From Text Segmentation to Enhanced Representation Learning: A Novel Approach to Multi-Label Classification for Long Texts

Wang Zhang, Xin Wang, Qian Wang*, Tao Deng, Xiaoru Wu
Southwest Petroleum University, Chengdu, China
202221000467@stu.swpu.edu.cn,xinwang@swpu.edu.cn
wangqian2272@gmail.com,{202321000489,202321000483}@stu.swpu.edu.cn

Abstract

Multi-label text classification (MLTC) is an important task in the field of natural language processing. Most existing models rely on high-quality text representations provided by pre-trained language models (PLMs). They hence face the challenge of input length limitation caused by PLMs, when dealing with long texts. In light of this, we introduce a comprehensive approach to multi-label long text classification. We propose a text segmentation algorithm, which guarantees to produce the optimal segmentation, to address the issue of input length limitation caused by PLMs. We incorporate external knowledge, labels' co-occurrence relations, and attention mechanisms in representation learning to enhance both text and label representations. Our method's effectiveness is validated through extensive experiments on various MLTC datasets, unraveling the intricate correlations between texts and labels.

1 Introduction

Multi-label text classification (MLTC) aims to assign multiple relevant labels to a given text. It plays a crucial role in various applications such as content recommendation (Wu et al., 2023), sentiment analysis (Sentiment, 2020), and information retrieval (Henzinger et al., 2023).

The pre-trained language models (PLMs), *e.g.,* BERT (Devlin et al., 2018) and Roberta (Liu et al., 2019) significantly improve the performance of various natural language processing tasks including MLTC. However, PLMs have an inherent limitation in handling long texts due to their maximum input sequence length of 512 tokens. This limitation could lead to information missing when dealing with long texts. More than this, text representations directly obtained from PLMs should be enhanced further, by incorporating *e.g.,* domain knowledge (Liu et al., 2023). Another difficulty for MLTC arises from intricate label-label and word-label relations. These complicated relations, if effectively captured, are expected to improve prediction performance.

To overcome the input length limitation of PLMs, investigators have proposed a series of techniques that are based on *e.g.,* text truncation (Park et al., 2022), sparse attention mechanism (Zaheer et al., 2020; Beltagy et al., 2020), and text segmentation (Ding et al., 2020; Jaiswal and Milios, 2023). Specifically, Park et al. (2022) proposes two methods Bert+Rank and Bert+Random. Bert+Rank concatenates the first 512 tokens of a text representation with the representations of the top-ranked sentences (up to 512 tokens). Unlike Bert+Rank, Bert+Random concatenates the first 512 tokens of a text representation with at most 512 tokens randomly selected from the text corpus. These methods often perform poorly due to the loss of text information. The reason why PLMs limit the input length is that they incorporate a self-attention mechanism, whose computational complexity is in quadratic time of the input length. Therefore, Bigbird (Zaheer et al., 2020) and Longformer (Beltagy et al., 2020) utilize the sparse self-attention mechanism to handle longer input sequences. Nevertheless, they haven't fundamentally solved the issue and can not process sequences of arbitrary length. Ding et al. (2020) proposes a dynamic programming method, named CogLTX, for text segmentation. Jaiswal and Milios (2023) splits a long text into fixed-length chunks, encodes these chunks by BERT, and concatenates the representations of chunks as the representation of the long text. However, the methods above do not take into consideration the semantic change before and after segmentation, hence may damage the semantic coherence of a text.

The above methods mainly focus on the content of texts alone, ignore the integration of external knowledge. As pointed out by Liu et al. (2023),

[*] Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 6864–6873
November 12-16, 2024 ©2024 Association for Computational Linguistics

the utilization of high-quality external knowledge can enhance text representations, and then improve prediction accuracy. However, to incorporate external knowledge, two crucial questions have to be answered: (a) how to establish the connection between texts and external knowledge? (b) how to develop a model to produce text-related knowledge representations? If the above issues were not effectively resolved, external knowledge would become noise and may worsen prediction performance.

Label correlations have been explored in MLTC. A typical model is LDGN Ma et al. (2021). The model first captures labels' co-occurrence relation and then employs the Graph Convolutional Network (GCN) to update label representations based on the co-occurrence relation. However, LDGN initializes label representations randomly without considering the labels' semantic information. Indeed, when the semantic information is incorporated, not only label representations can be improved, but also the importance of words to labels can be captured.

To tackle the issues above, we propose a model (SKFRL) that integrates dynamic Segmentation, Knowledge Fusion, and Representation Learning for multi-label long text classification. SKFRL is equipped with a text segmentation method to partition a long text into sub-texts. It also fuses text representations with external knowledge as enhanced text representations. It further enhances both text and label representations by capturing intricate label-label and word-label relations.

In summary, our contributions are as follows:

- We propose a text segmentation method, named as TXTSEG to address the input length limitation of PLMs. TXTSEG measures the cost of a segmentation on a long text by taking context information of each cut point into consideration, and moves towards the optimal solution via dynamic programming.

- We introduce a knowledge fusing method that fuses text representations from PLMs with concept representations from external knowledge, to obtain updated text representations.

- We achieve enhanced label and text representations learning by integrating external knowledge, label co-occurrence relation and max attention schema.

