southern California, CA, USA
[1]{yuningm2, hanj}@illinois.edu [2]tianjj97@pku.edu.cn [3]xiangren@usc.edu

Abstract

While existing hierarchical text classification (HTC) methods attempt to capture label hierarchies for model training, they either make local decisions regarding each label or completely ignore the hierarchy information during inference. To solve the mismatch between training and inference as well as modeling label dependencies in a more principled way, we formulate HTC as a Markov decision process and propose to learn a **L**abel **A**ssignment **P**olicy via deep reinforcement learning to determine *where to place* an object and *when to stop* the assignment process. The proposed method, **HiLAP**, explores the hierarchy during both training and inference time in a *consistent* manner and makes *inter-dependent* decisions. As a general framework, HiLAP can incorporate different neural encoders as *base models* for end-to-end training. Experiments on five public datasets and four base models show that HiLAP yields an average improvement of 33.4% in Macro-F1 over flat classifiers and outperforms state-of-the-art HTC methods by a large margin.[1]

1 Introduction

In recent years there has been a surge of interest in leveraging hierarchies (taxonomies) to organize objects (*e.g.*, documents), leading to the development of hierarchical text classification (HTC)—a task that aims to predict for an object multiple appropriate labels in a given label hierarchy, which together constitute a sub-tree. HTC methods have found a wide range of applications such as question answering (Qu et al., 2012), online advertising (Agrawal et al., 2013), and scientific literature organization (Peng et al., 2016). In contrast to "flat" classification, the key challenges of HTC lie in modeling the large-scale, imbalanced, and in particular, structured label space.

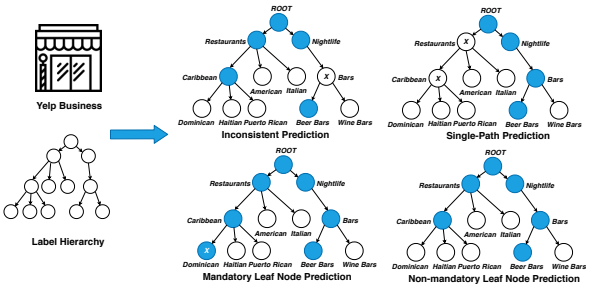

Figure 1: We aim at **consistent, multi-path, and non-mandatory leaf node prediction**. For a Caribbean restaurant with a beer bar, inconsistent prediction may place it to node "Beer Bars" but not "Bars", which contradicts with each other; Single-path prediction may only recognize that it is a beer bar; Mandatory leaf node prediction would have to assign a leaf node "Dominican" even if the nation of the cuisine is uncertain.

Based on how the hierarchy is explored, HTC methods can be summarized into *flat*, *local*, and *global* approaches (Silla and Freitas, 2011). Flat approaches (Hayete and Bienkowska, 2005; Johnson and Zhang, 2014) assume all the labels in the given hierarchy are independent. Some predict labels at the leaf nodes and heuristically add their ancestor labels, which is problematic as the labels of some objects may not be at the leaf nodes (*non-mandatory leaf node prediction*, see Fig. 1) and all the non-leaf nodes are completely neglected. Some simply ignore the hierarchy and perform standard multi-label classification, in which *label inconsistencies* (one label is predicted positive but its ancestors are not) may occur and post-processing is needed to correct such contradictions. Local approaches (Koller and Sahami, 1997; Cesa-Bianchi et al., 2006) train a set of local classifiers that function *independently* and predictions are usually made in a top-down order: one node is visited if and only if its ancestors have

[1]Data and code can be found at https://github.com/morningmoni/HiLAP.

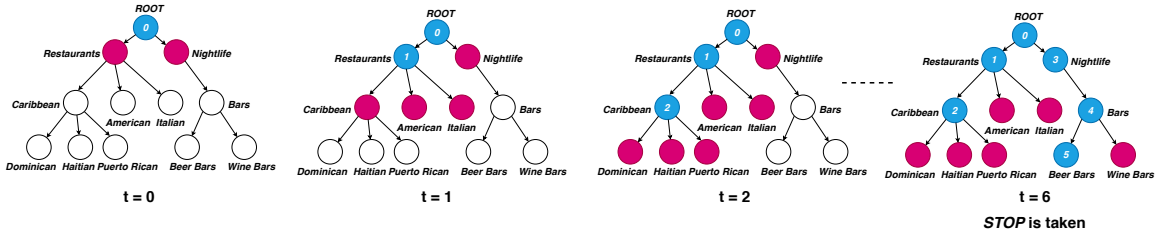

Figure 2: **An illustrative example of the label assignment policy.** At $t = 0$, x_i is placed at the root label and the policy would decide if x_i should be placed to its two children (*red*). At $t = 1$, x_i is placed at label "Restaurants", which adds its three children as the candidates. At $t = 6$, the *stop* action is taken and the label assignment is thus terminated. We then take all the labels where x_i has been placed (*blue*) as x_i's labels.

been predicted positive. One critical issue is that the number of local classifiers depends on the size of the label hierarchy, making local approaches infeasible to scale.

Global approaches use one single classifier and model the label hierarchy more explicitly. Traditional global approaches (Wang et al., 2001; Silla Jr and Freitas, 2009) are largely based on specific flat models and often make unrealistic assumptions (Cai and Hofmann, 2004) as in flat approaches. Recent neural approaches (Kim, 2014; Yang et al., 2016) mainly focus on flat classification while their performance in HTC is relatively less studied. Even if the classification is supposed to be hierarchical, prior work (Gopal and Yang, 2013; Johnson and Zhang, 2014; Peng et al., 2018) still make *flat* and *independent* predictions or utilize simple constraints without considering the holistic quality of label assignment. One recent framework (Wehrmann et al., 2018) attempts to leverage both local and global information but it uses static features as input and its inference process is still flat.

In this paper, we formulate HTC as a Markov decision process to better capture *label dependencies* and measure the *holistic quality* of label assignment. We present HiLAP, a global framework that learns a label assignment policy to determine *where to place* the objects and *when to stop* the assignment process. HiLAP explores the label hierarchy during both training and inference in a consistent manner, which alleviates the exposure bias often found in prior local and global approaches. By learning when to stop, HiLAP is more flexible than approaches that only support mandatory leaf node prediction or require thresholding. In addition, HiLAP supports multi-path prediction and its predictions of one object on different paths are *inter-dependent*, which not only guarantees *label consistency* but matches the nature of HTC. Furthermore, HiLAP estimates the holistic quality of all the labels assigned to one object via reinforcement learning instead of evaluating each label independently via maximum likelihood as in prior studies. To summarize, HiLAP achieves better effectiveness compared to flat and local approaches as it examines the label hierarchy during both training and inference. HiLAP has more flexibility and generalization capacity than previous global approaches in that it has no constraints on the structure of the hierarchy or the labels of the objects (Cai and Hofmann, 2004), generalizes to neural representation learning models (Gopal and Yang, 2013), and makes inter-dependent predictions while ensuring label consistency (Wehrmann et al., 2018; Peng et al., 2018).

