Utilizing Keyphrase Generation and Semantic Similarity for Extreme Multi-Label Text Classification

Xiangting DAI † Mizuho IWAIHARA ‡

Graduate School of Information, Production and Systems, Waseda University

2-7 Hibikino, Wakamatsu ku, Kitakyushu-shi, Fukuoka, 808-0135 Japan

E-mail: † daixt@akane.waseda.jp, ‡ iwaihara@waseda.jp

Abstract Extreme multi-label classification (XMC) is an important problem of assigning relevant labels to a text from a vast label domain. The existing XMC methods utilize machine learning or deep learning methods to extract dense representation from the input text and select labels from a vast label set. There are two problems in the existing methods: 1) The performance of abstract label extraction is poor, and 2) statistical sampling of negative labels is reducing the accuracy. Recently, BERT or other pre-trained language models are extensively used as a document encoder for the XMC problem, whereas text2text models have not been used. In this paper, we propose a method based on generating keyphrases by a text2text language model, rather than retrieving labels from the label set. Keyphrase generation is the process of predicting both present and absent keyphrases from a given document. The vast majority of generated keyphrases can be found in the whole label set, but a portion of generated keyphrases is not in the label set. We consider replacing non-existing labels by existing labels. Finally, generated candidate labels are ranked for selecting the final labels, where a bi-encoder-based label ranking model is trained with false labels not in the label set, and produces sentence-pair relevance scores. Our experiments over XMC benchmark collections show notable superiorities of our approach.

Keyword Extreme Multi-Label Classification, Keyphrase generation, Pre-trained Model, Semantic Similarity

1. Introduction

Multi-label text classification is assigning more than one label to a document, which is a common task in real world applications, such as e-commerce commodity recommendation [25] and tagging Wikipedia articles [17]. Also, in this scenario the set of possible labels from which document labels are chosen can be extremely large, which is called extreme-multi label classification (XMC) [9] [17] [19]. In the XMC task, the target document shall be assigned one or more labels from a numerous label collection. XMC datasets have many labels, ranging from 4K in the EUR-Lex dataset and 3M labels in the Amazon3M dataset [10].

Various methods have been proposed in recent years to tackle XMC, including traditional machine learning and statistical methods like DiSMEC [18] and FastXML [23]. Indeed, with the development of deep learning, more researches have focused on deep learning methods such as XML-CNN [9]. However, these methods are commonly based on certain statistical representation model like Bag-of-words (BOW) or shallow neural network models that can produce word embeddings like word2vec. On the other hand, recent methods, such as AttentionXML [17], have used hierarchical label trees to classify candidate labels from a large label set. However, these approaches do not consider utilizing the raw label texts. In recent years, with the success of pre-trained language models like BERT [7], RoBERTa [22], and XLNet [25], which have a remarkable performance because of their contextualized semantic representation of texts. In various NLP tasks, XMC methods like LightXML [19] utilize pre-trained language models for text representations and dynamic negative label sampling to avoid the long tail, sparse label space of XMC.

Although the methods mentioned above have obtained salient results in XMC tasks, we also find that the existing methods have yet to exploit semantic relatedness of label texts for XMC tasks. Virtually, most methods directly use BOW features rather than the raw original label texts. Meanwhile, because of the superficial semantic information and large-scale spare vector representations of BOW features, the problems of high computing costs and lack of semantic relations between labels are evident.

Nowadays, almost all of existing methods try to utilize a pre-trained language model to obtain text representations for retrieving labels from the label set. A number of methods that utilize pre-trained language models for obtaining text vectors, like LightXML, merely use the encoder structure of the transformer. Indeed, the pre-trained model composed with a transformer encoder has a brilliant performance for various NLP tasks. However, they are inapplicable to generation tasks such as summarization

and question answering.

In recent years, with the remarkable progress of the pre-trained language models [7], a vast number of sequence-to-sequence models have appeared, such as BART [14], which is trained by corrupting texts with random noise and reconstructing the original document.

In recent years, more and more NLP tasks can be converted into a unified pattern, converting most of NLP tasks into the text-to-text formulation. Text-to-text model T5 [3][12] shows the capacity of reframing all NLP tasks into a unified text-to-text format. Accordingly, it is feasible to recall labels from a large label set by a text-to-text language model. For instance, there exist researches on keyphrase generation with a text-to-text pre-trained language model [1][21]. So, we are naturally enlightened by this direction. There exists potential for utilizing a text-to-text pre-trained language model for generating labels for XMC.

To address the abovementioned approaches, we propose a method based on keyphrase generation and semantic similarity for XMC tasks. Firstly，we try to use a text-to-text language model to generate keyphrases from input texts. These keyphrases can be regarded as concise representations of the input text. Since the text-to-text model may generate keyphrases which do not belong to the label set, we need to select labels close to these non-existent labels.