- We conduct extensive experiments on benchmark datasets to validate the superiority of our model SKFRL compared with other baseline models. The source code of SKFRL is available on GitHub[1].

2 Related Work

Multi-Label Text Classification. MLTC involves assigning multiple relevant labels to a single text, which is more complex than single-label classification. Traditional deep learning methods such as Convolutional Neural Networks (CNNs) (Liu et al., 2017) and Recurrent Neural Networks (RNNs) (Nam et al., 2017) are widely used in MLTC. However, the methods above often overlook the correlation between texts and labels. To address this issue, a method named CORE is introduced by Zhang et al. (2021). CORE takes a text and all the labels as input simultaneously, hence obtains the representations of the text and all the labels in the same space. However, CORE struggles with long texts or large-scale label datasets. Zhang et al. (2023) introduces a two-stage label reduction method that reduces the number of labels through association rules and label merging. The label reduction method heavily depends on the distribution of high-frequency labels in a dataset despite its effectiveness. You et al. (2019) and Xiao et al. (2019) utilize attention mechanisms to construct label-specific text representations, which capture the importance of each word to each label. The correlation among labels is also crucial and plays a key role in MLTC (Yang et al., 2018; Ma et al., 2021). Yang et al. (2018) explores the label-label correlations by treating labels as a sequence and then using the sequence generation model to effectively capture high-order correlations among labels. However, label sequence that is derived from an unordered set, inevitably influences the model's performance. Although LDGN Ma et al. (2021) captures the co-occurrence relation among labels in prediction, it overlooks labels' semantic information but simply initializes label representations with a randomized strategy.

Extensions on PLMs. Pre-trained language models like BERT (Devlin et al., 2018), Roberta (Liu et al., 2019) significantly advance the field of natural language process. However, one limitation of PLMs is their fixed input sequence length, typically capped at 512 tokens. This limitation causes a significant challenge when dealing with long texts. Sun et al. (2019) directly truncates a text and discards the remaining part. Pappagari et al.

[1] https://github.com/Coder-Jeffrey/SKFRL

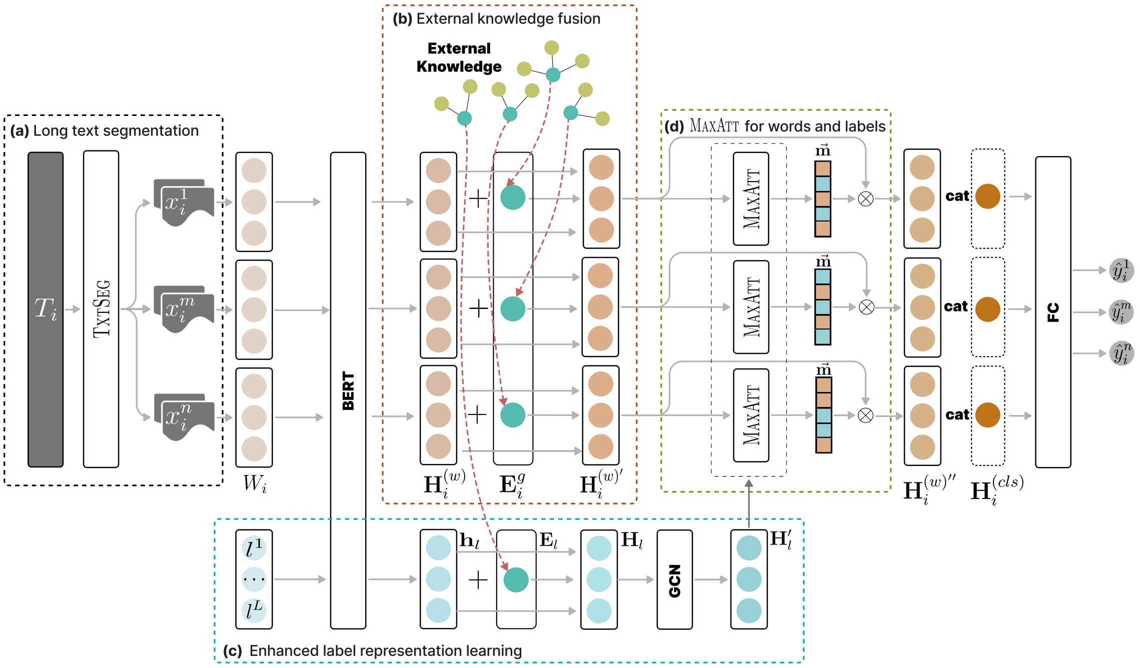

Figure 1: The Architecture of SKFRL. SKFRL consists of four parts. (a) Long text segmentation (TXTSEG): TXTSEG splits the long text T_i into several sub-texts x_i^1, x_i^m, x_i^n by dynamic programming. (W_i contains the words of sub-texts.) (b) External knowledge fusion: External knowledge $\mathbf{E}_i^g$ is incorporated into text representation $\mathbf{H}_i^{(w)}$, resulting in the fused text representation $\mathbf{H}_i^{(w)\prime}$. (c) Enhanced label representation: Similar to (b), the external knowledge of label $\mathbf{E}_l$ is incorporated into label representation $\mathbf{h}_l$ and obtains the fused label representation $\mathbf{H}_l$. GCN is employed to obtain the enhanced label representation $\mathbf{H}_l'$. (d) MAXATT for words and labels: A max attention mechanism (MAXATT) is used to calculate the importance of each word to each label. In addition, $\mathbf{H}_i^{(w)\prime\prime}$ is the label-weighted text representation obtained through (d), $\mathbf{H}_i^{(cls)}$ is the representation of the special token [CLS].

