Large Scale Semantic Indexing with Deep Level-wise Extreme Multi-label Learning

Dingcheng Li, Jingyuan Zhang, Ping Li
Cognitive Computing Lab (CCL), Baidu Research
{lidingcheng,zhangjingyuan03,liping11}@baidu.com

ABSTRACT

Domain ontology is widely used to index literature for the convenience of literature retrieval. Due to the high cost of manual curation of key aspects from the scientific literature, automated methods are crucially required to assist the process of semantic indexing. However, it is a challenging task due to the huge amount of terms and complex hierarchical relations involved in a domain ontology. In this paper, in order to lessen the curse of dimensionality and enhance the training efficiency, we propose an approach named *Deep Level-wise Extreme Multi-label Learning and Classification (Deep Level-wise XMLC)*, to facilitate the semantic indexing of literatures. Specifically, Deep Level-wise XMLC is composed of two sequential modules. The first module, deep level-wise multi-label learning, decomposes the terms of a domain ontology into multiple levels and builds a special convolutional neural network for each level with category-dependent dynamic max pooling and macro F-measure based weights tuning. The second module, hierarchical pointer generation model merges the level-wise outputs into a final summarized semantic indexing. We demonstrate the effectiveness of Deep Level-wise XMLC by comparing it with several state-of-the-art methods on automatic labeling of MeSH, on literature from PubMed MEDLINE and automatic labeling of AmazonCat13K.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; **Transfer learning**; **Classification and regression trees**; **Neural networks**.

KEYWORDS

deep level-wise extreme multi-label learning and classification, online macro F-measure optimization, pointer generation

ACM Reference Format:
Dingcheng Li, Jingyuan Zhang, Ping Li. 2019. Large Scale Semantic Indexing with Deep Level-wise Extreme Multi-label Learning. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313636

1 INTRODUCTION

With the explosively growing amount of scientific literature, semantic indexing methods with high efficiency are required to build retrieval systems. Even with effective techniques, the semantic indexing process still involves manual curation of key aspects from scientific literature. In order to summarize the main topics of articles, domain experts are usually invited to manually index articles with keywords that are selected from the domain ontology.

In the medical field, MEDLINE [3, 19, 27] is perhaps the world largest biomedical literature database, and Medical Subject Headings (MeSH) is the domain ontology for indexing articles in MEDLINE. It has greatly improved the experience of medical literature search by mapping queries to MeSH headings. For example, the query *teen drug use* is mapped to the MeSH headings *Adolescent* and *Substance–Related Disorders*. Currently, most of the mapping rules as well as the final indexing of medical literature from MEDLINE are manually generated by domain experts. It is expensive and time-consuming for the human-labeling process of semantic indexing. Automated methods are therefore crucially desired.

The task of automated curation, however, faces significant challenges. First of all, an article is often labeled with multiple keywords or concepts. In addition, the domain ontology involves hundreds of thousands or even millions of labels. Those labels are typically organized in hierarchical structures that are represented in the form of a forest. It is a non-trivial task to simultaneously deal with massive labels, data samples and complex hierarchical structures.

In this paper, we consider the task of automated semantic indexing as an extreme multi-label learning and classification (XMLC) problem. Different from the traditional multi-class [15, 16, 34], XMLC allows for the co-existence of millions of labels for each data sample. Recently, several approaches are proposed to deal with XMLC, including FASTXML [29], LOMTrees [9], SLEEC [4], robust Bloom filters [10], label partitioning [33], fast label embeddings [25] and several deep learning methods, Hierarchical multi-label classification using local neural networks [7], DXML [20] and XML-CNN [37]. To a good extent, those methods have achieved good progress in handling XMLC. However, the curse of dimensionality (for which we refer to the huge label space) and the high demand of hand-crafted feature engineering are two major barriers for further improving the effectiveness and efficiency.

In order to address these two issues, we propose a novel framework named *Deep Level-wise Extreme Multi-label Learning and Classification (Deep Level-wise XMLC)* to deal with the problem of large scale semantic indexing. Deep Level-wise XMLC consists of two sequential modules. The first module is a level-wise multi-label classification model. It addresses the curse of dimensionality effectively by decomposing massive labels (in a higher dimensional space) into multiple levels (in a lower dimensional space). For each level, a convolutional neural network is constructed with two novel designs: One design is a category-based dynamic max pooling strategy aiming at capturing both label co-occurrences and categorical

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313636

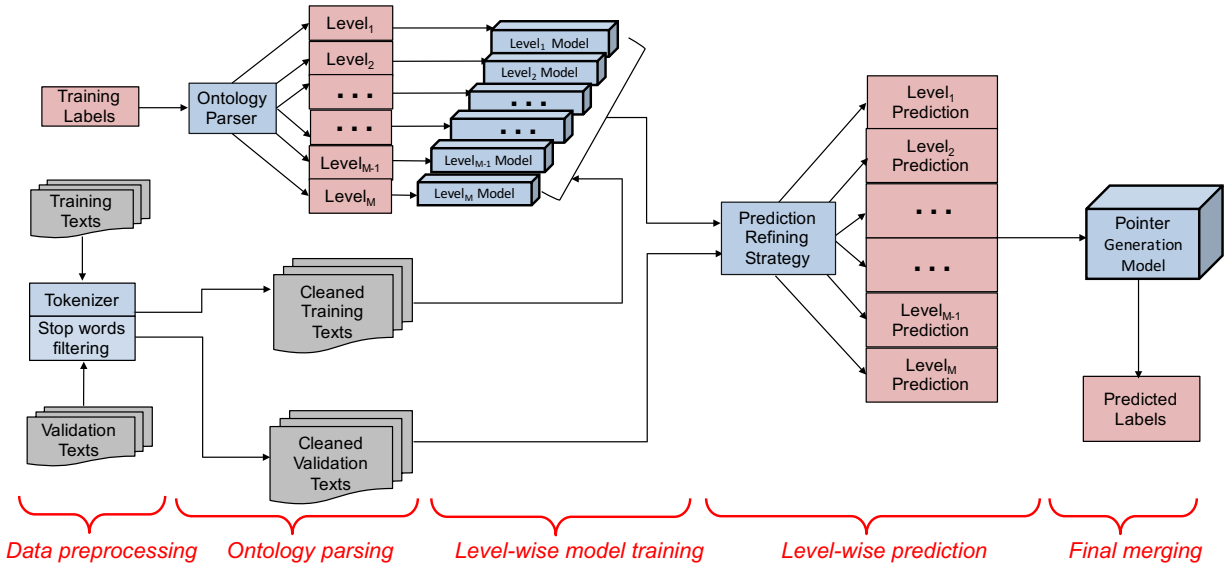


Figure 1: System architecture of the proposed framework. From left to right, there are five steps or blocks. In the data preprocessing stage, as usual, tokenization and filtering are made to obtain clean texts. Training and validation data are randomly selected according some proportions. Different from usual NLP tasks, an extra step of ontology parsing on the training labels is needed so that labels will be split into multiple levels based on their ontological hierarchies. In the third stage, the neural model described in the methodology section is employed to train level-wise models. Then, in the testing stage, testing data is fed into the trained level-wise models for label predictions or tagging after the testing data is preprocessed in a similar fashion. In the last stage, a final merging is made with a pointer generation model so that some less related labels are sifted out.

relations among labels. It helps connect the level-wise classification models tightly. The other design is a prediction refining strategy based on macro F-measure optimization, which enables the module to automatically select the labels in an incremental manner. The second module of Deep Level-wise XMLC is a hierarchical pointer generation model that merges predicted labels for each level into final summarized semantic indexing by the way of copying and generation mechanism [31]. As a whole, Deep Level-wise XMLC avoids the high cost of human interferences by learning semantic indexing without any feature engineering. We show the entire system architecture in Figure 1.