HiLAP can be combined with various neural encoding models and trained in an end-to-end fashion. In our experiments, we select four representative encoding models as the *base models* to evaluate the effectiveness of HiLAP. Experimental results on five public datasets from different domains show that combining the base models with HiLAP yields an average performance improvement of 33.4% in Macro-F1 over corresponding flat classifiers and outperforms state-of-the-art HTC methods by a large margin. In particular, ablation study shows that HiLAP is especially beneficial to those unpopular labels at the bottom levels.

2 Hierarchical Label Assignment

2.1 Overview

Problem Formulation. We define a label hierarchy $H = (L, E)$ as a tree or DAG (directed acyclic graph)-structured hierarchy with a set of nodes (labels) L and a set of edges E indicating the parent-child relation between the labels. Taking a set of objects $\mathcal{X} = \{x_1, x_2, ..., x_N\}$ and their labels $\mathcal{L} = \{L_1, L_2, ..., L_N\}$ as input, we aim to

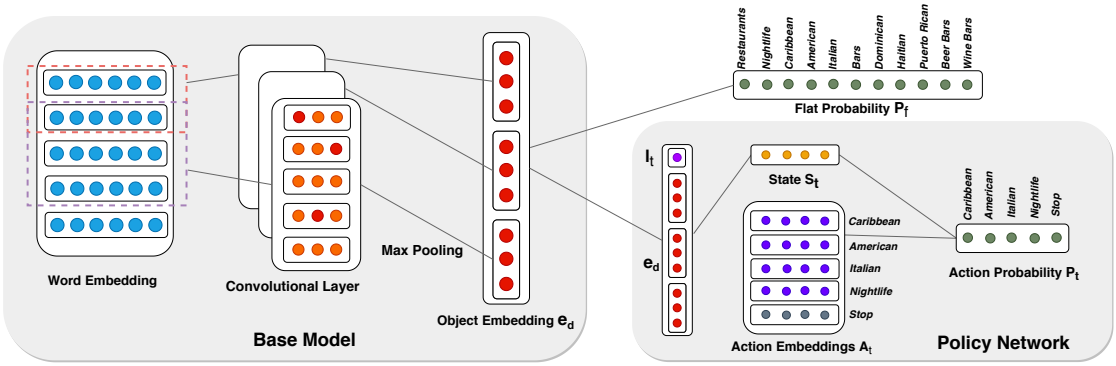

Figure 3: **The architecture of the proposed framework HiLAP**. One CNN model (Kim, 2014) is used as the base model for illustration. The object embedding $\mathbf{e}_d$ generated by the base model is combined with the embedding of currently assigned label $\mathbf{l}_t$ and used as the state representation $\mathbf{s}_t$, based on which actions are taken by the policy network. The time corresponds to $t = 1$ in Fig. 2.

learn a label assignment policy $\mathcal{P}$ to place each object x_i to its labels L_i on the label hierarchy H. The label assignment is supposed to be *consistent, multi-path, and non-mandatory leaf node prediction* (refer to Figs. 1 and 2). We define one *base model* $\mathcal{B}$ as a mapping f that converts raw object x_i to a finite dimensional vector, *i.e.*, the object embedding $\mathbf{e}_d \in \mathbb{R}^D$. $\mathcal{B}$ can be any neural representation learning model and its output $\mathbf{e}_d$ is used as the input of $\mathcal{P}$ for policy learning. The major challenge, compared to standard classification setup, is that we need to model E, *i.e.*, the relation between labels.

Our Framework. Prior studies either have a mismatch between training and inference as different routines are followed in the two phases, or compute losses with respect to each individual label and make flat predictions during inference time. In contrast, we learn a policy that (1) makes consistent, inter-dependent predictions by traversing the label hierarchy and maintaining state representation; (2) measures the holistic quality of label assignment via reinforcement learning. Specifically, the policy $\mathcal{P}$ puts x_i at the root label in the beginning. At each time step, $\mathcal{P}$ decides which label x_i should be further placed to, among all the *children* labels of where x_i has been placed, until a special *stop* action is taken. An illustration of how HiLAP labels one object is shown in Fig. 2 and the overall architecture of HiLAP is shown in Fig. 3.

2.2 Reinforcement Learning for Hierarchical Label Assignment

We describe the details of policy learning including its actions, rewards, states, and the policy network in this section. We formulate HTC as a Markov decision process (MDP): at each time step, the agent observes current state, takes an action, and receives a reward. The end goal is to train a policy network to determine where to place the objects and when to stop.

Actions. Specifically, we regard the process of placing an object x_i to the right positions on the label hierarchy as making a sequence of actions, where an action a_t at time step t is to select one label l_{t+1} from the action space $\mathcal{A}_t$ and place x_i to that label l_{t+1}. We denote the children of label l_t as $\mathcal{C}(l_t)$. At the beginning of each episode, x_i is placed at the root label l_0 and the action space $\mathcal{A}_0 = \mathcal{C}(l_0)$, *i.e.*, all the labels at level 1. When x_i is placed at another label l_1, its children $\mathcal{C}(l_1)$ are then added to the action space $\mathcal{A}_1$ while l_1 itself is removed. In addition, one *stop* action with embedding $\mathbf{e}_{\text{stop}} \in \mathbb{R}^C$ is included in the action space so that the model can automatically learn when to stop placing object x_i to new labels. Intuitively, when the confidence of placing x_i to another label is lower than the *stop* action, the label assignment process would be terminated.

In short, the action space $\mathcal{A}_t$ consists of all the *unvisited* children labels of where the object x_i *has been placed* and the *stop* action. One distinction of HiLAP is that it takes the inter-dependencies of labels across different paths and levels into consideration while previous approaches make independent predictions on different paths. For example, HiLAP can first place x_i to a label at level 3 if the probability of that label is high and then place it to another label at level 1 on another path.

Rewards. The agent receives scalar rewards as feedback for its actions. Different from exist-

ing work where each label of one example[2] is treated independently, HiLAP measures the quality of all the labels assigned to each example x_i by rewarding the agent with the *Example-based F1* (see Sec. 4.1 for details of this metric). Intuitively, the agent would realize how similar the assigned and the ground-truth labels of *one example* are. Instead of waiting until the end of the label assignment process and comparing the predicted labels with the gold labels, we use reward shaping (Mao et al., 2018), *i.e.*, giving intermediate rewards at each time step, to accelerate the learning process. Specifically, we set the reward r of x_i at time step t to be the difference of Example-based F1 scores between current and the last time step: $r_t^{x_i} = \text{F1}_t^{x_i} - \text{F1}_{t-1}^{x_i}$.

If current F1 is better than that at the last time step, the reward would be positive, and vice versa. The cumulative reward from current time step to the end of an episode would cancel the intermediate rewards and thus reflect whether the current action improves the holistic label assignment or not. As a result, the learned policy would not focus on the current placement but have a long-term view that takes following actions into account.

