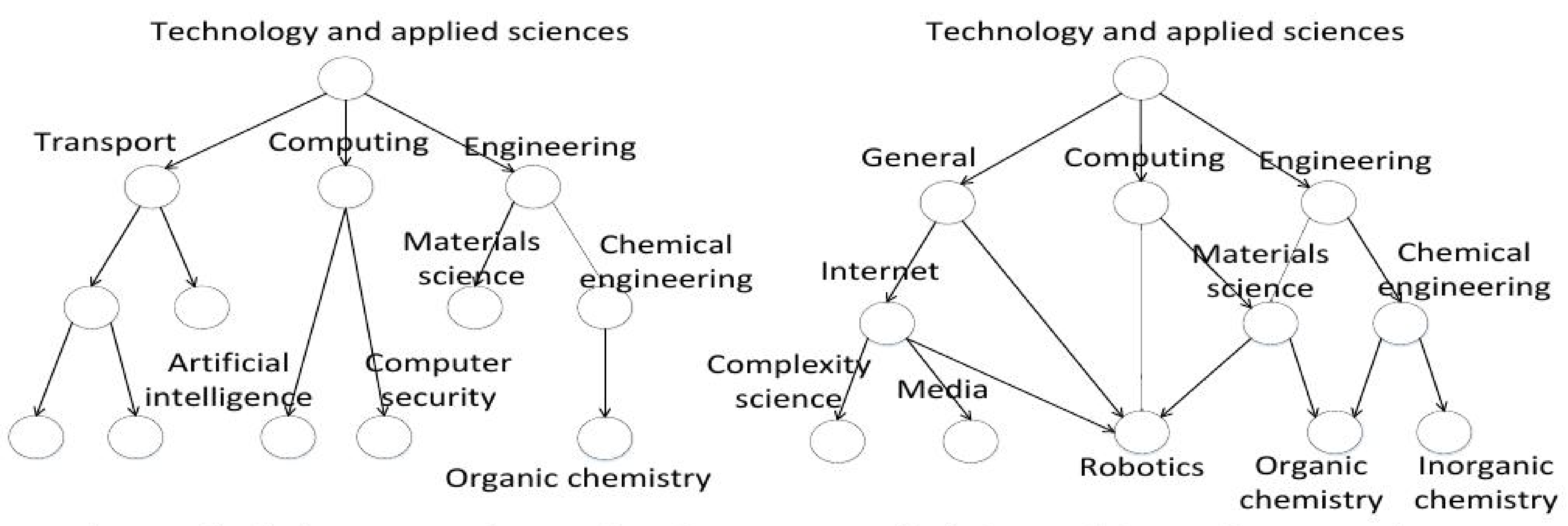

Figure 3: Label hierarchies. The left is a pure hierarchical structure of labels, and the right is graph structure of labels but has no cycles in the graph. Both are commonly used in practice and can be handled by recursive regularization.