In summary, our major contributions are summarized as follows:

- We propose *Deep Level-wise XMLC* to learn large scale semantic indexing. It divides labels into multiple levels to lessen the curse of dimensionality while improving the training efficiency.
- We introduce a new strategy with category-dependent dynamic max pooling to capture both co-occurrences and categorical relations among labels.
- We explore a prediction refining technique derived from macro F-measure optimization to intelligently select the best labels in an online fashion.
- We develop a hierarchical pointer generation model to merge the level-wise outputs into the final summarized semantic indexing.
- We demonstrate the effectiveness of Deep Level-wise XMLC by comparing it with several state-of-the-art methods on automatic labeling of MeSH from MEDLINE, as well as AmazonCat13K [1], which is the XMLC dataset with similar nature as MeSH.

[1]https://manikvarma.github.io

2 METHODOLOGY

There are two primary challenges in XMLC. Firstly, the number of labels in one dataset can be more than 10,000 or even as large as one million. Secondly, one data sample can be indexed with multiple labels, the number typically ranging from one to several dozens.

In this paper, we propose Deep Level-wise XMLC to handle these two challenges by decomposing labels of each data sample into multiple levels. Five steps are involved in Deep Level-wise XMLC as shown in Figure 1. Firstly, the input raw texts for training and validations are processed with NLP preprocessors. Words are tokenized and stop words are filtered out. Secondly, the training labels are parsed into level-wise labels with ontology parsers. Thirdly, level-wise models are trained with Deep Level-wise XMLC. Fourthly, level-wise prediction is made on the datasets with refining strategies. Finally, a point generation model is trained to merge level-wise models into a unified label set.

The following subsections will focus on 1) our deep level-wise multi-label learning framework and 2) the pointer generation model to merge labels of all levels into one unified label set.

2.1 Deep level-wise multi-label learning

Formally, the problem can be defined as follows. Given a set of input pairs $\cup_{i=1}^{N}\{\mathbf{x}_i, \mathbf{y}_i\}$, Deep Level-wise XMLC decomposes them into M levels and trains M neural models on the training data. The whole label set is denoted as $\mathcal{L}$ and $|\mathcal{L}|$ refers to the total number of labels in $\mathcal{L}$. Each $\mathbf{y}_i$ is a multi-hot vector with length $|\mathcal{L}|$. Each model at level m predicts the most probable K labels, $\{\hat{y}_1^m, ..\hat{y}_j^m, ..., \hat{y}_K^m\}$ on each data sample. K is determined with a refining strategy. In the

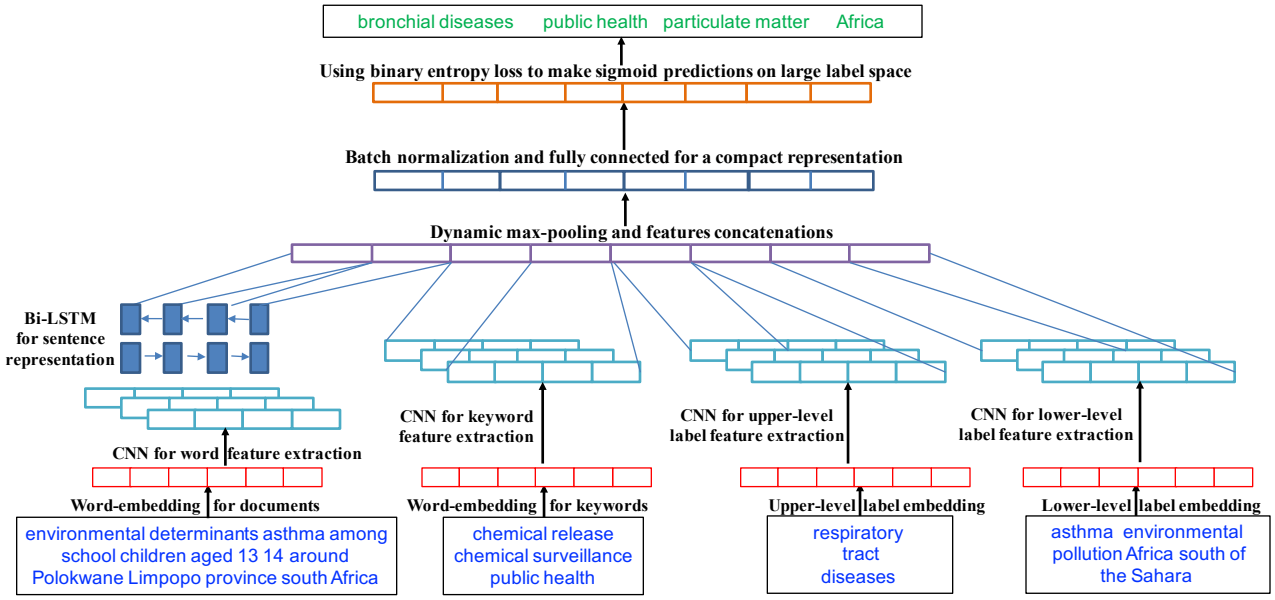


Figure 2: Neural structure of the proposed deep multi Label learning framework. As shown in this figure, the inputs involve four parts: word embedding for documents, word embedding for key words, upper-level and lower level embedding. Embeddings are pretrained with documents, keywords and labels for each document of the training data. Note that label embeddings are made with the same source, namely, the corresponding ontology. After the embedding representations are obtained, all of them are fed into CNN for feature extractions. In order to keep the order for documents, Bi-LSTM is employed for sentence representation. Then, dynamic max-pooling and batch normalization follow those feature extractions. Finally, binary entropy loss is computed to make sigmoid predictions on each label. We can also view this network structure as a multi-task learning.

end, a pointer generation model is trained to merge the predicted $\{\hat{\mathbf{y}}_i^1, ..., \hat{\mathbf{y}}_i^m, ..., \hat{\mathbf{y}}_i^M\}$ of M levels for each data sample $\mathbf{x}_i$ into one unified label set $\mathbf{y}_i$.

2.1.1 Feature embedding construction. We construct models in a level-wise manner. A neural model is built at each level with four parallel inputs as shown in Figure 2. The four inputs include documents, keywords and level-related information. They provide diverse information for the construction of more discriminative features. We employ CNNs to learn a rich number of feature representations. Document and keyword embedding are learned through CNN directly. The other two embeddings, upper-level and lower-level label embeddings, are learned from the embedding of prediction results from upper and lower levels. Specifically, two steps are involved. Firstly, similar to word embedding for input texts and keywords, we employ Gensim to train label embeddings from the annotated MeSH. Secondly, in both training and testing, predicted labels for some documents at some levels will be utilized as input features for their upper level or lower level. The two embeddings can not only help capture level-wise dependencies, but also deal with label imbalance issues in XMLC [4, 37]. In this way, both label co-occurrences and the knowledge from their upper and lower levels can help enhance the representation learning of rare labels.

For example, in MeSH, *Lymphangioma* is a rare label and it can not be easily represented by itself. With the information of its upper level MeSH, *Lymphatic Vessel Tumors* and lower level MeSH *Lymphangioma*, *Cystic Lymphangioma* can be better represented in the embedding space.

After the four embeddings are learned, they are concatenated and delivered into the max-pooling layer.

Due to the order information, raw tokens/words cannot be directly concatenated to the embeddings of keywords, upper and lower level labels. Hence, we construct a Bi-LSTM for raw tokens/words over their CNN features to keep the language order information [14] before concatenation.

2.1.2 The objective function of the learning framework. After the embedding concatenation, a dynamic max pooling layer, a binary cross-entropy loss over sigmoid output and a hidden bottleneck layer are employed. The loss function L of the binary cross-entropy objective is formulated as

$$L = \tag{1}$$

$$\underset{f}{\text{argmin}} -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{L}|} y_{ij}\log(\sigma(f_j(\mathbf{x}_i)) + (1-y_{ij})\log(1-\sigma(f_j(\mathbf{x}_i))),$$

where $\sigma(f_j(\mathbf{x}_i)) = \frac{1}{1+e^{-f_j(\mathbf{x}_i)}}$ and $f_j(\mathbf{x}_i)$ denote the output layer function. In addition, $f_j(\mathbf{x}_i) = \mathbf{w}_o g_h(\mathbf{w}_h[P(\mathbf{c}_1), ..., P(\mathbf{c}_\iota))])$. Here $\mathbf{w}_h \in R^{h\times(\iota p)}$ and $\mathbf{w}_o \in R^{|\mathcal{L}|\times h}$ are weight matrices associated with the bottleneck layer and output layer, g_h is the element-wise activation functions, e.g., *sigmoid* or *tanh* applied to the bottleneck layer and ιp is the product of ι and p at the dynamic max pooling layer. ι refers to the number of features fed into pooling layers and

p refers to pooling numbers. Both are determined by the number of features in $\mathbf{x_i}$. In addition, $\mathbf{c}_i$ is the vector of convolutional features after the pooling operation $P(.)$ from lower layers.