(2019); Jin et al. (2021) simply split a long text into small segments to meet the input length of PLMs. Ding et al. (2020) proposes an objective function, that measures the goodness of a segmentation purely by cut points, and cuts a long text into sub-texts via optimizing the objective function. Jaiswal and Milios (2023) splits a long text into fixed-length chunks and processes these chunks by BERT independently. However, the segmentation of the above methods does not consider the semantic information and the coherence of a text, hence may damage the semantic information of the original text. Chen et al. (2022) proposes a digest algorithm to extract more important sentences at the beginning and end of a text. Recently, extended transformer architectures enable PLMs to handle longer sequences by introducing sparse attention mechanisms. Typical works include Transformer-XL (Dai et al., 2019), Reformer (Kitaev et al., 2020), Longformer (Beltagy et al., 2020), and BigBird (Zaheer et al., 2020). Specifically, Dai et al. (2019) develops a segment-level recurrence mechanism along with a novel positional encoding scheme. Kitaev et al. (2020) replaces the dot-product attention by one that uses locality-sensitive hashing, to reduce computational complexity. Longformer (Beltagy et al., 2020) combines local windowed attention with a task motivated global attention. BigBird (Zaheer et al., 2020) designs a sparse attention mechanism to reduce the quadratic dependency to linear. Though they attempt to reduce complexity by introducing various attention mechanisms, they still face challenges in balancing computational efficiency and model performance.

Our goal is threefold. (1) We develop a text segmentation method that segments a long text by considering semantic coherence, to address the input length constraints of PLMs. (2) We incorporate external knowledge to enrich both text and label representations. (3) We leverage label-label and word-label relations to further enhance text and label representations. By doing so, we aim to obtain a robust model for multi-label long text classification.

3 Method

In this section, we introduce an approach to addressing the issue of MLTC on long texts. As shown in Figure 1, our approach consists of four parts, *i.e.,* Long text segmentation, External knowledge fusion, Enhanced label representation learning, and Representation enhancing with attention (MaxAtt for words and labels). Overall, given a long text, SKFRL first cuts it into several sub-texts with the text segmentation algorithm TxtSeg. After segmentation, all the sub-texts can be encoded via BERT directly, thereby addressing the issue of input length limit. SKFRL then introduces external knowledge to enhance the representation of each sub-text. Meanwhile, SKFRL processes the labels along the similar line as that on texts and gains the fused label representations. It then updates the fused label representations through GCN and obtains the enhanced label representations. Finally, to capture the importance of words to different labels, SKFRL utilizes the max attention mechanism (MaxAtt) to obtain the label-weight text representations. We next elaborate on each of them, starting from the problem formulation.

Problem Formulation: Let $D = \{(T_i, Y_i)\}_{i=1}^{N}$ be the given training text set containing total N texts with corresponding labels. Here T_i refers to the i-th text in D, $Y_i = \{l_i^1, \cdots, l_i^L\}$ is the corresponding label set of 0,1 for T_i, and L denotes the total number of labels. The goal is to train a classifier, which predicts the most relevant labels for a given text, *which might be quite long and exceeds the input length limit of a PLM.*

3.1 Long text segmentation

To overcome the input length limitation of PLMs, we develop an algorithm TxtSeg, for long text segmentation. The pseudo-code of TxtSeg is shown in Algorithm 1. Its core idea is as follows.

Core idea. TxtSeg works in two steps. It first identifies a set of cut points, along with their costs, and maintains them in *poses*. Based on *poses*, it finds the set of cut points that has the least total cost, by following dynamic programming.

Algorithm. We start from the cost metric, followed by the algorithm details.

Cost metric. Specifically, for each character s in a text sequence T, we define a metric $\nu(s)$ to evaluate its cost when s is chosen as a cut point.

$$\nu(s) = pc(s) + sc(s) \quad (1)$$

Here, function $pc(s)$ sets a value for a character s based on the type of s; in this work, period, question mark, and exclamation mark have a cost of 1, comma has a cost of 2, and the cost of other characters is set as 8, by following Ding et al. (2020). Function $sc(s)$ is defined as $\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$, where $\vec{a}$, $\vec{b}$ are the vectors including 20 tokens before and after s in T, respectively.