There are two strategies for non-existent labels. First is directly removing the non-existent labels. However, this way may lose the semantic information obtained from the text-to-text model. Therefore, our method uses a more flexible approach by sentence-transformer [15] to transform non-existent labels. Sentence-transformer is a framework based on Sentence-BERT, which can obtain embeddings from sentences and calculate semantical similarities between sentences. We use a sentence-transformer to calculate the similarity between each generated keyphrase which is not in the label set and each label in the label set. Moreover, we replace all non-existent labels with labels having the highest semantic similarities. Finally, the candidate labels are ranked to choose the final predicted labels.

Our contribution is converting label recall and classification tasks into generation tasks by a text-to-text language model. On the other hand, our label recalling method based on keyphrase generation is distinct from any other existing label recalling method of XMC tasks. The comparison and difference between our method and other conventional methods are shown in Figure 1.

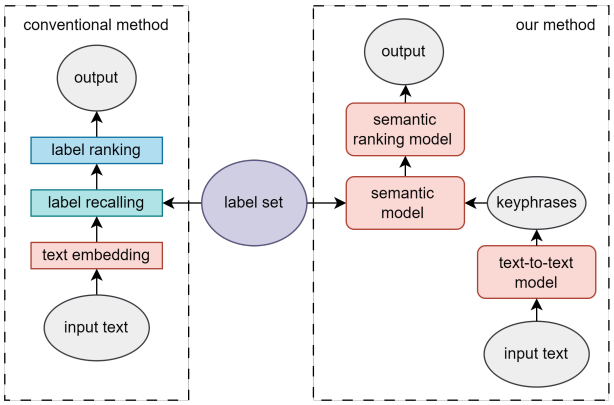

Figure 1: the comparison between our method and conventional method.

We believe the generation model can be finetuned to generate texts that are semantically close to the predefined labels. Since our method utilizes text information of labels, we can also compute semantic relatedness between label names.

In this paper, we proposed a brand-new method based on keyphrase generation and semantic similarity for the extreme multi-label classification task. We evaluate our model on four public datasets of the XMC task. Compared with the state-of-the-art model, the result from our proposed method shows a competitive result.

2. Related Work

2.1 Machine Learning method

Previous XMC methods conventionally use machine learning techniques to obtain labels from the label set by BOW and TF-IDF features. The enormous number of labels means huge spare vector dimensions. There are three basic directions: One-vs-all (OVA) method, label tree-based method, and embedding-based method. OVA methods like Parable [24], DiSMEC [18], PD-Sparse [6] apply binary classification tasks to each label. This method has several problems, such as computing cost and model size.

Another category is the label tree-based method, which focuses on decreasing computing cost caused by the large label number in the OVA method. The method like FastXML [23] and CRAFTML [20] can build a hierarchical tree with partitioned labels clusters. The tree-based methods' difficulties are the tree structure's design and how

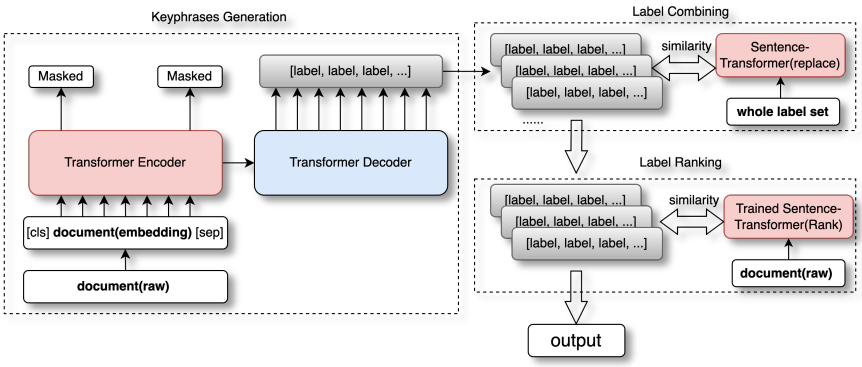

Figure 2: The overview of our proposed model based on keyphrases generation and semantic similarity.

to build an efficient and accurate classifier for each tree leaf node. Most of label tree-based methods use k-means [18] or other clustering algorithms to partition labels.

The final category is embedding methods, which aim to transfer high label dimension into lower dimension space. Some modified embedding methods like SLEEC [13] utilizes label compression and decompression on label embeddings for XMC tasks.

2.2 Deep learning method

Deep learning models have proved their capability and potential in various NLP tasks. Compared with the classical machine learning based method, the deep learning method have made significant progress in text information extraction and representation. However, they need higher float computing resources like GPU and TPU. Generally, machine learning methods will use BOW features as the representation of input text. In contrast, deep learning methods like CNN-XML and LightXML can apply deep learning models to represent input texts, which can gather more contextualized semantic information.

2.3 text-to-text language model

The pre-trained language model has become the most popular topic in the NLP field for the last few years. Some existing methods, Like LightXML, have used pre-trained language models like BERT and RoBRETa for text representation. On the other hand, the text-to-text models like BART, T5, and Pegasus have different structures compared with each other model. BART uses a denoising autoencoder architecture and bidirectional attention mechanisms. T5 can be finetuned for a wide range of language tasks. And Pegasus focuses on extracting gap-sentences from a document with more parameters than the other two models.