2.1.3 Categorical-oriented dynamic max pooling. In traditional CNN models for text classification, a max-over-time [11] scheme is often adopted, as intuitively the maximum element of a feature map should take the most important information, i.e., $P(\mathbf{c}) = \max\{\mathbf{c}\}$. This approach, however, exhibits a severe drawback. Using only one value to represent the whole feature map may miss information when the input document includes multiple topics. For multiple-label learning tasks, multiple pooling can capture richer information [20]. In this work, we dynamically perform pooling as $P(\mathbf{c}) = [\max\{\mathbf{c}_{(1:\frac{l}{p})}\}, ..., \max\{\mathbf{c}_{(l-\frac{l}{p}+1:l)}\}] \in R^p$, where $\mathbf{c}_{(1:\frac{l}{p})}$ refers to the sub-vector of $\mathbf{c}$ starting from index 1 to $\frac{l}{p}$. Previous work used a fixed p. If p is set too large, redundant features may be included. If it is set too small, relevant features may be missing.

In our work, level-wise related information, i.e., categorical information of labels is incorporated into the neural structures to help select p dynamically. Specifically, we tune p with the distribution of the label levels. For example, in MeSH, all terms are divided into 16 categories, like *Anatomy*, *Organisms*, *Diseases* and etc. Each category involves diverse subcategories and each label involves different distributions. Based on the distribution, we assign different weights to determine the p. The larger the weight of the category is, the larger the p is. The weight of the category or the label is initialized from the training data.

2.1.4 Refining predictions with macro F-measure maximization. With embeddings and dynamic max pooling, the network can make level-wise predictions as shown in Figure 1. At each level, we select the top-K predicted labels for each data sample. However, a fixed K may yield high recall but low precision. In this work, we refine the predictions with a more flexible weight adjustment strategy.

We apply the online F-measure optimization (OFO) [6, 13] for the weight adjustment. With OFO, a dynamic balance of precision and recall can be achieved. Formally, the OFO algorithm optimizes the binary F-measure through threshold tuning in an online fashion.

$$F_{ij}^t = \frac{2\sum_{l=1}^{i} y_{lj}^t \hat{y}_{lj}^t}{\sum_{l=1}^{i} y_{lj}^t + \sum_{l=1}^{i} \hat{y}_{li}^t} = \frac{2\alpha_{ij}^t}{\beta_{ij}^t}, \tag{2}$$

with $\alpha_{ij}^t = \sum_{l=1}^{i} y_{lj}^t \hat{y}_{lj}^t$ and $\beta_{ij}^t = \sum_{l=1}^{i} y_{lj}^t + \sum_{l=1}^{i} \hat{y}_{li}^t$. Here y_{lj}^t is the j-th label of the l-th data sample. F_{ij}^t is the accumulated F-score from the first to the i-th data sample on label y_j at iteration t.

Due to the incremental property, the threshold of OFO is updated by two rules. At the same iteration, the threshold λ_{ij}^t is updated as $\lambda_{ij}^t = \alpha_{(i-1)j}^t / \beta_{(i-1)j}^t$. At different iterations, it is updated as $\lambda_{ij}^t = \alpha_{Nj}^{(t-1)} / \beta_{Nj}^{(t-1)}$. Given the i-th data sample, OFO refines predicted labels as $\hat{y}_{ij}^t = \left[\hat{\eta}(\mathbf{x}_{ij}^t) > \lambda_{ij}^t\right]$, where $\hat{\eta}(\mathbf{x}_{ij}^t)$ refers to the prediction probability of $\mathbf{x}_i$ on label y_j at iteration t. The optimal F-measure $\hat{F}(\lambda)$ is twice the value of the optimal threshold $\hat{\lambda}$ as $\hat{F}(\lambda) = 2\hat{\lambda}$. Since our refining mechanism is dynamic, level-wise and incremental, the optimal threshold $\hat{\lambda}$ will not be fixed until the end of training. It will be saved as a parameter for testing.

2.2 Pointer generation model for final merging

After we have level-wise outputs, we need to merge those outputs into one unified label set. However, we cannot simply combine them together because a simple concatenation will lead to a much larger number of labels than the gold standard labels. In this paper, we propose a filtering method to remove some level-wise labels to make sure that the final distributions of predicted labels are consistent with the gold standard ones. Inspired by the text summarization [23, 31], we treat each level-wise predication as one sentence and the gold standard as a summarized output. We take into consideration the hierarchical relations of labels among levels during decoding, encoding and attention states [23, 31].

2.2.1 Hierarchical pointer generation model. Inspired by [31], we design our hierarchical pointer generation model by allowing both copying labels from the level-wise predictions and generating labels from the whole label set.

The model is composed of five parts as in Figure 3. The first part, i.e., the input, is about the level-wise predictions from level 1 to level M. In the second part, the input is encoded to M sequences of hidden states. Each encoded hidden state reflects the inner relations of the predicted labels at a certain level. We represent the *encoder hidden state* as $\mathbf{e}^\tau = \mathbf{v}^T \tanh(\mathbf{w}_h \gamma^\tau + \mathbf{w}_s \mathbf{s}^\tau + b_{attn})$. Here $\mathbf{s}^\tau$ and γ^τ are the predicted *label sequence vector* and the *context vector* surrounding the predicted labels, respectively. The terms $\mathbf{v}$, $\mathbf{w}_h$, $\mathbf{w}_s$ and b_{attn} are weight parameters. The *context vector* is about the co-occurrences of labels.

In the third part, *attention generators* are derived from the *encoder hidden state* to generate an attention distribution $\mathbf{a}^\tau$ and a context vector γ^τ at time step τ. $\mathbf{a}^\tau$ is calculated as $\mathbf{a}^\tau = softmax(\mathbf{e}^\tau)$. The attention distribution is a probability distribution over the predicted level-wise labels. It is used to produce γ^τ as a hierarchical weighted sum of the *encoder hidden states*: $\gamma^\tau = \sum_q \mathbf{w}_q \mathbf{a}_q^\tau \gamma_q$.

In Figure 3, each attention generator is named as a coverage vector, which shows how much focus is given to labels of each level. As is known, summarizations may lead to repetitions. Hence, the same label may be generated more than once as well. The well-designed coverage vector plays the role of judging whether the label is a duplicate or not. If not a duplicate, the label with a high attention has more chance of being decoded as one right label. If a duplicate, the mechanism at 2.2.3 will filter the label out.

In the fourth part, i.e., the *decoder hidden states*, the generation probability $p_{gen} \in [0, 1]$ for time step τ is calculated from the context vector γ_τ, the encoder state $\mathbf{s}_\tau$ and the decoder input $\mathbf{y}^\tau$ as:

$$p_{gen} = \sigma(\mathbf{w}_h \gamma^\tau + \mathbf{w}_s \mathbf{s}^\tau + \mathbf{w}_y \mathbf{y}^\tau + b_{ptr}), \tag{3}$$

where $\mathbf{w}_h$, $\mathbf{w}_s$, $\mathbf{w}_y$ and b_{ptr} are weight parameters. Here p_{gen} is used as a soft switch to choose between *generating* a label from the whole label set by *sampling* from the label distribution, $p_{\mathcal{L}}$ (see how $p_{\mathcal{L}}$ is calculated in 2.2.2) or *copying* a label from the input sequences by sampling from the attention distribution $\mathbf{a}^\tau$.