Algorithm 1 TxtSeg

```
Input: Long text $T$, maximum length max
Output: List $segments$ containing the cut points
 1: Initialize $poses$
 2: Initialize arrays $dp$, $split$; a list $segments$
 3: $dp[0] \leftarrow 0$
 4: for $i \leftarrow 1$ to $len(poses) - 1$ do
 5:     for $j \leftarrow 0$ to $i - 1$ do
 6:         if $poses[i][0] - poses[j][0] \leq$ max then
 7:             $cost \leftarrow dp[j] + poses[i][1]$
 8:             if $cost < dp[i]$ then
 9:                 $dp[i] \leftarrow cost$
10:                 $split[i] \leftarrow j$
11:             end if
12:         end if
13:     end for
14: end for
15: $i \leftarrow len(poses) - 1$
16: while $i > 0$ do
17:     $segments$.append($poses[i][0]$)
18:     $i \leftarrow split[i]$
19: end while
20: $segments$.reverse()
21: return $segments$
```

Given a long text T and the maximum length max (set as 511), TxtSeg first identifies all the punctuation as cut points. For each pair of consecutive cut points cp_s, cp_e, if the length of the sub-text $T(cp_s, cp_e)$ exceeds max, TxtSeg repeatedly performs the following task to identify more cut points. That is, it identifies a new cut point cp_n, that is max distance away from cp_s within $T(cp_s, cp_e)$, and updates $T(cp_s, cp_e)$ with $T(cp_n, cp_e)$. When all the cut points are recognized, TxtSeg calculates their costs by following Eq. (1) and maintains cut points and their costs in array *poses* (line 1). The i-th entry in *poses* is a pair of position and cost. TxtSeg also initializes two arrays *dp* and *split* (line 2). The array *dp* keeps track of the minimum cost from the first cut point to the current cut point; while the array *split* is used to store the index of the previous best cut point for the current cut point. Afterwards, TxtSeg identifies the best set of cut points by following dynamic programming (lines 3-14). Specifi-

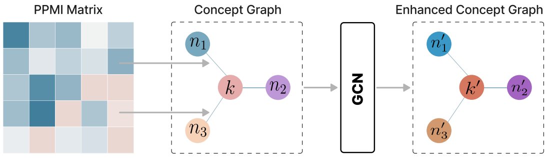

Figure 2: The working flow of enhancing the concept representation. Each item PPMI(w_i, w_j) in the PPMI Matrix denotes the PPMI value of words w_i and w_j in the dataset. k is the concept node, and n_1, n_2, n_3 are the first-order neighbors of concept k. k' is the enhanced concept node, and n_1', n_2', n_3' are the enhanced first-order neighbors of concept k'.

cally, for the i-th ($i \in [1, len(poses)-1]$) cut point $poses[i][0]$, TXTSEG iterates through all previous cut points $poses[j][0]$ and evaluates the cost of the j-th and the i-th cut points. If the length of the sub-text does not exceed `max`, TXTSEG calculates the cost and updates dp and $split$ accordingly. After completion of the above process, TXTSEG backtracks to reconstruct the cut points (lines 16-19). Finally, TXTSEG returns $segments$ that store the best cut points (line 21).

Using TXTSEG, a long text T_i can be divided into several sub-texts, *i.e.,* $T_i = \{x_i^1, ..., x_i^m, ..., x_i^n\}$, where n denotes the number of sub-texts, which will be used for subsequent operations *e.g.,* external knowledge fusion, and MAXATT calculation of words and labels. Hence, we illustrate subsequent operations on sub-texts, for the sake of brevity.

3.2 External knowledge fusion

To enhance the text representation, we fuse it with external knowledge. In this work, a publicly available knowledge graph CONCEPTNET (Speer et al., 2017) that is composed of a large number of triples like '`entity-relationship-entity`' is applied.

Given a sub-text $x_i^m = \{w_i^1, ..., w_i^j, ..., w_i^Q\}$ with Q words, we match each word in x_i^m with the corresponding concept in ConceptNet and obtain the concept sequence $\mathbf{K}_i = \{k_i^1, ..., k_i^j, ..., k_i^Q\}$. If no concept corresponding to w_i^j is found, k_i^j is set as [PAD].

After recognizing all the concepts in a sub-text, we identify the first-order neighbors from CONCEPTNET and construct a concept graph G_i^j based on the first-order neighbors for each concept k_i^j. Later on, TRANSE (Bordes et al., 2013) is employed to train these concept graphs, and the representation of each concept graph is used as the external knowledge representation $\mathbf{E}_i^t$, where $\mathbf{E}_i^t = \{e_i^{(t,1)}, ..., e_i^{(t,j)}, ..., e_i^{(t,Q)}\} \in \mathbb{R}^{Q \times D}$, $e_i^{(t,j)}$ is the representation of the concept k_i^j, and D denotes the dimension of word embedding.

We next incorporate the co-occurrence relationship of text words into the external knowledge representation, to further enhance their representation. The process is shown in Figure 2.