States and Policy Network. We parameterize action a_t by a policy network $\pi(\mathbf{a} \mid \mathbf{s}; \mathbf{W})$. For each object, its representation $\mathbf{e}_d$ is generated by the base model $\mathcal{B}$. For each label, a label embedding $\mathbf{l} \in \mathbb{R}^C$ is randomly initialized and updated during training. The embeddings of the object $\mathbf{e}_d$ and currently assigned label $\mathbf{l}_t$ are concatenated and projected to a vector $\mathbf{s}_t \in \mathbb{R}^C$ via a two-layer feed-forward network. $\mathbf{s}_t$ has the same size as the label embedding $\mathbf{l}$ and is used as the state representation at time step t. By stacking the action embeddings (*i.e.*, the embeddings of candidate labels and *stop* action), we obtain an action matrix $\mathbf{A}_t$ with size $|\mathcal{A}| \times C$. $\mathbf{A}_t$ is multiplied with the state embedding $\mathbf{s}_t$, which outputs the probability distribution of actions. Finally, an action a_t is sampled based on the probability distribution of the action space.

$$\begin{aligned} \mathbf{s}_t &= \text{ReLU}(\mathbf{W}_l^1 \text{ReLU}(\mathbf{W}_l^2[\mathbf{e}_d; \mathbf{l}_t])), \\ \pi(\mathbf{a}_t \mid \mathbf{s}; \mathbf{W}) &= \text{softmax}(\mathbf{A}_t \mathbf{s}_t), \\ a_t &\sim \pi(\mathbf{a}_t \mid \mathbf{s}; \mathbf{W}). \end{aligned}$$

We use policy gradient (Williams, 1992) as the optimization algorithm. In addition, we adopt a *self-critical* training approach (Rennie et al., 2017). For each object x_i, two label assignments are generated: $\tilde{L}_{x_i}$ is sampled from the probability distribution, and $\hat{L}_{x_i}$, the baseline label assignment, is greedily obtained by choosing the action with the highest probability at each time step. We use $\tilde{r}_t^{x_i} = r_t^{\tilde{L}_{x_i}} - r_t^{\hat{L}_{x_i}}$ as the actual reward, which ensures that the policy network learns to place the object to positions with higher F1 score than the greedy baseline. Formally, we measure the global loss $\mathcal{O}_g$ as follows.

$$\mathcal{O}_g = -\sum_{i=1}^{N}\sum_{t=1}^{T} log\pi^{x_i}(a_t \mid \mathbf{s}; \mathbf{W}) \times v_t^{x_i},$$

where $v_j^{x_i} = \sum_{t=j}^{T} \gamma^{t-j} \tilde{r}_t^{x_i}$ is the cumulative future reward at time j and $\gamma \in [0, 1]$ is the discount factor. At the time of inference, we greedily select labels with the highest probability as $\hat{L}_{x_i}$.

3 End-to-End Model Learning

3.1 Top-Down Supervised Pre-Training

Instead of learning from scratch, we use supervised learning to pre-train HiLAP. We denote the supervised variant as HiLAP-SL. While most parameters of HiLAP-SL are shared and used to initialize HiLAP (except that $\mathbf{e}_{\text{stop}}$ is randomly initialized), its way of exploring the label hierarchy H is dissimilar.

The major difference is that HiLAP-SL explores the label hierarchy H in a top-down manner independently. At each time step t, the object goes down one level on the hierarchy and the labels under the same parent are discriminated locally. Specifically, the local per-parent label probability distribution $\mathbf{p}_t^{\text{Local}}$ is estimated as $\mathbf{p}_t^{\text{Local}} = \sigma(\mathbf{C}_t \mathbf{s}_t)$, where σ denotes the sigmoid function, and $\mathbf{C}_t \in \mathbb{R}^{|\mathcal{C}(l_t)| \times C}$ denotes the candidate embeddings of HiLAP-SL, *i.e.*, an embedding matrix consisting of the children of *current* label l_t, rather than *all* the labels where x_i has been placed.

Another difference is that in HiLAP the actions are sampled and thus might place the objects to incorrect labels, while in HiLAP-SL only the ground-truth positions are traversed during training. Specifically, if there are $K(\geq 1)$ *ground-truth* labels at the same level, the object embedding $\mathbf{e}_d$ would be copied K times and losses on the K different paths would be measured independently (see Fig. 6 in Appendix for illustration). The local loss of HiLAP-SL is defined as $\mathcal{O}_l = \sum_{t=0}^{T} \mathcal{O}_t$, where T is the lowest label's

[2] We use "example" and "object" interchangeably.

level of one example and $\mathcal{O}_t$ estimates the binary cross entropy over the candidate labels $\mathcal{C}(l_t)$: $\mathcal{O}_t = -\sum_{i=1}^{N}\sum_{l\in\mathcal{C}(l_{t,i})} L_i(l) \times log\mathbf{p}_{t,i}^{\text{Local}}(l) + (1 - L_i(l)) \times log(1 - \mathbf{p}_{t,i}^{\text{Local}}(l))$, where $L_i(l)$ and $\mathbf{p}_{t,i}^{\text{Local}}(l)$ evaluate label l of x_i. Intuitively, HiLAP-SL works as if there were a set of local classifiers, although most of its parameters (except for the label embedding $\mathbf{l}$) are shared by all the labels so that there is no need to train multiple classifiers.

3.2 Combining Flat, Local, and Global Information for Policy Learning

We further add a *flat component* to HiLAP as a regularization of the base model. Specifically, the flat component is a feed-forward network that projects the object embedding $\mathbf{e}_d$ to a label probability distribution $\mathbf{p}^{\text{Flat}}$ of all the labels on the hierarchy: $\mathbf{p}^{\text{Flat}} = \sigma(\mathbf{W}^{\text{Flat}}\mathbf{e}_d)$. The combination of the base model and the flat component functions the same as a flat model and ensures that the object representation $\mathbf{e}_d$ has the capability of flat classification. We denote the flat loss that measures the binary cross entropy over all the labels by $\mathcal{O}_f = -\sum_{i=1}^{N}\sum_{l\in L} L_i(l) \times log\mathbf{p}_i^{\text{Flat}}(l) + (1 - L_i(l)) \times log(1 - \mathbf{p}_i^{\text{Flat}}(l))$. Combining the flat and local losses, the supervised loss in HiLAP-SL is defined as $\mathcal{O}_{\text{SL}} = \lambda\mathcal{O}_f + (1 - \lambda)\mathcal{O}_l$, where $\lambda \in [0, 1]$ is the mixing ratio. Similar to Celikyilmaz et al. (2018), we also found that mixing a proportion of the supervised loss is beneficial to the learning process of HiLAP. Further combining the global information $\mathcal{O}_g$ (*i.e.*, $\mathcal{O}_{\text{RL}}$), the total loss of HiLAP is defined as $\mathcal{O} = \mathcal{O}_{\text{RL}} + \alpha\mathcal{O}_{\text{SL}}$, where α is a scaling factor accounting for the difference in magnitude between $\mathcal{O}_{\text{RL}}$ and $\mathcal{O}_{\text{SL}}$. While we do not use the flat component during inference, it helps the representation learning of the base model and improves the performance of both HiLAP-SL and HiLAP (see Sec. 4.5).