With the above four parts, the proposed hierarchical pointer generation model can be trained to generate the final summarized semantic indexing. In the fifth part, we learn the probability of generating the final labels. Given a training pair $(\cup_{m=1}^{M} \mathbf{y}^m, \mathbf{y})$, we compute the conditional probability $p(\cup_{m=1}^{M} \mathbf{y} | y, \theta)$ to estimate the

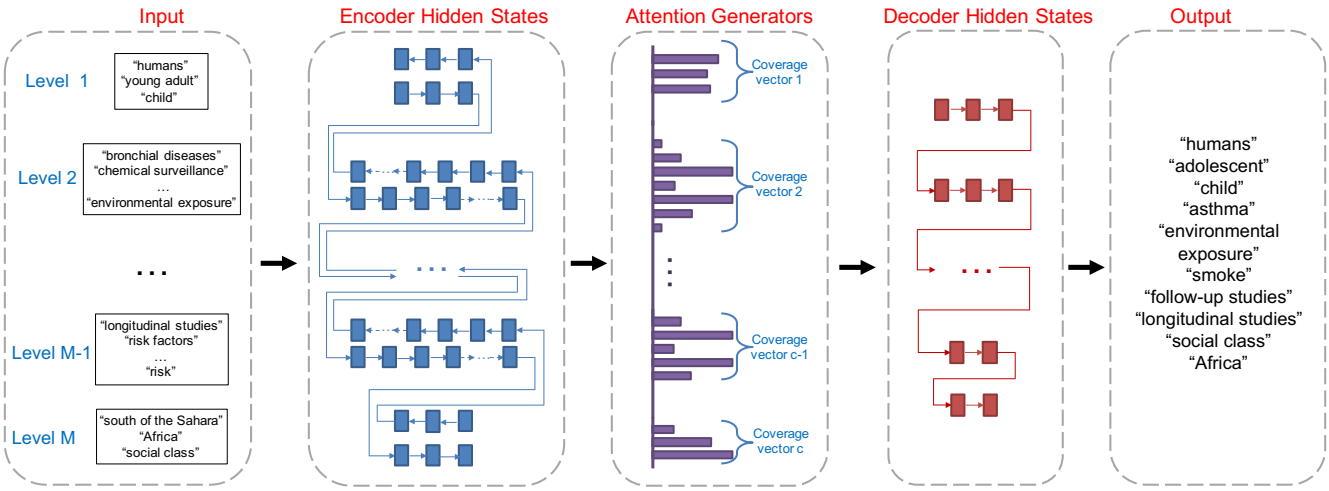


Figure 3: Pointer generation model for final merging. This step follows the level-wise neural structure learning. Five components can be seen from left to right. The input is the predicted level-wise labels. They are organized as sequences level by level. Then, they are encoded into hidden states. Based on coverage vectors described in the methodology, attentions are generated. Then, decoder is working to generate outputs with reduced sizes.

labels of the terms of the probability chain rule as

$$p(\mathbf{y}|\cup_{m=1}^{M}\mathbf{y}^m;\theta)=\prod_{\delta=1}^{|\mathcal{L}|}p_\theta(y_\delta|y_1,...,y_{\delta-1};\theta), \tag{4}$$

where $\mathbf{y}=\{\mathbf{y}_1,...,\mathbf{y}_{|\mathcal{L}|}\}$ is a sequence of $|\mathcal{L}|$ vectors. The parameters of the model are learned by maximizing the conditional probabilities for the training set as $\theta=\underset{\theta}{\text{argmax}}\sum_{\mathbf{y},\cup_{m=1}^{M}}\log p(\mathbf{y}|\cup_{m=1}^{M}\mathbf{y}^m;\theta)$, where the sum is over training examples.

2.2.2 Sequence-to-sequence probability calculation. The above procedure finally will produce the label vocabulary distribution $p_{\mathcal{L}}$:

$$p_{\mathcal{L}}=softmax(\mathbf{v}'(\mathbf{v}[s_t,h_{*t}]+b)+b'), \tag{5}$$

where $\mathbf{v}$, $\mathbf{v}'$, b and b' are learnable parameters. For a specific label, we can obtain from $p(y_{ij})=p_{\mathcal{L}}(y_{ij})$. The loss function is the negative log likelihood of the target label $\hat{y}_{ij}$. The following example illustrates the procedure of probability calculations for one label given other labels.

Suppose we have a label set $\{Nutritional\ and\ Metabolic\ Diseases\}$. The hierarchical relations are shown in Figure 4. The left words in the figure are acronyms for the labels. For example, $Wolfram\ Syndrome$ is shortened with their initial letters as ws.

Given $context=\{nmd,md,dm,dmt1\}$, we follow the hierarchical relations among those labels to calculate $p(\mathbf{e}_{ws}|context)$ as

$$p(\mathbf{e}_{ws}|context)=(1-\sigma(b_{ptr}+\mathbf{e}_{nmd}^T\mathbf{h}_{nmd})\sigma(b_{ptr}+\mathbf{e}_{md}^T\mathbf{h}_{md})$$
$$\sigma(b_{ptr}+\mathbf{e}_{dm}^T\mathbf{h}_{dm})\sigma(b_{ptr}+\mathbf{e}_{dmt1}^T\mathbf{h}_{dm1})). \tag{6}$$

2.2.3 Mechanism to avoid duplications. A problem for pointer generation model or sequence-to-sequence model is that it may copy terms from input multiple times [24, 30]. In our task, we do not need any repetitive terms since each label should be unique. Previous works [31, 32] adapts a coverage mechanism to avoid repetition. Namely, if labels have been seen in the output, the probability of generating them will become low. We take similar approaches by combining a coverage mechanism into the whole pointer generation model. Specifically,

$$\varkappa^m=\sum_{\mathbf{y}^m=0}^{|\varkappa|}\varkappa^{\mathbf{y}^m}, \tag{7}$$

where, $\varkappa$ refers to a coverage vector, $\mathbf{y}^m$ refers to m-th level.

Our coverage vectors are composed of a set of vectors for all levels. For each coverage vector, $\varkappa^{\mathbf{y}^m}$ is an unnormalized distribution over the level-wise inputs that represents the degree of coverage that those labels have received from the attention mechanism so far. Since we order labels with levels, there should be no repetitions at different sections of levels, this mechanism aims at removing all duplicate labels found in different sections and also avoid duplicates within the same level. $w_\varkappa\varkappa^{\mathbf{y}^m}$ is added to the attention mechanism and a $covloss_{\mathbf{y}^m}$ is also added to the total loss function of the pointer generation model as the penalty for any duplications.

3 EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness of our Deep Level-wise XMLC with MEDLINE dataset labeled with MeSH and AmazonCat13K. As described in Section 1, MEDLINE is the largest biomedical literature database in the world and Medical Subject Headings (MeSH) is the domain ontology for tagging articles in MEDLINE. The other dataset we use, AmazonCat13K, is one of benchmark datasets for developing extreme classification algorithms. It involves 13330 labels, with all labels hierarchically organized, similar to MeSH. The dataset scale, the expert labeling and the hierarchical nature provide a perfect testbed for our proposed framework.