Specifically, we first construct the positive point-wise mutual information (PPMI) of the text words with the following formulas:

$$\text{PMI}(w_i, w_j) = \log\left(\frac{P(w_i, w_j)}{P(w_i)P(w_j)}\right) \tag{2}$$

$$\text{PPMI}(w_i, w_j) = \max(\text{PMI}(w_i, w_j), 0) \tag{3}$$

where $P(w_i)$ and $P(w_j)$ are the probabilities that words w_i and w_j appear in the training set, respectively; $P(w_i, w_j)$ is the co-occurrence probability that words w_i and w_j appear together in a sliding window of size 20 of a text. The size of the sliding window is suggested by Lin et al. (2021). We then assign edge weights to concept graphs as follows: for each edge (v_a, v_b) in a concept graph G_i^j, we update its weight $w(v_a, v_b)$ with PPMI(w_a, w_b) if (w_a, w_b) is in the matrix, otherwise, we set $w(v_a, v_b)$ as 0. Here, w_a (resp. w_b) is the corresponding word of v_a (resp. v_b). After updating the edge weights, we obtain a normalized adjacency matrix $\hat{\mathbf{A}}$ of each concept graph, with Eq. (5) (will be given in Section 3.3). Lastly, we employ GCN to update the representation of a concept graph G_i^j with its corresponding $\hat{\mathbf{A}}$, extract the representation of each concept $e_i^{(g,j)}$ from G_i^j and obtain an enhanced representation $\mathbf{E}_i^g = \{e_i^{(g,1)}, ..., e_i^{(g,j)}, ..., e_i^{(g,Q)}\} \in \mathbb{R}^{Q \times D}$ as the enhanced representation of $\mathbf{K}_i$.

In addition, we encode the sub-text x_i^m via a PLM and obtain two representations. One is the output of the sequence of the last hidden layer $\mathbf{H}_i^{(w)} = \{h_i^{(w_1)}, ..., h_i^{(w_j)}, ..., h_i^{(w_Q)}\} \in \mathbb{R}^{Q \times D}$ and the other one is the output of [CLS] token $\mathbf{H}_i^{(cls)} \in \mathbb{R}^{1 \times D}$, which will be used later. We incorporate $\mathbf{H}_i^{(w)}$ with external knowledge representation $\mathbf{E}_i^g$ with Eq. (4) and obtains the final representation of x_i^m.

$$\mathbf{H}_i^{(w)'} = \mathbf{H}_i^{(w)} + \mathbf{E}_i^g \tag{4}$$

where $\mathbf{H}_i^{(w)'} \in \mathbb{R}^{Q \times D}$ is the fused text representation.

3.3 Enhanced label representation

In MLTC, a label, *e.g.*, '`international affairs`', often carries semantic information. The application of semantic information has been proven effective in improving classification performance. Hence, we first encode all the labels in a dataset through a PLM and obtain their initial representations. Along the similar line to the knowledge fusion on texts, we also fuse the label representations with external knowledge. In contrast to the fusion operation on texts, we use GraphSAGE (Hamilton et al., 2017) instead of GCN to update label representations, since label words are much less than text words. After fusion, we obtain the label representations $\mathbf{H}_l \in \mathbb{R}^{L\times D}$, where L is the number of labels.

Prior works show that the co-occurrence relation among labels is useful in classification. To this end, we further enhance the label representations by incorporating the co-occurrence relation of labels. Specifically, we first construct a label co-occurrence graph, where each node represents a label and each edge indicates the label co-occurrence between two labels, from the training set. We next obtain the adjacency matrix $\mathbf{A}$ from the label co-occurrence graph and normalize it as $\hat{\mathbf{A}}$ with Eq. (5):

$$\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{5}$$

where $\mathbf{D} \in \mathbb{R}^{L\times L}$ is a diagnal degree matrix.

We finally feed the label representations $\mathbf{H}_l$ and $\hat{\mathbf{A}}$ into GCN and obtain the enhanced label representations (Eq. (6)):

$$\mathbf{H'}_l = LeakyReLU(\hat{\mathbf{A}}\mathbf{H}_l\mathbf{W}_1) \tag{6}$$

where $\mathbf{W_1} \in \mathbb{R}^{D\times D}$ is a learnable transformation matrix, and $\mathbf{H'}_l \in \mathbb{R}^{L\times D}$ is the label representations with the labels' co-occurrence relation incorporated.

3.4 Enhancing text representation with max attention mechanism

In MLTC, the importance of each word in a text is different for labels. Taking "`The football club's stock soared after signing a major sponsorship deal with a top bank`" as an example, it is clear that the word '`football`' is the most important word for the label '`sports`'. It is hence necessary to capture the importance of a word to a label and enhance text representation with the attention. To this end, we introduce an attention mechanism, which is named MaxAtt and works as Algorithm 2.

Algorithm 2 MaxAtt

```
Input: Text and label representations $\mathbf{H}_i^{(w)'}$, $\mathbf{H'}_l$.
Output: Updated label representation.
      /*calculate the importance of word-to-label */
  1: $\mathbf{M} \leftarrow \mathbf{H}_i^{(w)'} \cdot \mathbf{H'}_l^{\mathrm{T}}$
  2: $\mathbf{M}_a \leftarrow \{\tanh(\max(\mathbf{M}[i])) \mid i \in [1:Q]\}$
  3: $\vec{\mathbf{m}} \leftarrow \textsc{SoftMax}(\mathbf{M}_a)$
      /* compute the label-weighted text representation */
  4: $\mathbf{H}_i^{(w)''} \leftarrow \mathrm{mean}(\vec{\mathbf{m}} \cdot \mathbf{H}_i^{(w)'})$
      /* obtain the final text representation */
  5: return $\mathbf{H}_i^{(w)''}$
```

Taking a text representation $\mathbf{H}_i^{(w)'}$ and a label representation $\mathbf{H'}_l$ as input, MaxAtt first obtains a word-to-label weight matrix $\mathbf{M}$ by multiplying $\mathbf{H}_i^{(w)'}$ with $\mathbf{H'}_l$ (line 1). MaxAtt then performs max pooling and nonlinear transformation on each row of $\mathbf{M}$, to obtain the max attention vector $\vec{\mathbf{m}}$ of the text (lines 2-3). Next, MaxAtt multiplies $\vec{\mathbf{m}}$ by $\mathbf{H}_i^{(w)'}$ and applies average pooling to obtain the label-weighted text representation $\mathbf{H}_i^{(w)''}$ (line 4).