4 Experiments

4.1 Experiment Setup

Datasets. We conduct extensive experiments on five public datasets from various domains (summarized in Table 1 and detailed in Appendix A). The first two datasets are related to news categorization, including RCV1 (Lewis et al., 2004) and the NYT annotated corpus (Sandhaus, 2008). The third dataset is the Yelp Dataset Challenge 2018[3]. We hypothesize that one business can be represented by its reviews and use the reviews to predict business categories. The last two datasets are related to protein functional catalogue (FunCat) and gene ontology (GO) prediction (Vens et al., 2008), which are used to test the generalization ability of HiLAP to non-textual data. For all the datasets, the lowest labels of one example may not be at the leaf nodes and there could be multiple labels at each level, making them harder and more realistic than *mandatory-leaf* or *single-path* datasets such as IPC (WIPO, 2014) and LSHTC (Partalas et al., 2015).

Table 1: **Statistics of the datasets.** $|L|$ denotes the number of labels in the label hierarchy. Avg($|L_i|$) and Max($|L_i|$) denote the average and maximum number of labels of one object, respectively.

Dataset	Hierarchy	$\|L\|$	Avg($\|L_i\|$)	Max($\|L_i\|$)	Training	Validation	Test
RCV1	Tree	103	3.24	17	23,149	2,315	781,265
NYT	Tree	115	2.52	14	25,279	2,528	10,828
Yelp	DAG	539	3.77	32	87,375	8,737	37,265
FunCat	Tree	499	8.76	45	1,628	848	1,281
GO	DAG	4,125	34.9	141	1,625	848	1,278

Evaluation Metrics. We use standard metrics (Johnson and Zhang, 2014; Meng et al., 2018; Peng et al., 2018) for HTC, including **Micro-F1**, **Macro-F1**, and **Example-based F1 (EBF)** (Partalas et al., 2015; Peng et al., 2016). Let TP_i, FP_i, FN_i denote the true positive, false positive, and false negative for the i-th *example* x_i in object set X, respectively. EBF calculates the F1 scores of all the *examples* independently and averages them. $P^{x_i} = \frac{TP_i}{TP_i+FP_i}$, $R^{x_i} = \frac{TP_i}{TP_i+FN_i}$, $\text{F1}^{x_i} = \frac{2P^{x_i}\times R^{x_i}}{P^{x_i}+R^{x_i}}$, and $\text{EBF} = \frac{1}{N}\sum_{i=1}^{N}\text{F1}^{x_i}$. Recall that F1^{x_i} is used as the reward in HiLAP.

Base Models for Feature Encoding. Different from most of existing *global* HTC methods that rely on pre-specified features (Gopal and Yang, 2013) as input or build on specific models (Cai and Hofmann, 2004; Vens et al., 2008; Silla Jr and Freitas, 2009), our framework is trained in an end-to-end manner by leveraging a differentiable feature representation learning model as the *base model*. Specifically, we use **TextCNN** (Kim, 2014), **HAN** (Yang et al., 2016), **bow-CNN** (Johnson and Zhang, 2014) on the three textual datasets, and a **feed-forward network** on the two non-textual datasets. The details of the base models

[3] https://www.yelp.com/dataset/challenge

Table 2: **Performance comparison on RCV1.** * denotes the results reported in Peng et al. (2018) on the same dataset split. Note that the results of HR-SVM reported in Gopal and Yang (2013) are not comparable as they use a different hierarchy with 137 labels.

	Method	Micro-F1	Macro-F1	EBF
Flat	Leaf-SVM*	69.1	33.0	-
	SVM	80.4	46.2	80.5
	TextCNN	76.6	43.0	75.8
	HAN	75.3	40.6	76.1
	bow-CNN	82.7	44.7	83.3
Local & Global	TD-SVM	80.1	50.7	80.5
	HSVM*	69.3	33.3	-
	HR-SVM*	72.8	38.6	-
	HR-DGCNN*	76.1	43.2	-
	HMCN	80.8	54.6	82.2
	HiLAP (TextCNN)	78.6	50.5	80.1
	HiLAP (HAN)	75.4	45.5	77.4
	HiLAP (bow-CNN)	**83.3**	**60.1**	**85.0**

Table 3: **Performance comparison on the NYT and Yelp datasets.** We mainly compare with competitive baselines that perform well on RCV1.

Method	NYT			Yelp		
	Micro-F1	Macro-F1	EBF	Micro-F1	Macro-F1	EBF
SVM	72.4	37.1	74.0	66.9	36.3	68.0
TextCNN	69.5	39.5	71.6	62.8	27.3	63.1
HAN	62.8	22.8	65.5	66.7	29.0	67.9
bow-CNN	72.9	33.4	74.1	63.6	23.9	63.9
TD-SVM	73.7	43.7	75.0	67.2	40.5	67.8
HMCN	72.2	47.4	74.2	66.4	42.7	67.6
HiLAP (TextCNN)	69.9	43.2	72.8	65.5	37.3	68.4
HiLAP (HAN)	65.2	28.7	68.0	**69.7**	38.1	**72.4**
HiLAP (bow-CNN)	**74.6**	**51.6**	**76.6**	68.9	**42.8**	71.5

are provided in Appendix C due to limited space.

To incorporate one base model into our framework, we remove its final feed-forward layer that projects the object representation $\mathbf{e}_d$ to a flat probability distribution of all labels ($\mathbf{p}^{\text{Flat}}$), and use $\mathbf{e}_d$ directly as the input of HiLAP. As one will see in the later experiments, HiLAP consistently improves the base model by modeling the label hierarchy in an effective manner.

4.2 Compared Methods

1. Traditional HTC Methods. A major line of work for HTC is Support Vector Machines (SVM) and its hierarchical variants. Specifically, **SVM** performs standard multi-label classification using one-vs-the-rest (OvR) strategy. **Leaf-SVM** treats each leaf node as a label and adds the ancestors of predicted leaf nodes. Variants such as **HSVM** (Tsochantaridis et al., 2005), Top-Down SVM (**TD-SVM**) (Liu et al., 2005), and Hierarchically Regularized SVM (**HR-SVM**) (Gopal and Yang, 2013) are also tested. Other state-of-the-art HTC methods that we compare with include **Clus-HMC** (Vens et al., 2008) and **CSSA** (Bi and Kwok, 2011).