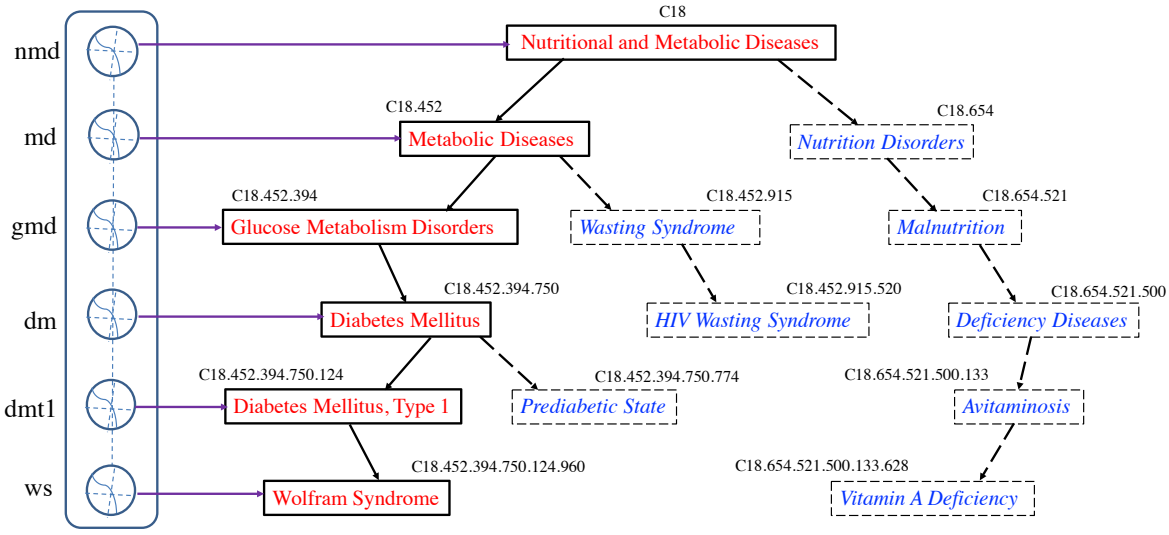


Figure 4: Example of sequence-to-sequence probability calculation. The calculation of the example is described in section 2.2.2. A little bit clarification may be necessary is on those short forms. On the left, for space saving and for better illustration of the process, Sigmoid symbols are drawn there with those short forms, which are the initial letters of the mesh terms.

3.1 Data setting and preprocessing

The total number of MeSH labels in MEDLINE is 26,000, among which 60% appear more than 1000 times. We remove those MeSH labels occurring less than 10 times in the experiment. The MEDLINE has 26 million articles with abstracts. 90% of these articles have about 20 MeSH labels. 4 to 16 MeSH labels are assigned to 82% of articles. In MeSH, 3.5 million abstracts have both MeSH labels and keywords. The ontology of MeSH labels can be decomposed into 7 levels, where the lowest level (the 7th level) includes the most specific MeSH labels while the highest level (the 1st level) has the most general and abstract MeSH labels. For articles with only MeSH labels of the lowest level, we expand them by the following method. Starting from labels at the lowest level, we find out all labels of their upper levels. We construct 7 datasets for the proposed Deep Level-wise XMLC framework.

Meanwhile, 102,167 abstracts with MeSH labels from all the 7 levels are put aside for testing. The statistics of the dataset at each level is shown in Table 1. It can be observed that the middle levels have the largest number of labels while the highest level has only 83 labels and the lowest level has 2445 labels. Similar trend can been found for data volumes. Two million articles have labels from level 2, 3 and 4 while less than one million articles have labels from level 1, 6 and 7.

For AmazonCat13K, we cannot directly use their preprocessed dataset since Deep Level-wise XMLC requires text data. Meanwhile, we need divide the data based on their level-wise categories. It is found that all labels can be decomposed into 9 levels. Somewhat differently, if a document from AmazonCat13K has lower labels, it must have higher labels while a document from MeSH is not necessarily so. Therefore, it is straightforward to find a common set for testing for AmazonCat13K (simply use documents with lower categories). In order to keep a reasonable pool of testing data, we ignore documents which have levels higher than 6 (only 9990, 385 and 31 documents for level 7,8, and 9, respectively).

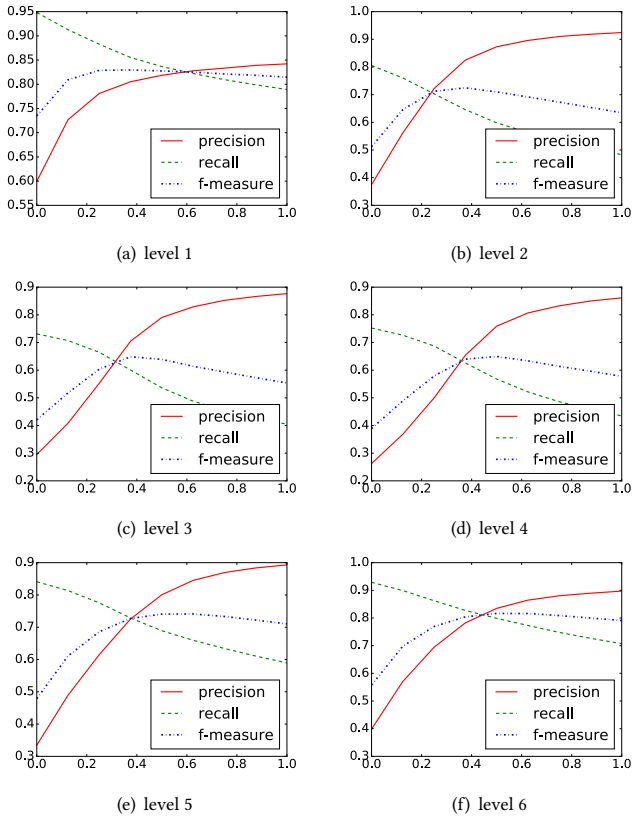
Table 1: The statistics of the datasets. For each level, there are different data volumes. Papers in Medline, do not necessarily imply that they can be tagged higher level MeSH terms even if they are tagged with lower level MeSH terms.

Levels	Data Volumes		The number of labels	
	MeSH	AmazonCat13K	MeSH	AmazonCat13K
Level 1	969,233	858,795	83	44
Level 2	2,444,854	812,249	1,382	362
Level 3	2,405,321	549,326	4,484	2,281
Level 4	2,182,885	427,378	6,568	6,181
Level 5	1,522,195	178,042	5,750	5,372
Level 6	906,873	71,041	3,895	1,998
Level 7	402,794	9,990	2,445	441

In the experiments, for the MEDLINE articles and keywords, at each level, we first train an individual neural network according to the first component of Deep Level-wise XMLC. The trained model is employed to make predictions on the testing data for each level. Then the predicted level-wise labels as well as the gold standard labels from the training data are utilized by the pointer generation model for the final merging. Likewise, we train level-wise model for AmazonCat14K except that the latter do not have keywords.

3.2 Evaluation metrics

In extreme multi-label classification datasets, even though there are usually huge label spaces, only limited number of relevant labels for each document. This means that it is important to present a short ranked list of relevant labels for each test document. The



(a) level 1 (b) level 2

(c) level 3 (d) level 4

(e) level 5 (f) level 6

Figure 5: Macro precision, recall and F-measure obtained by OFO with Deep Level-wise XMLC for the first six levels.

evaluation thus focuses on the quality of such ranked lists with emphasis on the relevance of the top portion of each list. In our work, however, we use two evaluation metrics for the purpose of comparisons with the two sources of datasets. The medical community prefers to use precision, recall and F-score while those from the general domains prefer precision at K (P@K) and the Normalized Discounted Cummulated Gains (NDCG@K or G@K for short).

Specifically, given a predicted label list $\mathbf{y}_1^m = \{y_1, y_2, ..., y_K\}$ with top K items at level m, precision, recall and F-score are defined as follows,

$$MiP = \frac{\sum_{i=1}^{N} c(K, i, y_1^K)}{\sum_{i=1}^{N} K_i}, \qquad MaP = \frac{1}{N}\sum_{i=1}^{N} \frac{c(K, i, y_1^K)}{K_i}, \tag{8}$$

$$MiR = \frac{\sum_{i=1}^{N} c(K, i, y_1^K)}{\sum_{i=1}^{N} AK_i}, \qquad MaR = \frac{1}{N}\sum_{i=1}^{N} \frac{c(K, i, y_1^K)}{AK_i}, \tag{9}$$

$$MiF = \frac{2 * MiP * MiR}{MiP + MiR}, \qquad MaF = \frac{2 * MaP * MaR}{MaP + MaR}, \tag{10}$$

where N is the number of data samples and $c(K, i, y_1^K)$ is the number of correct labels among the top K ranked labels; AK_i is the total number of the gold standard labels for article i; The difference between micro measures and macro measures lies in the calculation of the predicted probabilities. For micro measures, the probability calculation is not done until all correct predictions are added together, while for macro measures, the probability calculation will be done for each article and in the end, an average is used as the macro scores. We report both measures in order to see how accurate the model is for a single article and for an entire dataset.