Model learning & testing. Given a label-weighted representation $\mathbf{H}_i^{(w)''}$, SKFRL concatenates it with $\mathbf{H}_i^{(cls)}$ to obtain the final text representation $\mathbf{H}_i^{(w,f)}$, and feeds $\mathbf{H}_i^{(w,f)}$ into a fully connected layer to produce the predicted results. To train the model, we select the Binary Cross-Entropy (BCE), which is commonly used in MLTC task, as the loss function (Ma et al., 2021; Xiao et al., 2019).

$$\mathcal{L}_{bce}^{ij} = -(y_i^j \log(\hat{y}_i^j) + (1 - y_i^j)\log(1 - \hat{y}_i^j)) \tag{7}$$

$$\mathcal{L}_{bce} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L}\mathcal{L}_{bce}^{ij} \tag{8}$$

where N and L denote the total number of texts and labels, respectively; y_i^j and $\hat{y}_i^j$ represent the ground truth (0 or 1) and the prediction result of the j-th label for the i-th text, respectively.

To predict the labels of a text T_i, SKFRL first outputs predicted results $\{\hat{y}_i^1, ..., \hat{y}_i^m, ..., \hat{y}_i^n\}$ for each of its sub-text, performs max pooling (Eq. (9)) over the results of sub-texts, and picks labels whose predicted values $\hat{y}_i^m$ $(m \in [1, n])$ are above a threshold (0.5 in this work).

$$\hat{y}_i = \max(\hat{y}_i^1, ..., \hat{y}_i^m, ..., \hat{y}_i^n) \tag{9}$$

where $\hat{y}_i$ is a vector of dimension L that serves as the label prediction result of the text T_i.

4 Experiment

4.1 Experimental Setup

Datasets. To ensure a fair comparison with the baseline, we select four MLTC benchmark datasets. Table 1 shows the details of the four datasets. CMU Book Summary is an MLTC dataset containing 12788 samples (texts) with multiple labels (up to 227). Pairs of book summaries are created by combining text pairs from the CMU Book Summary dataset. EURLEX-57K is a large-scale MLTC dataset containing 57,000 European Union legal texts with labels (up to 4271). Inverted EURLEX-57K is a modified version of the EURLEX-57K dataset in which the order of chapters is reversed to ensure that the key information focuses on the end of the text.

Baselines. The following baselines are compared with our proposed method.

- **Bert** (Devlin et al., 2018) is adopted by (Park et al., 2022) to conduct MLTC, which achieves a good result.
- **Bert+TextRank** (Park et al., 2022) obtains the features of MLTC by concatenating the representation of the first 512 tokens of the sentence with the representation of the top-ranked sentences.
- **Bert+Random** (Park et al., 2022) concatenates the representation of the first 512 text tokens with the other 512 text tokens randomly selected from the text to conduct MLTC.
- **Longformer** (Beltagy et al., 2020) utilizes a sparse attention mechanism that scales linearly with sequence length, to make it easy to process more tokens (up to 4,096 tokens).
- **ToBERT** (Pappagari et al., 2019) adopts a hierarchical approach to processing long texts for the MLTC task.
- **CogLTX** (Ding et al., 2020) splits long text into chunks by dynamic programming. The chunks with high scores are used for MLTC.
- **ChunkBERT** (Jaiswal and Milios, 2023) splits a long text into chunks with fixed-length. The concatenated chunk representation is used for MLTC.

Parameter setting. We implement our model in PyTorch and train it with 1 NVIDIA 4090. BERT is used for the PLM. The maximum text length and batch size are set to 512 (including the [CLS] token) and 2 respectively. The learning rate is 3e-5 for EURLEX and Paired Summary, 5e-5 for Inverted EURLEX, and 7e-6 for Book Summary. We train our model for 20 epochs. The checkpoint with the best performance is used to test our proposed model.

Evaluation metrics. We use Micro-F_1 (%) as the evaluation metric.

4.2 Results and Analysis

Our proposed SKFRL is evaluated with various baseline methods across four benchmark datasets. To evaluate the performance of SKFRL on long text, the texts (tokens exceeding 512) of four benchmark datasets are picked out as a new test set. In addition, the results on the full test set (*i.e.,* long texts are not picked out) are given in the Appendix (see the Appendix A.1 for details).