2. Neural HTC Methods. There are not many neural methods that specifically target HTC. We mainly compare with two *latest* neural models: **HR-DGCNN** (Peng et al., 2018), which extends hierarchical regularization (Gopal and Yang, 2013) to Graph-CNN and compares favorably to flat models like RCNN (Lai et al., 2015) and XML-CNN (Liu et al., 2017), and **HMCN** (Wehrmann et al., 2018), which outperforms state-of-the-art HTC methods such as HMC-LMLP (Cerri et al., 2016). We also compare with the base models that we use for feature encoding. The main aim is to see how much gain they could obtain by combining each one of them with HiLAP.

4.3 Implementation Details

For datasets without held-out set, we randomly sample 10% from the training set as the validation set following Johnson and Zhang (2014); Peng et al. (2018). We only use the first 256 tokens of each document for representation learning. All the models are trained using an Adam optimizer with initial learning rate 1e-3 and weight decay 1e-6. We use GloVe (Pennington et al., 2014) with size 50 as word embeddings for TextCNN (Kim, 2014) and HAN (Yang et al., 2016). We create a vocabulary of the most frequent 30,000 words in the training data and generate multi-hot vectors as the input of bow-CNN (Johnson and Zhang, 2014). For our framework, since the parameter updates are performed after T steps, we cache the object representation $\mathbf{e}_d$ and reuse it at each step for better efficiency. More details are provided in Appendix D for reproducibility.

4.4 Performance Comparison

1. Comparison with State-of-the-art Methods. We compare the performance of HiLAP to state-of-the-art HTC methods and show the results in Tables 2 and 3. On RCV1, HiLAP (HAN) achieves similar performance to HR-DGCNN even though the corresponding base model HAN is originally worse than HR-DGCNN. HiLAP (TextCNN) outperforms most baselines in Macro-F1 and perform similarly to TD-SVM despite that it uses one global classifier while TD-

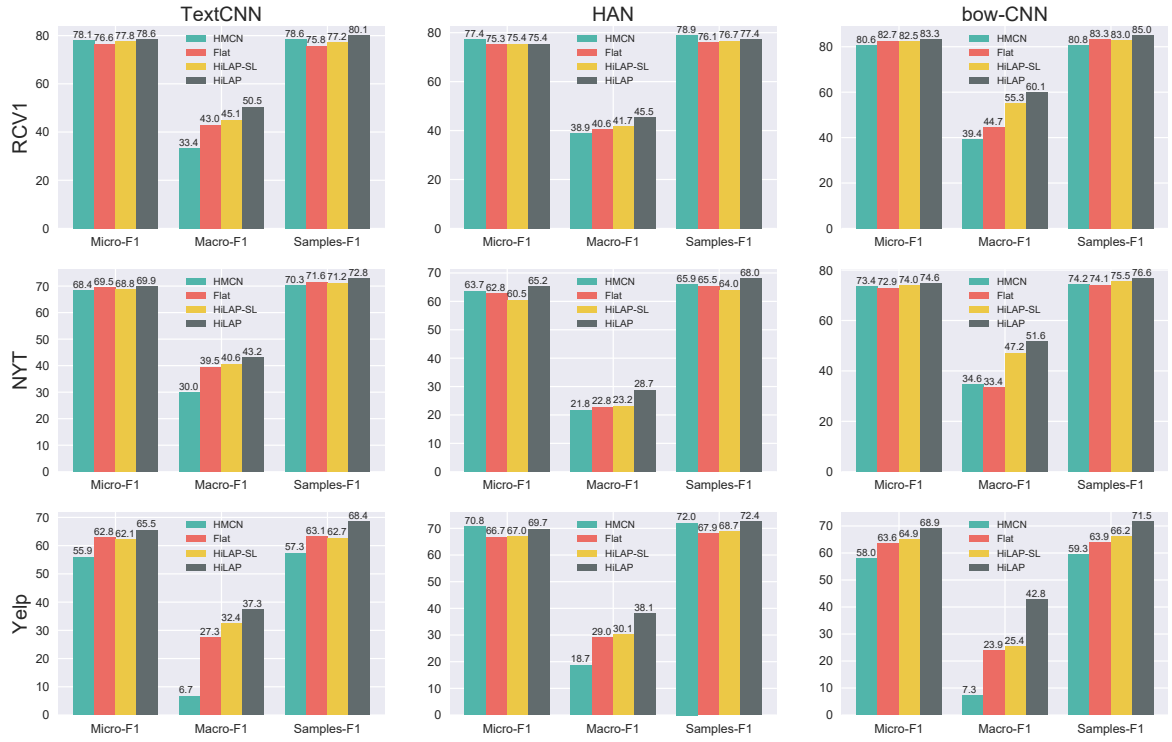

Figure 4: **Performance comparison of different classification frameworks using the same base models.** We compare HiLAP with its flat, supervised variants, and HMCN. Results show that HiLAP exhibits consistent improvement over flat classifiers and larger gains than HMCN.

SVM uses a set of classifiers. Among all compared methods, HiLAP (bow-CNN) achieves the best performance on all the three metrics.[4] On NYT, similar results are observed: TextCNN and HAN are both improved when combining with HiLAP and HiLAP (bow-CNN) again achieves the best performance. On Yelp, HiLAP (HAN) achieves the best Micro-F1 and EBF, while HiLAP (bow-CNN) obtains the highest Macro-F1.

2. Comparison using Same Base Models. We compare the performance of different frameworks that support the use of *exactly the same* base models and summarize the results in Fig. 4.[5] Due to the extreme imbalance of the data, directly applying a flat model may suffer from low Macro-F1, *i.e.*, the predictions of flat models are inevitably biased to the most popular labels. HMCN also has the same issue, resulting in Macro-F1 lower than 10 when combining with some base models. In contrast, HiLAP outperforms the baselines significantly in Macro-F1, which implies that our method is better at tackling labels with relatively few examples. It is also observed that HiLAP-SL sometimes may have a negative effect in terms of Micro-F1, although it is usually marginal compared with the gains in Macro-F1. However, such negative effects are eliminated by HiLAP through better exploration of the label hierarchy. Overall, HiLAP achieves the highest performance on 24 of 27 results among the combinations of three datasets, three base models, and three evaluation metrics. In particular, HiLAP yields an average improvement of 33.4% in Macro-F1 compared to corresponding base models.

Table 4: **Performance comparison on Functional Catalogue and Gene Ontology.** We compare with state-of-the-art hierarchical classification methods that take exactly the same raw features as input (*i.e.*, we exclude models designed specifically for text objects).

Method	FunCat			GO		
	Micro-F1	Macro-F1	EBF	Micro-F1	Macro-F1	EBF
SVM	2.72	1.21	3.42	34.1	1.46	36.8
CSSA	16.0	4.60	14.8	11.6	0.76	11.5
CLUS-HMC	25.2	4.14	24.1	41.4	3.01	40.3
HMCN	21.3	5.07	21.5	43.2	3.81	43.3
HiLAP	**26.5**	**7.50**	**27.4**	**45.4**	**5.87**	**45.2**

3. Results on Functional Genomics Prediction. We compare HiLAP with CSSA (Bi and

[4]The results are not comparable with Johnson and Zhang (2014) due to implementation details and the fact that they tune the threshold for each label using k-fold cross-validation. See Appendix B for more discussions.