In contrast, the definition of P@K and NDCG@K is,

$$P@k = \frac{1}{k}\sum_{l \in r_k(\hat{\mathbf{y}})} y_l \tag{11}$$

$$DCG@k = \sum_{l \in r_k(\hat{\mathbf{y}})} \frac{y_l}{log(l+1)} \tag{12}$$

$$NDCG@k = \frac{DCG@k}{\sum_{l=1}^{min(k, \|\vec{y}\|)} \frac{y_l}{log(l+1)}} \tag{13}$$

where $\vec{y} \in \{0, 1\}^L$ is denoted as the vector of true labels of a document and $\hat{\vec{y}} \in R^L$ as the system-predicted score vector for the same document. We use $k = 1, 3, 5$, following the convention of P@K and NDCG@K.

3.3 Parameter settings

For the neural network of Deep Level-wise XMLC, we use the rectified linear units. The filtering windows are set to 3, 4, 5. The dropout rate is set to 0.5 and the L2 constraint is set to 3. The mini-batch size is set to 256. The embedding dimensions vary for different features. For Mesh, word embedding for medical words involves 500,000 unique tokens, keyword embedding involves over 100,000 phrases and label embedding 26,000 MeSH terms. Gensim is employed to train the embedding with 300 as the dimension. For AmazonCat13K, pretrained GoogleNews-vectors-negative300.bin is utilized with 3 million tokens and 300 as the dimension. The values for other hyperparameters are chosen via a grid search on a smaller validation set from the training data.

3.4 Performance with online F-measure optimization

As discussed in Section 2.1.4, the online macro F-measure optimization (OFO) is integrated into the proposed framework. In order to show the effectiveness of OFO, we report the macro precision, recall and F-score for the first 6 levels in Figure 5 for MeSH. Due to space limit, we do not show the result of the 7th level and that of AmazonCat13K in this paper. Similar performances can be obtained for them. We can observe that OFO helps achieve a balance between macro precision and recall. It is further observed that the optimal F-score is different at different levels. If we always select the top K ($k = 10$ in our experiment) for the level-wise prediction, we cannot obtain the best F-score though the recall at each level can be as high as around 80%. The precision can be as low as or less than 20%. The reason is that after we divide MeSH labels of each article into 7 levels, most of articles have only 2 to 5 labels at each level. This means that even if all of labels are within the top 10, the precision is only from 20% to 50% although the recall can be 100%. In this case, the F-scores are not high either. The OFO greatly removes less relevant labels so that the number of labels in the final prediction set of each level ranges from 2 to 5 as well. Meanwhile, most of

Table 2: The level-wise performance for micro measures with Top K and OFO. This table aims at showing the incremental improvements on the micro measures when the new features added stepwise for each level. Meanwhile, for each level, results for TopK without optimization and with optimization as OFO are shown there as well.

Embeddings	Levels	Level 1		Level 2		Level 3		Level 4		Level 5		Level 6		Level 7	
MEDLINE Collections	Measures	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO
	MiP	23.32	55.83	29.32	90.79	22.96	79.82	20.63	76.80	20.15	46.45	22.18	55.40	20.29	66.63
	MiR	95.68	84.77	65.17	37.57	38.32	35.10	37.21	37.65	49.85	62.31	72.24	73.42	83.64	82.34
	MiF	37.50	67.32	40.44	53.15	28.72	48.75	26.54	50.53	28.70	53.22	33.93	63.15	32.66	73.66
MEDLINE Collections & keywords		Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO
	MiP	31.45	72.45	36.53	88.10	31.56	52.90	29.36	76.80	31.25	84.81	32.88	69.43	33.43	77.19
	MiR	92.56	79.56	66.72	43.03	58.53	51.09	46.31	37.65	52.75	45.11	74.43	69.18	86.48	80.63
	MiF	46.95	75.84	47.21	57.82	41.00	51.98	35.94	50.53	39.24	58.90	45.61	69.30	48.21	78.87
MEDLINE Collections & keywords & upper & lower labels		Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO
	MiP	39.52	81.01	43.64	69.95	40.65	68.63	38.45	63.62	40.33	73.31	41.94	76.64	45.38	84.94
	MiR	86.75	73.93	63.27	56.05	55.33	43.70	52.31	46.76	51.14	53.07	69.45	65.88	77.36	77.30
	MiF	54.30	**77.33**	51.65	**62.23**	46.87	**53.40**	44.32	**53.90**	45.09	**61.57**	52.30	**70.85**	57.20	**80.94**
AmazonCat13K		Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO
	MiP	72.21	94.17	48.57	61.49	30.43	32.51	20.99	21.34	20.37	20.88				
	MiR	99.56	92.24	97.42	99.55	87.47	99.86	72.03	99.92	69.24	99.91				
	MiF	83.71	**93.19**	64.82	**76.02**	45.15	**49.05**	32.51	**35.17**	31.48	**34.54**				

Table 3: The level-wise performance for macro measures with Top K and OFO. This table aims at showing the incremental improvements on the micro measures when the new features added stepwise for each level. Meanwhile, in a smilar fashion to Table 2, for each level, results for TopK without optimization and with optimization as OFO are shown there as well.

Embeddings	Levels	Level 1		Level 2		Level 3		Level 4		Level 5		Level 6		Level 7	
MEDLINE Collections	Measures	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO
	MaP	23.32	59.96	29.32	56.18	22.96	81.40	20.63	36.84	20.15	48.95	22.18	69.94	20.29	78.73
	MaR	96.26	94.83	65.17	76.03	47.00	43.99	44.18	72.64	56.18	81.53	75.90	48.65	85.73	65.01
	MaF	37.55	73.47	40.44	64.61	30.85	57.11	28.13	48.89	29.66	61.12	34.32	57.38	32.81	71.22
MEDLINE Collections & keywords		Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO
	MaP	33.54	92.56	37.69	85.33	33.84	82.88	32.63	80.68	34.52	61.53	33.92	69.44	36.34	76.97
	MaR	91.35	70.23	68.27	55.80	61.33	48.79	48.94	52.16	55.77	77.55	72.94	86.25	82.88	92.67
	MaF	49.07	79.87	48.57	67.48	43.61	61.42	39.15	63.36	42.64	68.62	42.64	79.63	46.30	84.09
MEDLINE Collections & keywords & upper & lower labels		Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO	Top K	OFO
	MaP	42.33	80.53	45.46	82.52	41.46	70.48	38.99	75.90	43.49	84.52	43.49	86.46	49.34	92.09
	MaR	87.77	85.53	62.37	64.59	58.34	60.06	52.35	56.80	52.35	65.96	70.44	77.32	78.63	83.58
	MaF	57.11	**82.96**	52.59	**72.46**	48.47	**64.85**	46.05	**64.97**	44.69	**74.10**	53.78	**81.64**	60.63	**87.63**

the correct predictions are still kept in the prediction set. Evidently, this tuning strategy greatly boosts the performance.

3.5 Level-wise Performance

As discussed in Section 2, our framework decomposes the task of XMLC into level-wise model constructions. Therefore, in this section, we report level-wise prediction results in order to see the intermediate developments and improvements of the whole model.

As is shown in Figure 2, the level-wise neural models learn label embeddings from the MEDLINE collections, keywords and the predicted labels from the upper or lower levels. Here we report the performance of the level-wise neural models with different embeddings. We further show the effectiveness of OFO by comparing with the level-wise neural models by fixing the top K labels for each level. We test on different K from 1 to 10 and find that the best performance is achieved when K is 5.

Table 2 reports the micro performance for the level-wise model with OFO and top K fixing strategies. Here K is set to 5 for the best results. We also show the performance for the macro measurement in Table 3. We can see that OFO always perform better than the strategy of fixing top K, no matter on the micro or the macro measurements.

Table 2 and Table 3 also report the level-wise prediction with three different embeddings for MeSH. Although the evaluation for AmazonCat13K dataset is not based on F-scores, we also report micro measures for AmazonCat13K to show the advantages of OFO. After all, the result of P@K and NDCG@K for it is computed on the filtered output with OFO. From this result, we can see an evident incremental trend for all seven levels. Namely, with keywords and predicted MeSH terms of upper and lower levels added, the prediction sees quick improvements accordingly. It is not hard to see that in general, macro results are better than micro results. Among

them, the third level and the fourth level of MeSH while the fourth and the fifth level of AmazonCat13K yield worse results than others while level 1 obtain much better results for both datasets. This is understandable considering the larger number of labels for third and the fourth (4,484 and 6,568 for MeSH while 6,181 and 5,372 for AmazonCat13K respectively).