Table 2 shows the results on long texts. Our proposed SKFRL achieves the state-of-the-art performance on 4 datasets. Specifically, SKFRL improves the previous best results from 67.03, 64.31, 62.34, 58.55 to 67.12 (+0.09), 66.04 (+1.73), 62.47 (+0.13), 62.92 (+4.37) respectively. In addition, the simple method, *i.e.,* Bert, performs well on the first three datasets, except for the Paired Summary. Compared with Bert, Bert+TextRank and Bert+Random achieve a better result by enhancing the text representation with incorporating additional information from TextRank or random tokens. In addition, while the performance of popular models such as Longformer, ToBERT, ChunkBERT, and CogLTX achieve encouraging results, their performance still is limited by long text. Different from the methods above, our proposed SKFRL leverages a novel long-text segmentation method and knowledge fusion strategy to improve the performance of MLTC on long texts. Experimental results also show that SKFRL achieves a remarkable improvement on MLTC of long texts.

4.3 Ablation Study

We conduct ablation studies to verify the effectiveness of the sub-modules in SKFRL. As shown in Table 3, our observations are as follows: (1) Our proposed SKFRL outperforms all variants based on SKFRL. It is shown that each sub-module is effective on improving the performance. (2) Compared with Variant 1, Variant 2 performs better,

Dataset	#Total	#Train	#Dev	#Test	#Labels	#BERT Tokens	%Long
EURLEX-57K	57000	45000	6000	6000	4271	707.99±538.69	51.3
- Inverted	57000	45000	6000	6000	4271	707.99±538.69	51.3
Book Summary	12788	10230	1279	1279	227	574.31±659.56	38.46
- Paired	6393	5115	639	639	227	1148.62±933.97	75.54

Table 1: Statistical information of the datasets. #BERT Tokens indicates the average token number obtained via the BERT tokenizer. %Long means the proportion of texts exceeding 512 BERT tokens.

Model	EURLEX	Inverted EURLEX	Book Summary	Paired Summary
Bert	66.76	62.88	60.56	52.23
Bert+TextRank	66.56	64.22	61.76	56.24
Bert+Random	67.03	64.31	62.34	56.77
Longformer	44.66	47.00	59.66	58.55
ToBERT	61.85	59.50	61.38	58.17
CogLTX	61.95	63.00	60.71	55.74
ChunkBERT	64.94	62.94	57.80	57.73
SKFRL	**67.12**	**66.04**	**62.47**	**62.92**

Table 2: Model performances on long texts. The highest and second-highest scores are bolded and underlined.

Variant	EKF	ELR	ATT	EURLEX	Inverted EURLEX	Book Summary	Paired Summary
1	×	×	×	65.99	64.76	61.37	61.23
2	✓	×	×	66.21	65.02	61.58	61.53
3	✓	×	✓	66.91	65.77	62.10	62.57
4	×	✓	✓	66.93	65.85	62.23	62.60
SKFRL	✓	✓	✓	**67.12**	**66.04**	**62.47**	**62.92**

Table 3: The results of the ablation study. EKF and ELR denote the external knowledge fusion and enhanced label representation learning respectively. ATT refers to the MAXATT for word and label. '✓' (resp. '×') denote the Variant with (resp. without) the corresponding operations (*i.e.,* EKF, ATT, and ELR).

Variant	EURLEX	Inverted EURLEX	Book Summary	Paired Summary
Seg_p+KG_s	66.67	65.56	61.93	62.19
Seg_{p+s}+KG_s	66.91	65.88	62.36	62.68
Seg_{p+s}+KG_p (SKFRL)	**67.12**	**66.04**	**62.47**	**62.92**

Table 4: Ablation study results of long text segmentation and external knowledge training method. Seg_p denotes that the text segmentation is conducted by punctuation. Seg_{p+s} denotes that the text segmentation is performed by punctuation and text semantic similarity. KG_s denotes that text representation is enhanced by external knowledge with GraphSAGE. KG_p denotes that the text representation is enhanced by external knowledge with PPMI and GCN.

which indicates the effectiveness of EKF. (3) Compared with Variant 2, Variant 3 adds the max attention mechanism MAXATT. As we mentioned in Section 3.4, MAXATT can help the model learn the importance of words for different labels, resulting in a great improvement. (4) The performance of SKFRL is better than Variant 3, verifying the effectiveness of ELR. ELR enhances the label representation through GCN, which enables the model to capture the label semantic information and the label co-occurrence relation.

In addition, in Section 3.1, different from Ding et al. (2020), we incorporate text semantic similarity as a cost of segmentation. In Section 3.2, we introduce PPMI and employ GCN to obtain the enhanced text external knowledge representation, which differs from the GraphSAGE method used by Liu et al. (2023). To verify the effectiveness of our improved methods, we conduct ablation study. The results are shown in Table 4. The experimental results show that our proposed method $Seg_{p+s}+KG_p$ achieves the best performance compared with other variant methods (*i.e.,* Seg_p+KG_s and $Seg_{p+s}+KG_g$). Specifically, the model with Seg_p+KG_s can achieve good results across the four datasets. Once the text segmentation method (*i.e.,* $Seg_{p+s}+KG_s$) is applied, the performance of the model is improved. The performance of the model is further improved and optimized on all datasets by introducing the external knowledge representation with PPMI and GCN method (*i.e.,* $Seg_{p+s}+KG_p$).