[5]For HMCN, we replace its static features with the same base model for fair comparison.

Table 5: **Ablation study of HiLAP.** We evaluate variants of HiLAP using bow-CNN (Johnson and Zhang, 2014) on RCV1 (Lewis et al., 2004).

Method	Micro-F1	Macro-F1	EBF
Flat-Only	82.7	44.7	83.3
HiLAP-SL-NoFlat	81.0	52.1	81.7
HiLAP-SL	82.5	55.3	83.0
HiLAP-NoSL	83.2	59.3	85.0
HiLAP-NoFlat	83.0	59.8	84.7
HiLAP	**83.3**	**60.1**	**85.0**

Kwok, 2011), CLUS-HMC (Vens et al., 2008), and HMCN (Wehrmann et al., 2018) on the FunCat and GO datasets, as they represent the state-of-the-art on these datasets. An SVM classifier is also evaluated to better understand the difficulties of the task. We use the same raw features as the input of all the methods for apples-to-apples comparison and list the results in Table 4. Note that the metric *area under the average precision-recall curve (AUPRC)* (Wehrmann et al., 2018) is not applicable because HiLAP does not use a flat probability distribution of all the labels. As one can see, HiLAP outperforms all the baselines on both datasets by a large margin. In particular, we observe significant improvement on Macro-F1 over the best baseline (47.9% and 53.9%, respectively), which shows that our method is especially better at classifying sparse labels than previous approaches.

4.5 Performance Analysis

1. Ablation Study on Different Framework Components. We show the ablation analysis of HiLAP in Table 5. Using *Flat-Only* degenerates HiLAP to the flat baseline. By comparing the results of *Flat-Only* and HiLAP-SL-NoFlat (a variant of HiLAP-SL without flat loss), we further confirm that flat approaches are likely to neglect sparse labels, which results in low Macro-F1. Local approaches (HiLAP-SL-NoFlat), on the other hand, are slightly worse in terms of Micro-F1 and EBF but significantly better on Macro-F1. By combining flat and local information, HiLAP-SL achieves performance close to *Flat-Only* on Micro-F1 and EBF, and even higher Macro-F1 than HiLAP-SL-NoFlat. HiLAP-NoSL is initialized by the pre-trained HiLAP-SL model without mixing the supervised loss during its training. We can see that using the reinforced loss alone still improves the performance on all the three metrics. After removing the flat loss during the training of HiLAP, HiLAP-NoFlat shows slightly lower performance than the full HiLAP model, indicating that the flat component serves as a regularization of the base model and is beneficial to the overall performance.

2. Performance Study on Label Granularity and Popularity. We analyze the sources of performance gains by dividing the labels based on their levels and number of supporting examples. Fig. 5 shows the absolute Macro-F1 differences between several methods and the base model. We observe similar results for other setups and omit them for a clearer view. As depicted in Fig. 5, HiLAP and HiLAP-SL are especially beneficial to unpopular labels (P3) at the bottom levels (L3).

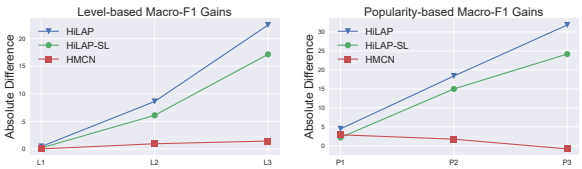

Figure 5: **Performance Study on Label Granularity and Popularity.** We compute level-based and popularity-based Macro-F1 gains on NYT with bow-CNN as base model. We denote the levels of the hierarchy with L1, L2, and L3 (*left*) and divide the labels into three equal sized categories (P1, P2, and P3) in a descending order by their number of examples (*right*).

3. Analysis of Label Inconsistency. Label inconsistencies often happen in approaches that perform flat inference, but they are not measured by standard evaluation metrics like F1 scores. To provide a picture of how severe the issue is, we further conduct experiments to check the percentage of objects that are predicted with inconsistent labels (Table 6). We found, for example, 29,186/781,265 (3.74%) predictions of TextCNN have inconsistent on RCV1. In contrast, HiLAP ensures **0%** label inconsistency without the need of post-processing, because its predictions are always valid sub-trees of the label hierarchy (refer to Fig. 2).

Table 6: **Analysis of Label Inconsistency.** We compare various methods by the percentage of predictions with inconsistent labels on RCV1 (Lewis et al., 2004).

SVM	TextCNN	HMCN	HiLAP
4.83%	3.74%	3.84%	**0%**

5 Related Work

Hierarchical classification approaches have been developed for many applications. For text classification, both traditional methods (Lewis et al., 2004; Gopal and Yang, 2013) and neural methods (Johnson and Zhang, 2014; Peng et al., 2018) have been proposed to classify, *e.g.*, the topics of newswire and web content (Sun and Lim, 2001) or categories of laws and patents (Bi and Kwok, 2015; Cai and Hofmann, 2004; Rousu et al., 2005). Many previous studies (Liu et al., 2005; Sun and Lim, 2001) train a set of local classifiers and make predictions in a top-down manner. In particular, Bi and Kwok (2015) develop Bayes-optimal predictions that minimize the global risks but their model is still locally trained. Such local approaches are not popularly used among recent neural-based HTC models (Johnson and Zhang, 2014; Peng et al., 2018) since it is usually infeasible to train many neural classifiers locally.

Global methods, on the other hand, train only one classifier. Although global methods are desirable, they are relatively less studied due to the complexity of the problem. Existing global models are generally modified based on specific flat models. Hierarchical-SVM (Cai and Hofmann, 2004; Qiu et al., 2009) generalizes Support Vector Machine (SVM) learning based on discriminant functions that are structured in a way that mirrors the label hierarchy. One limitation is that Hierarchical-SVM only supports balanced tree (all possible labels are presumed to be at the same level in their experiments). Hierarchical naive Bayes (Silla Jr and Freitas, 2009) modifies naive Bayes by updating weights of one's ancestors as well whenever one label's weights are updated. There are other global methods that are based on association rules (Wang et al., 2001), C4.5 (Clare and King, 2003), kernel machines (Rousu et al., 2005), and decision tree (Vens et al., 2008). Constraints such as the regularization that enforces the parameters of one node and its parent to be similar (Gopal and Yang, 2013) are also proposed to leverage the label hierarchy while maintaining scalability. However, their use of the label hierarchies is somewhat limited compared with HiLAP.

