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

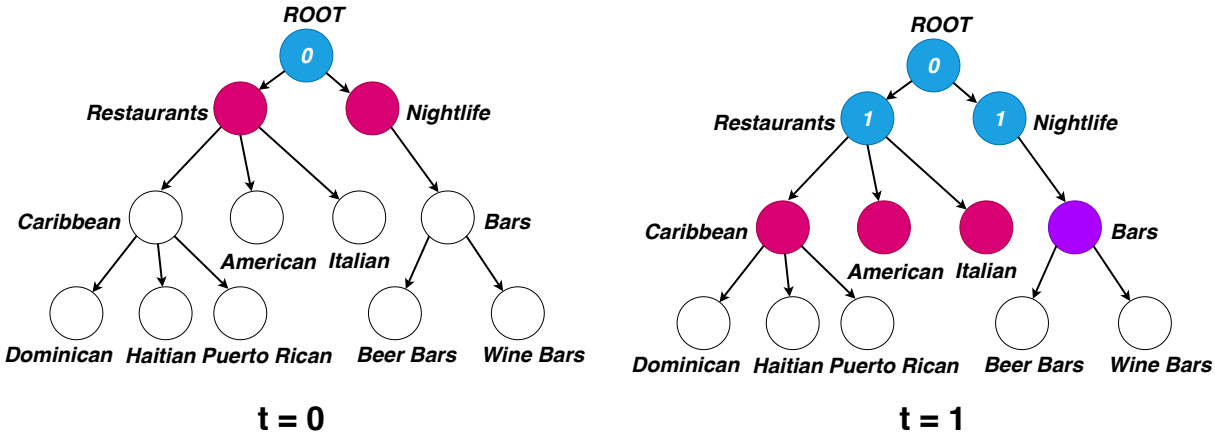

Figure 6: **One time step in HiLAP-SL.** At $t = 1$, two ($K = 2$) local per-parent probabilities $\mathbf{p}_1^{\text{Local}}$ are measured independently and aggregated in the loss function $\mathcal{O}_1$.