5 Conclusion

In this work, we propose a novel model for multi-label long text classification. The model achieves superior text segmentation by following dynamic programming, thereby addressing the issue of input length limitation caused by PLMs. The use of external knowledge enriches the text representation encoded by PLMs effectively. To further enhance the text representation, GCN and max-attention mechanisms are utilized to obtain the label-weighted text representation. Experimental results show that our model outperforms baselines, which confirms the effectiveness of our model in text segmentation and incorporation of external knowledge.

Limitations

There are some limitations for our work. Firstly, due to the limitation of computational resources, we only verify our method on the BERT-base. Extending our experiments to larger PLMs would make our work more convincing. Secondly, the performance of the model could be further improved by employing the correlation among sub-texts. We will solve the limitations above in the future.

Acknowledgements

This work is supported by Sichuan Scientific Innovation Fund (No. 2022JDRC0009), China.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150.*

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Xinying Chen, Peimin Cong, and Shuo Lv. 2022. A long-text classification method of chinese news based on bert and cnn. *IEEE Access*, 10:34046–34057.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33:12792–12804.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Alexandra Henzinger, Matthew M Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. 2023. One server for the price of two: Simple and fast {Single-Server} private information retrieval. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3889–3905.

Aman Jaiswal and Evangelos Milios. 2023. Breaking the token barrier: Chunking and convolution for efficient long text classification with bert. *arXiv preprint arXiv:2310.20558.*

Yong Jin, Qisi Zhu, Xuan Deng, and Linli Hu. 2021. Weighted hierarchy mechanism over bert for long text classification. In *International Conference on Artificial Intelligence and Security*, pages 566–574. Springer.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451.*

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. Bertgcn: Transductive text classification by combining gcn and bert. *arXiv preprint arXiv:2105.05727.*

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.

Ye Liu, Kai Zhang, Zhenya Huang, Kehang Wang, Yanghai Zhang, Qi Liu, and Enhong Chen. 2023. Enhancing hierarchical text classification through knowledge graph integration. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5797–5810.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692.*

Qianwen Ma, Chunyuan Yuan, Wei Zhou, and Songlin Hu. 2021. Label-specific dual graph neural network for multi-label text classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3855–3864.

Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. *Advances in neural information processing systems*, 30.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 838–844. IEEE.

Hyunji Hayley Park, Yogarshi Vyas, and Kashif Shah. 2022. Efficient classification of long documents using transformers. *arXiv preprint arXiv:2203.11258.*

Tweet Sentiment. 2020. Sentiment analysis of social media response on the covid19 outbreak. *Brain, behavior, and immunity*, 87:136–137.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*, pages 194–206. Springer.

Chuhan Wu, Fangzhao Wu, Yongfeng Huang, and Xing Xie. 2023. Personalized news recommendation: Methods and challenges. *ACM Transactions on Information Systems*, 41(1):1–50.

Lin Xiao, Xin Huang, Boli Chen, and Liping Jing. 2019. Label-specific document representation for multi-label text classification. In *Proceedings of the 2019 Conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing, EMNLP-IJCNLP 2019*, pages 466–475.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018*, pages 3915–3926.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *Advances in Neural Information Processing Systems, NeurIPS 2019*, pages 5812–5822.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Qian-Wen Zhang, Ximing Zhang, Zhao Yan, Ruifang Liu, Yunbo Cao, and Min-Ling Zhang. 2021. Correlation-guided representation for multi-label text classification. In *IJCAI*, pages 3363–3369.

Wang Zhang, Xin Wang, Yuhong Wu, Xingpeng Zhang, and Huayi Zhan. 2023. Enhancing representation learning with label association for multi-label text classification. In *2023 IEEE International Conference on Big Data (BigData)*, pages 1012–1021. IEEE.

A Appendix

A.1 Result on full test sets

Table 5 presents the performance of all methods on full test sets. Overall, the results are similar to the trends observed in Table 2. As shown in Table 5, we can observe that our proposed model SKFRL outperforms other comparison models on all datasets. Specifically, SKFRL improves the previous best result from 73.22, 71.47, 59.36, 57.76 to 73.36 (+0.14), 72.75 (+1.28), 59.77 (+0.41), 61.71 (+3.95). These simple modes such as Bert, Bert+TextRank, and Bert+Random show competitive performance, outperforming the majority of popular models(*i.e.,* CogLTX, and ChunkBERT). In addition, only Longformer and ToBERT achieve a good result on Paired Summary. The results indicate that existing models' performance on MLTC exactly still are limited by long text.

Model	EURLEX	Inverted EURLEX	Book Summary	Paired Summary
Bert	73.09	70.53	58.18	52.24
Bert+TextRank	72.87	71.30	58.94	55.99
Bert+Random	<u>73.22</u>	<u>71.47</u>	<u>59.36</u>	56.58
Longformer	54.53	56.47	56.53	<u>57.76</u>
ToBERT	67.57	67.31	58.16	57.08
CogLTX	70.13	70.80	58.27	55.91
ChunkBERT	70.92	69.49	54.08	56.42
SKFRL	**73.36**	**72.75**	**59.77**	**61.71**

Table 5: Experimental results on the full test set. The highest score in each column is bolded, while the second-highest score is underlined.