6 Conclusions

We proposed an end-to-end reinforcement learning approach to hierarchical text classification (HTC) where objects are labeled by placing them at the proper positions in the label hierarchy. The proposed framework makes *consistent* and *inter-dependent* predictions, in which any neural-based representation learning model can be used as a base model and a label assignment policy is learned to determine where to place the objects and when to stop the assignment process. Experiments on five public datasets and four base models showed that our approach outperforms state-of-the-art HTC methods significantly. For future work, we will explore the effectiveness of the proposed framework on other base models and forms of data (*e.g.*, images). We will introduce more losses covering other aspects in the objective function to further improve the performance of our framework.

Acknowledgments

Research was sponsored in part by U.S. Army Research Lab under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, grant 1U54GM-114838 awarded by NIGMS, National Science Foundation SMA 18-29268, DARPA MCS and GAILA, IARPA BETTER, Schmidt Family Foundation, Amazon Faculty Award, Google Research Award, Snapchat Gift, and JP Morgan AI Research Award. We thank Chao Zhang, Xiao-Yang Liu, Qingrong Chen, Jun Yan, collaborators in the INK research lab, and anonymous reviewers for their help and valuable feedback.

References

Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, pages 13–24. ACM.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Wei Bi and Jame T Kwok. 2015. Bayes-optimal hierarchical multilabel classification. *TKDE*, 27(11):2907–2918.

Wei Bi and James T Kwok. 2011. Multi-label classification on tree-and dag-structured hierarchies. In *ICML-11*, pages 17–24.

Lijuan Cai and Thomas Hofmann. 2004. Hierarchical document categorization with support vector machines. In *CIKM*, pages 78–87. ACM.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *NAACL*, pages 1662–1675.

Ricardo Cerri, Rodrigo C Barros, André CPLF de Carvalho, and Yaochu Jin. 2016. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC bioinformatics*, 17(1):373.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. 2006. Hierarchical classification: combining bayes with svm. In *ICML*, pages 177–184. ACM.

Amanda Clare and Ross D King. 2003. Predicting gene function in saccharomyces cerevisiae. *Bioinformatics*, 19(suppl_2):ii42–ii49.

Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *KDD*, pages 257–265. ACM.

Boris Hayete and Jadwiga R Bienkowska. 2005. Gotrees: predicting go associations from protein domain composition using decision trees. In *Biocomputing 2005*, pages 127–138. World Scientific.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Daphne Koller and Mehran Sahami. 1997. Hierarchically classifying documents using very few words. In *ICML*, pages 170–178. Morgan Kaufmann Publishers Inc.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

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

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM.

Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *Acm Sigkdd Explorations Newsletter*, 7(1):36–43.

Yuning Mao, Xiang Ren, Jiaming Shen, Xiaotao Gu, and Jiawei Han. 2018. End-to-end reinforcement learning for automatic taxonomy induction. In *ACL*, pages 2462–2472. Association for Computational Linguistics.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *CIKM*.

Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artières, George Paliouras, Éric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Gallinari. 2015. LSHTC: A benchmark for large-scale text classification. *CoRR*, abs/1503.08581.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *WWW*, pages 1063–1072.

Shengwen Peng, Ronghui You, Hongning Wang, Chengxiang Zhai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2016. Deepmesh: deep semantic representation for improving large-scale mesh indexing. *Bioinformatics*, 32(12):i70–i79.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Xipeng Qiu, Wenjun Gao, and Xuanjing Huang. 2009. Hierarchical multi-class text categorization with global margin maximization. In *acl-ijcnlp 2009*, pages 165–168.

Bo Qu, Gao Cong, Cuiping Li, Aixin Sun, and Hong Chen. 2012. An evaluation of classification models for question topic categorization. *JASIST*, 63:889–903.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, page 3.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. 2005. Learning hierarchical multi-category text classification models. In *ICML*, pages 744–751. ACM.

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

Carlos N Silla and Alex A Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72.

Carlos N Silla Jr and Alex A Freitas. 2009. A global-model naive bayes approach to the hierarchical prediction of protein functions. In *ICDM'09*, pages 992–997. IEEE.

Aixin Sun and Ee-Peng Lim. 2001. Hierarchical text classification and evaluation. In *ICDM*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484.

Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185.

Ke Wang, Senqiang Zhou, and Yu He. 2001. Hierarchical classification of real life documents. In *SDM*, pages 1–16. SIAM.

Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical multi-label classification networks. In *ICML*, pages 5225–5234.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

IPC WIPO. 2014. International patent classification (ipc). *World Intellectual Property Organization, Geneve*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*, pages 1480–1489.

A Reproducibility Details of Datasets

In this section, we describe the details of the datasets used in our experiments.

The RCV1 dataset (Lewis et al., 2004) is a manually labeled newswire collection of Reuters News from 1996 to 1997. Its news documents are categorized with three aspects: industries, topics, and regions. We follow the original training/test split for RCV1 and use its topic-based label hierarchy for classification as it has been well used in prior work (Gopal and Yang, 2013; Johnson and Zhang, 2014; Peng et al., 2018; Wehrmann et al., 2018). There are 103 categories and four levels in total including all labels except for the root label in the hierarchy.

The NYT annotated corpus (Sandhaus, 2008) is a collection of New York Times news from 1987 to 2007. Due to its large size, we randomly sampled 36,107 documents from all the news documents, and further split them into training and test set of 25,279 and 10,828 examples, respectively. We use the first three levels in the hierarchy and keep the labels with at least 40 supporting examples.

For the Yelp dataset, the label hierarchy is taken from the Yelp Business Categories[6], which Fig. 2 is a subset of. For preprocessing, we first removed categories that have fewer than 100 businesses and then businesses that have fewer than 5 reviews. We concatenated (at most) the first 10 reviews of each business as its representation. We set the training/test ratio to 70%/30%, which results in a training set of 87,375 examples and a test set of 37,517 examples. This is an even more challenging task because the reviews are usually written in an informal way and it is more imbalanced than the RCV1 or NYT datasets. For example, label *Restaurants* has 32,357 businesses in the training set while *Retirement Homes* has 23.

For the FunCat and GO datasets, we take the *cellcycle* data from (Vens et al., 2008)[7]. Compared with the text datasets above, raw features are provided as input for all compared methods. Furthermore, their training data is rather limited while the label space is much larger (4,125 vs. 539). Since there are many labels that do not have any example in either training set or test set, we exclude such labels when calculating Macro-F1. Note that it does not have any effect on the ratio of results from two different methods as the F1 scores of those labels without supporting examples are always zero. The features provided by the datasets are taken as input as they are except that the missing values are replaced with the mean value of corresponding features. All the compared methods take the same raw features for fair comparison.

[6] https://www.yelp.com/developers/documentation/v3/all_category_list

[7] https://dtai.cs.kuleuven.be/clus/hmcdatasets/

B Performance Analysis of Baselines

There are several things to note in terms of the performance of the baselines. First, our results are not comparable to Lewis et al. (2004); Johnson and Zhang (2014) due to implementation details (*e.g.*, we only take the first 256 tokens) and the fact that they tune the threshold for each label using *scutfbr* (Lewis et al., 2004). According to the implementation in LibSVM[8], the *scutfbr* threshold tuning algorithm uses two nested 3-fold cross validation for each of the 103 labels and the classifier is trained $3 \times 3 \times 103 = 927$ times, which is infeasible in our case.