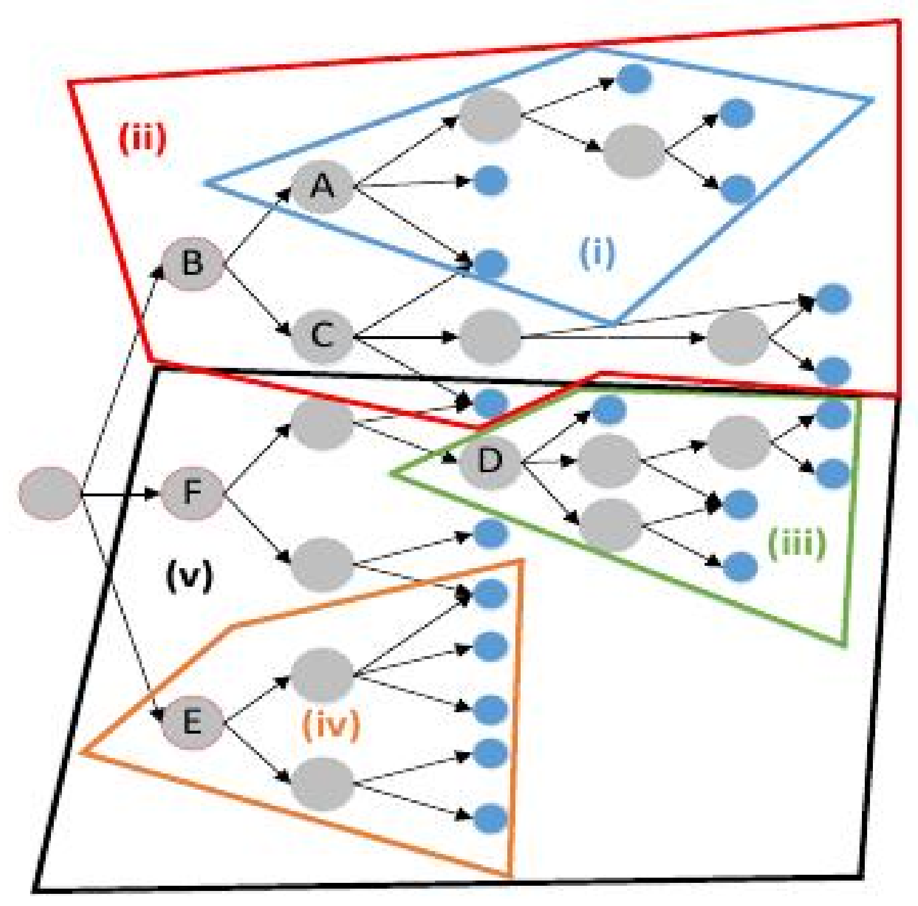

Figure 4: Illustration of Recursive Hierarchical Segmentation. The leaf nodes refer to class-labels, and the internal nodes are super topics. The hierarchy is recursively divided into 5 parts of sub-graphs.

Table 1: Dataset Statistics. The training/test split for RCV1 is done by [27]. The training/test split for NYTimes is done by ourselves, which is 90% for training and 10% for test. For both of the data, we randomly sample 10% from the training data as development sets.

Dataset	Training	Testing	Class-Labels	Depth	avg#Tokens
RCV1	23,149	784,446	103/137	6	240
NYTimes	1,670,093	185,566	2,318	10	629

as one logical label as a leaf. When we backtrack to node B, the children leaf number also is 5. So it's divided into sub-tree (*ii*) and merged into one logical label as a leaf. According to the law of depth-first and preorder traversal, the sub-graph (*iii*) will also be divided, and merged into one logical label as a leaf. When the number of children leaf nodes is less than the threshold value, it will continue to traverse until no more nodes. So the sub-graph *iv* is divided and merged into one logical label. The children leaf nodes of F is 4, so we combine the nodes E and F to meet the requirement of having five leaf nodes. Although the hierarchy is divided into sub-graphs for parallel and distributed deep CNN models, the master program learns to classify top level nodes, such as the B, E, and F in Figure 4, and recursively calls other deep CNN models when testing.

5 EXPERIMENTS

In the experiments, we compare our proposed algorithms with state-of-the-art traditional hierarchical text classification as well as recently developed deep learning architectures on two datasets.

5.1 Datasets and Evaluation Metrics

The two datasets we used are RCV1 and NYTimes datasets. A summarization of statistics of these two datasets is shown in Table 1.

• **RCV1** [27]. RCV1 dataset is a manually labeled newswire collection of Reuters News from 1996-1997. The news documents are categorized with respect to three controlled vocabularies: industries, topics, and regions. We use the topic-based hierarchical classification as it has been most popular in evaluation. There are 103 categories including all classes except for root in the hierarchy. According to [13–15], we also consider the 137 categories by adding some virtual classes.[4] On average, there are 225 documents per label for 103 labels for training. The distribution of RCV1 labels is shown in Figure 5.

• **NYTimes** [35]. This corpus contains nearly every article published in the New York Times between January 01, 1987 and June 19th, 2007. As a large scale corpus, NYTimes was widely used in document routing, document categorization, entity extraction, cross document coreference resolution, and information retrieval, etc. We use the standard of taxonomic classifier labeled in this corpus to test large-scale hierarchical text classification. On average, there are 720 documents per label for 2,318 labels for training. The distribution of NYTimes labels is shown in Figure 6.

We follow [13] to use Micro-F_1 and Macro-F_1 as our evaluation metrics for hierarchical classification.

• **Micro-**F_1 is a F_1 score considering overall precision and recall of all the labels. Let TP_t, FP_t, FN_t denote the true-positives, false-positives, and false-negatives for the *t-th* label in label set $\mathcal{L}$ respectively. Then micro-averaged F_1 is: $P = \frac{\sum_{t\in\mathcal{L}} TP_t}{\sum_{t\in\mathcal{L}} TP_t+FP_t}$, $R = \frac{\sum_{t\in\mathcal{L}} TP_t}{\sum_{t\in\mathcal{L}} TP_t+FN_t}$, $Micro - F_1 = \frac{2PR}{P+R}$.

• **Macro-**F_1 is another F_1 which evaluates averaged F_1 of all different class-labels in the hierarchy. $Macro - F_1$ gives equal weight to each label. Formally, macro-averaged F_1 is defined as: $P_t = \frac{TP_t}{TP_t+FP_t}$, $R_t = \frac{TP_t}{TP_t+FN_t}$, $Macro - F_1 = \frac{1}{|\mathcal{L}|}\sum_{t\in\mathcal{L}} \frac{2P_tR_t}{P_t+R_t}$.

[4] If any internal node in the hierarchy had positive examples, we created a new leaf under it and re-assigned all instances to the leaf node. For all graph based dependencies, if there are two adjacent nodes both of which have training examples, we created an empty dummy node in between them. This is to prevent classes from getting directly regularized towards each other, but regularize towards their mean (the parameters of the dummy node) instead.