3.6 Performance of final merging

The proposed Deep Level-wise XMLC will merge the level-wise predictions into one unified label set with a pointer generation model. In this section, we further compare with five state-of-the-art approaches to demonstrate the effectiveness of the pointer generation model, including MTIDEF [12], MeSH Now [22], MeSHLabeler, MeSHRanker [21] and Deep Mesh [28] for MeSH results. All these existing systems make heavy use of feature engineering. In contrast, Deep Level-wise XMLC uses limited external resources. For AmazonCat13K, we report results of XML-CNN, the state of the art systems on this benchmark dataset.

Let us start from MeSH labeling. After we achieve the level-wise results, the hierarchical pointer generation model is trained with predicted results from all levels as the input and the gold standard labels as the output. For model training, the inputs can be organized with each label as an independent unit or with labels of the same level as one unit (known as *sentence* in the summarization community). Hence, two pointer generation models are trained, with the former known as Deep Level-wise XMLC$_{label}$ and with the latter as Deep Level-wise XMLC$_{level}$. For comparison, we also report results which adds results of all levels together and then filter less relevant labels by their prediction probabilities and by the label distributions in the gold standard (Deep Level-wise XMLC$_{sampling}$).

Table 4: Performance of Deep Level-wise XMLC for MeSH dataset. From the bold numbers, we can see that the best performances come from Deep Level-wise XMLC. It is obvious to see that level based and dynamic pooling obtain better performance than label based and dynamic pooling.

Methods	MaP	MaR	MaF
No Level-wise XMLC			
MTIDEF	49.39	51.40	50.37
BC_D2V-TFIDF	47.41	46.33	46.86
MeSHRanker	53.64	54.13	53.89
MeSHLabeler	54.50	51.72	50.54
DeepMesh	53.80	55.05	54.42
MeSH Now	51.28	53.72	52.47
Deep Level-wise XMLC without pointer generation network			
Sampling	49.56	51.21	50.37
Max pooling	51.56	65.52	57.70
Deep Level-wise XMLC with pointer generation network			
Dynamic pooling & label-based	**61.20**	57.21	59.14
Dynamic pooling & level-based	53.22	**70.65**	**60.77**

As shown in Table 4, both Deep Level-wise XMLC$_{label}$ and Deep Level-wise XMLC$_{level}$ outperform other systems much on macro measures in precisions, recalls as well as F-scores. Due to the page limit, we do not report the micro measures, which have similar trends.

By involving embeddings from MEDLINE collections and keywords, Deep Level-wise XMLC$_{label}$ and Deep Level-wise XMLC$_{level}$ achieve much better performances than all other existing cutting edge frameworks. To our interest, it seems that different organizations of the inputs will lead to different performances in precision and recall although F scores are quite similar. Deep Level-wise XMLC$_{label}$ achieves better precision while Deep Level-wise XMLC$_{level}$ better recall. This seems to indicate that our hierarchical pointer generation model takes into considerations the correlations between labels within the unit. Therefore, Deep Level-wise XMLC$_{level}$, which has longer input unit, obtains better recall. Yet, it also includes more false positives, thus reducing its precision. In contrast, Deep Level-wise XMLC$_{label}$ wins in precision probably it considers more smaller units and then misses more true positives.

Meanwhile, we can see that Deep Level-wise XMLC$_{sampling}$ obtains much poorer results than most of existing systems. This shows that the hierarchical pointer generation model plays essential roles in reaching the optimal performances in the end. Besides, we also report results of Deep Level-wise XMLC$_{level}$ with max pooling. By default, all of our systems work with dynamic max pooling. Evidently, the result shows that dynamic max pooling gains advantages over the usual max pooling strategies.

Table 5: Performance of Deep Level-wise XMLC for AmazonCat13K. We use the version with pointer generation network and dynamic pooling & level-based. As stated in 3.1, in order to extend our methodology from medical field to more general ones, we test our model on AmazonCat13K as well. For those who use AmazonCat13K, they prefer reporting precision@K and NDCG@K. We also list the performance of XML-CNN for AmazonCat13K for comparisons.

	P@1	P@3	P@5	G@1	G@3	G@5
XML-CNN	95.06	79.86	63.91	95.06	89.48	87.06
Deep Level-wise XMLC	96.52	83.72	67.89	97.48	92.32	87.52

For AmazonCat13K, the result is given in Table 5. We also copy the state of the art results from XML-CNN from their work [20] in Table 5. We can see higher results from our work. Due to the differences of the testing data, we cannot tell which work is better from those results (our testing dataset are extracted from the raw text data with labels of each level for better evaluations). Their work is tested on the standard test dataset prepared by data collectors. Yet, it at least shows that our model can achieve results as comparable to the best model.

4 RELATED WORK

4.1 Tree-based methods

Due to the huge number of labels, the prediction of XMLC may involve high cost in both time and space. Tree-based methods make efforts to reduce both training and testing cost. For example, the label partitioning by sub-linear ranking (LPSR) method attempts to reduce the prediction time by learning a hierarchy over a base

classifier [17, 18, 33]. Argrawal et al., 2013 [1] proposes a method as the multi-label random forest (MLRF), which seeks to learn an ensemble of randomized trees instead of relying on the learning of a base classifier. FastXML [29] is proposed to learn a hierarchy not over the label space but over the feature space. It defines the set of labels active in a region to be the union of the labels of all training points present in that region. At each node of the hierarchy, an NDCG-based objective is optimized. Namely, at each node, a hyperplane is induced and it splits the set of documents in the current node into two subsets. Predictions are made by returning the ranked list of the most frequently occurring labels in all the leaf nodes. Recently, Mu et al [26] develops multi-label classifications for social streams based on ensemble random forests. It integrates a base learner and a label-based learner to learn hierarchical labels. However, these approaches suffer from high cost of training due to the dimensionality of both label space and feature space.

4.2 Embedding methods

Embedding methods attempt to overcome the intractability issue brought by the huge number of labels by projecting label vectors onto a low dimensional space and thus reducing the number of labels. The assumption is that the label matrix is low-rank. Due to its strong theoretical foundations and the ability to handle label correlations, embedding methods have proved to be the most popular approach for tackling XMLC problems [2, 5, 8, 10]. In particular, the recently proposed embedding method SLEEC [4] greatly increases the accuracy after they incorporates the non-linear neighborhood constraints in the low-dimensional embedding space for training and use a simple k-nearest neighbor (k-NN) clustering in the embedding space for testing. In our approach, we take steps further by exploring level-wise label embedding to improve the predictions of our neural structure.

4.3 Max-margin method

Max-margin method is also employed to handle multi-label classification. Yen et al [36] proposes a model named as PD-Sparse. Essentially, a linear classifier is learned for each label with $L1$ and $L2$ norm penalty on the weight matrix associated with this label. This results in a sparse solution in both the primal and dual spaces. Their fully-Corrective Block-Coordinate Frank-Wolfe training algorithm achieves sub-linear training time w.r.t the number of primal and dual variables while getting better performance than 1-vs-all SVM and logistic regression on multi-label classification, with significantly reduced training time and model size. However, same as 1-vs-all SVM, the method is algorithmically not scalable to extreme multi-label learning.