Secondly, we found that the original performance of HMCN (Wehrmann et al., 2018) is sometimes much lower than expected. After tuning their model, we observed that if we first conduct a weighted sum of the local and global outputs and then apply the sigmoid function, the performance of HMCN becomes much better (see Table 7) than doing them in the opposite order as in Wehrmann et al. (2018). In addition, we found that HMCN + HAN (Yang et al., 2016) would result in extremely low performance. We had to remove HMCN's batch normalization to make it compatible with HAN. Combining HMCN with other base models did not encounter similar issues.

Thirdly, our implementation of TextCNN (Kim, 2014) and HAN (Yang et al., 2016) shows better performance than those reported in Peng et al. (2018) due to implementation details. A comparison can be found in Table 8.

C Details of Base Models

1. Base Models for Encoding Text Objects. For the text classification datasets, three representative text encoding models with different characteristics are selected as the base models to prove the robustness and versatility of HiLAP. We briefly describe

[8] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/multilabel/

Table 7: Comparison of different implementations of HMCN.

Model	RCV1			Yelp			NYT		
	Micro-F1	Macro-F1	EBF	Micro-F1	Macro-F1	EBF	Micro-F1	Macro-F1	EBF
HMCN (original)	78.2	33.2	78.9	56.3	8.5	57.3	62.1	32.4	62.7
HMCN (ours)	80.8	54.6	82.2	66.4	42.7	67.6	72.2	47.4	74.2

Table 8: Comparison of different implementations of HAN and TextCNN on the RCV1 dataset.

Model	Micro-F1	Macro-F1
TextCNN (in Peng et al. (2018))	73.2	39.9
TextCNN (ours)	76.6	43.0
HAN (in Peng et al. (2018))	69.6	32.7
HAN (ours)	75.3	40.6

the base models and the reasons we choose them as follows.

TextCNN (Kim, 2014) is a classic convolutional neural network for text classification. In our implementation, TextCNN is composed of one convolutional layer with three kernels of different sizes (3, 4, 5), followed by max pooling, dropout, and fully-connected layers. We choose TextCNN because it is one of the first successful and well used neural-based models for text classification.

HAN (Yang et al., 2016) first learns the representation of sentences by feeding words in each sentence to a GRU-based sequence encoder (Bahdanau et al., 2014) and then feeds the representation of the encoded sentences into another GRU-based sequence encoder, which generates the representation of the whole document. Attention mechanism such as word attention and sentence attention is also used. We choose HAN because it uses RNNs instead of CNNs and is shown to be effective on the *flat* Yelp Review datasets.

bow-CNN (Johnson and Zhang, 2014) employs bag of words (multi-hot zero-one vectors) as input to represent text objects and directly applies CNNs to the high-dimensional multi-hot vectors encoding. It learns the representation of small text regions (rather than single words) for use in classification. We choose bow-CNN since it does not use any word embeddings as in TextCNN and HAN. In addition, bow-CNN achieved state-of-the-art performance RCV1 (Lewis et al., 2004).

2. Base Model for Encoding Raw Features. For functional genomics prediction, one feed-forward neural network is used for simplicity as raw features are already provided in the datasets.

D Reproducibility Details of Implementation

We implement the base models and HMCN (Wehrmann et al., 2018) according to the original papers and existing implementations. We use the official implementation of Clus-HMC (Vens et al., 2008)[9] and one open-source implementation of CSSA (Bi and Kwok, 2011)[10]. We use *scikit-learn* for SVM-based methods. TF-IDF features are used for text classification when raw features are needed as input.

For our framework, we specify the number of steps in HiLAP-SL to be the number of levels in the label hierarchy. We set the maximum number of steps in HiLAP to be reasonably large (depending on the average number of labels of one object) so that it could explore the hierarchy and learn when to stop by itself. For the purpose of batch training, we convert the original indefinite-horizon MDPs to finite-horizon by adding an absorbing state, *i.e.*, after visiting the most fine-grained label in HiLAP-SL or entering the *stop* state in HiLAP, it would loop in the current state until the maximum number of steps, waiting for other objects in the same batch to finish.

We set the size of $\mathbf{W}_l^2$ to 500 and the sizes of $\mathbf{W}_l^1$ and label embedding $\mathbf{l}_t$ to 50 in all the text classification datasets and set them to 1,000 in the other datasets. We did not observe clear performance changes when varying the probability of dropout in base models like TextCNN. We set batch size to 32 as it performs well on the validation set and a batch size as large as 128 may cause performance losses.

E Additional Figure Illustration

[9] https://dtai.cs.kuleuven.be/clus/

[10] https://github.com/sushobhannayak/cssag

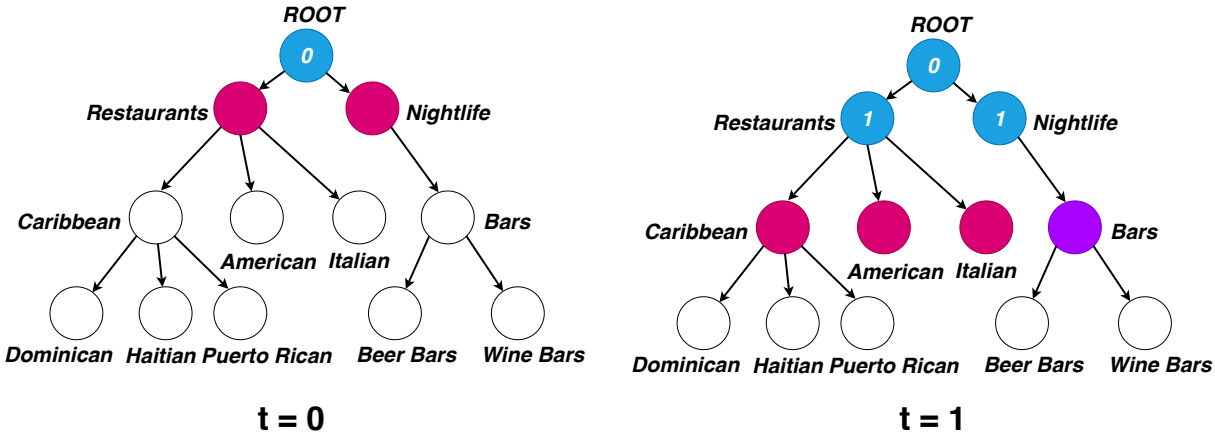

Figure 6: **One time step in HiLAP-SL.** At $t = 1$, two ($K = 2$) local per-parent probabilities $\mathbf{p}_1^{\text{Local}}$ are measured independently and aggregated in the loss function $\mathcal{O}_1$.