4.4 Deep learning based method

Deep learning based method has also been used for multi-label learning [20, 37]. Zhang et al. [37] incorporate label space embedding into feature embedding. Specifically, they construct an adjacency matrix for labels A and derive the label graph matrix with the equation $M = (A + A^2)/2$. Then, for each nonzero entry in the matrix, a tuple composed of the index p, q and M_{pq} is fed to a label embedding network to train a compound network together with the word-embedding. In the prediction stage, k-NN search is performed in the low-dimensional feature representation to find similar samples from training datasets. The average of the k-NN's labels is set as final label prediction. [20] proposes to take multi-label co-occurrence patterns into the nueral network objective to improve the classification performance. They also propose to employ dynamic max pooling to capture rich information from different regions of the document and an additional hidden bottleneck layer to reduce model size. Moreover, a binary cross-entropy loss over sigmoid output is tailored to XMLC. However, these methods are not applicable for data with complex hierarchical labels like ours since the decomposition of label hierarchies reduce the label space greatly. In addition, Yan et al [35] proposes Boltzmann CNNs-based hybrid learning network to handle biomedical literature classification. Their work is enriched with data sequence embeddings. This design is not good for huge label space. Their experiments only focus on classes fewer than 2,000 MeSH labels. The work mostly close to ours is hierarchical multi-label classification network (HMCN) proposed by [7]. Their HMCN is claimed to be capable of simultaneously optimizing local and global loss functions for discovering local hierarchical class-relationships and global information from the entire class hierarchy while penalizing hierarchical violations. But their work has higher computational complexity due to the utilization of fully feed-forward layers. Even if they simplify their network with an LSTM-like model with shared weights, it still has high computation burden. It seems that is why they only report works on datasets of at most about 4000 labels.

5 CONCLUSION

In this work, we propose deep learning based level-wise framework to handle extreme multi-label learning and classification, named as Deep Level-wise XMLC. The innovation of Deep Level-wise XMLC includes a few points. Firstly, we develop a split model training mechanism, with which, labels are divided into multiple levels so that the curse of dimensionality and training cost are both lessened to a large degree. Secondly, category-dependent dynamic max pooling and weights adjustments with macro F-measure are integrated into the neural architecture so that the final predictions fit more to the distributions of the levels and their hierarchical relations. Thirdly, we successfully design a hierarchical pointer generation model to merge level-wise outputs into one unified label prediction.

The results show that Deep Level-wise XMLC achieves the state-of-the-art results by utilizing MEDLINE collections, keywords and predicted labels from upper and lower levels are utilized. The results for AmazonCat13K also show that Deep Level-wise XMLC is generic enough to handle diverse datasets.

In this work, it is not hard to see that Deep Level-wise XMLC can be conveniently transferred to tasks, like large scale semantic indexing for constructing more efficient and accurate information retrieval engines and reducing expensive manual expert efforts as shown in this work.

In future work, more experiments will be done on diverse datasets. Besides, model improvements can be made on designing more robust loss functions as well as adding more layers for handling feature refinements or weight adjustments and meanwhile improving running efficiency.

REFERENCES

[1] 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 *Proceedings of the 22nd international conference on World Wide Web (WWW)*. Rio de Janeiro, Brazil, 13–24.

[2] Krishnakumar Balasubramanian and Guy Lebanon. 2012. The Landmark Selection Method for Multiple Output Prediction. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*. Edinburgh, Scotland, UK, 283–290.

[3] William A Baumgartner Jr, K Bretonnel Cohen, Lynne M Fox, George Acquaah-Mensah, and Lawrence Hunter. 2007. Manual curation is not sufficient for annotation of genomic databases. *Bioinformatics* 23, 13 (2007), i41–i48.

[4] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems (NIPS)*. Montreal, Canada, 730–738.

[5] Wei Bi and James Kwok. 2013. Efficient multi-label classification with many labels. In *International Conference on Machine Learning (ICML)*. Atlanta, GA, 405–413.

[6] Róbert Busa-Fekete, Balázs Szörényi, Krzysztof Dembczynski, and Eyke Hüllermeier. 2015. Online F-measure optimization. In *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Canada, 595–603.

[7] Ricardo Cerri, Rodrigo C Barros, and André CPLF De Carvalho. 2014. Hierarchical multi-label classification using local neural networks. *J. Comput. System Sci.* 80, 1 (2014), 39–56.

[8] Yao-Nan Chen and Hsuan-Tien Lin. 2012. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems (NIPS)*. Lake Tahoe, NV, 1529–1537.

[9] Anna E Choromanska and John Langford. 2015. Logarithmic time online multiclass prediction. In *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Canada, 55–63.

[10] Moustapha M Cisse, Nicolas Usunier, Thierry Artieres, and Patrick Gallinari. 2013. Robust bloom filters for large multilabel classification tasks. In *Advances in Neural Information Processing Systems (NIPS)*. Lake Tahoe, NV, 1851–1859.

[11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12, Aug (2011), 2493–2537.

[12] Minlie Huang, Aurélie Névéol, and Zhiyong Lu. 2011. Recommending MeSH terms for annotating biomedical articles. *Journal of the American Medical Informatics Association* 18, 5 (2011), 660–667.

[13] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hullermeier. 2016. Extreme F-measure maximization using sparse probability estimates. In *International Conference on Machine Learning (ICML)*. New York, NY, 1435–1444.

[14] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, 1746–1751.

[15] Dingcheng Li, Jingyuan Zhang, and Ping Li. 2018. Representation Learning for Question Classification via Topic Sparse Autoencoder and Entity Embedding. In *2018 IEEE International Conference on Big Data (IEEE Big Data)*. Seattle, WA, 126–133.

[16] Dingcheng Li, Jingyuan Zhang, and Ping Li. 2019. TMSA: A Mutual Learning Model for Topic Discovery and WordEmbedding. In *In Proceedings of the SIAM conference on Data Mining (SDM)*. Calgary, Canada.

[17] Ping Li. 2009. Abc-boost: Adaptive base class boost for multi-class classification. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*. Montreal, Canada, 625–632.

[18] Ping Li. 2010. Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*. Catalina Island, CA, 302–311.

[19] DA Lindberg. 1999. Internet access to the National Library of Medicine. *Effective clinical practice: ECP* 3, 5 (1999), 256–260.

[20] 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 (SIGIR)*. Tokyo, Japan, 115–124.

[21] Ke Liu, Shengwen Peng, Junqiu Wu, Chengxiang Zhai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2015. MeSHLabeler: improving the accuracy of large-scale MeSH indexing by integrating diverse evidence. *Bioinformatics* 31, 12 (2015), i339–i347.

[22] Yuqing Mao and Zhiyong Lu. 2017. MeSH Now: automatic MeSH indexing at PubMed scale via learning to rank. *Journal of biomedical semantics* 8, 1 (2017), 15.

[23] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. *CoRR* abs/1609.07843 (2016). arXiv:1609.07843 http://arxiv.org/abs/1609.07843

[24] Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage Embedding Models for Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, TX, 955–960.

[25] Paul Mineiro and Nikos Karampatziakis. 2015. Fast label embeddings via randomized linear algebra. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Porto, Portugal, 37–51.

[26] Xin Mu, Feida Zhu, Yue Liu, Ee-Peng Lim, and Zhi-Hua Zhou. 2018. Social Stream Classification with Emerging New Labels. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Melbourne, Australia, 16–28.

[27] David Newman, Sarvnaz Karimi, and Lawrence Cavedon. 2009. Using Topic Models to Interpret MEDLINE's Medical Subject Headings. In *AI 2009: Advances in Artificial Intelligence*, Ann Nicholson and Xiaodong Li (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 270–279.

[28] 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 (2016), i70–i79.

[29] Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*. New York, NY, 263–272.

[30] Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal Attention Model for Neural Machine Translation. *CoRR* abs/1608.02927 (2016). arXiv:1608.02927 http://arxiv.org/abs/1608.02927

[31] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada, 1073–1083.

[32] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, 76–85.

[33] Jason Weston, Ameesh Makadia, and Hector Yee. 2013. Label partitioning for sublinear ranking. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*. Atlanta, GA, 181–189.

[34] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. 2004. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research* 5, Aug (2004), 975–1005.

[35] Yan Yan, Xu-Cheng Yin, Chun Yang, Sujian Li, and Bo-Wen Zhang. 2018. Biomedical literature classification with a CNNs-based hybrid learning network. *PloS one* 13, 7 (2018), e0197933.

[36] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. 2016. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning (ICML)*. New York, NY, 3069–3077.

[37] Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. 2018. Deep Extreme Multi-label Learning. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval (ICMR)*. Yokohama, Japan, 100–107.