Currently, there is no research exploring the possibility of a text-to-text model for XMC tasks. Diya, A [1] discussed the keyphrases generating tasks by finetuning the BART model. In this paper, text-to-text models are finetuned by specific formats and can generate keyphrases by input text. The keyphrase format is like a label in the XMC tasks.

2.4 Sentence-transformer

Sentence-transformer is an improved model of the pre-trained-BERT model that operates at the sentence level [16]. It has been shown to be effective for a wide range of tasks, including semantic textual similarity and document classification. Sentence-transformer can encode sentences' semantics and contextual relationships, resulting in a strong performance on various benchmarks. There are two types of sentence-transformer models: Cross-encoder, which can deliver two sentences into a transformer model and generate a value of output as the relevance score of the sentence pair, and bi-encoder, which can generate embedding of sentences, and to cosine similarities of these embeddings are used to evaluate semantic relatedness of the sentences.

3. Methodology

Figure 2 shows the overview of our proposed new model for XMC tasks. This model can be separated into three parts: Keyphrase generation, label combining, and label ranking. Firstly, we should train a text-to-text model for keyphrase generation from documents and their ground truth labels. Next, we utilize a method to resolve the non-existent label problem and obtain filtered predicted labels after the combining part. Finally, we consider using the method of semantic comparison to sort the labels from the last step.

3.1 Problem definition

Based on the information above about the extreme multi-label classification, we should use a more specific expression to represent it. In XMC tasks, we denote the XMC datasets as D_x. The D_x consists of a set of tuples (d_i, L_i) including documents and their ground truth label sets, where d_i is the i-th document and $L_i = \{l_1, l_2, \ldots, l_i\}$ is

the set of ground truth for i-th document. Otherwise, D_x also have a whole label set $\boldsymbol{S}_{\mathrm{L}}$ that includes all labels that appear in all training and testing dataset.

3.2 Keyphrase Generation

In This part, we proposed a method that can be performed in text2text style downstream task to get some preliminary keyphrases which can be divided into raw labels in XMC tasks. These raw labels are close to the real labels rather than indexes in the XMC tasks. Based on the real labels pipeline in our method, some datasets of XMC tasks, such as Amazon-3M, are hard to evaluate because these datasets need complete real label information.

We train a standard text2text model for keyphrase generation tasks. Considering the model size and the masked and reconstruction strategy, we choose BART model as the primary language model, which has a bidirectional encoder and autoregressive decoder [14] and is suitable for text generation tasks. The basic mathematical expression for keyphrases generation is given a sequence of input variables$X = [x_1, x_2, \dots, x_m]$ and a sequence of output $\hat{Y} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n]$ based on the probability estimation:

$$P(\hat{Y}|X;\theta) = \prod_{i=1}^{n+1} P(\hat{y}_i\,|\hat{y}_{0:n-1}, x_{1:m};\theta) \tag{1}$$

where $\hat{y}_0$ is the <BOS> token and the $\hat{y}_{n+1}$ is the <EOS> token, θ is the parameters of the model. Otherwise, the loss function is the cross-entropy of predicting sequence $\hat{\boldsymbol{Y}}$ and ground truth sequence which consists of a set of labels. The loss function can be represented by:

$$L = -\sum_{j=1}^{T}\sum_{k=1}^{N} M_j y_{j,k} \log(\hat{y}_{j,k}) \tag{2}$$

where T is the length of ground truth sequence, N is the length of predicted keyphrases sequence, M_j is the masked value for the position j, $y_{j,k}$ is the ground truth sequence, $\hat{y}_{j,k}$ is the j-th token of predicted keyphrases sequence.

Keyphrases are a word sequence that can be shown the concise representations of the input texts with different organizational format compared with the label set of a XMC dataset. The processing of mapping keyphrases into XMC labels is:

$$\{\hat{Y}_{kg}^1, \hat{Y}_{kg}^2, \dots, \hat{Y}_{kg}^t\} \xrightarrow{semantic} \{\hat{L}_{pre}^1, \hat{L}_{pre}^2, \dots, \hat{L}_{pre}^t\} \tag{3}$$

where $\hat{Y}_{kg}$=$\{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n\}$ is the predicted keyphrases, and $\hat{L}_{pre} = \{\hat{l}_1, \hat{l}_2, \dots, \hat{l}_{n'}\}$, n and n' are the number of predicted token of keyphrases and the number of mapped labels, t is the number of instances in the dataset.

Because of the text-to-text feature, we need raw text of each document as the input text and raw label texts of the raw document. The pre-trained text-to-text model can predict keyphrases after training processing by our specific data format like Figure 3. In this part, we need to preprocess input text data first. The training dataset for the text-to-text model should be reformatted into two parts: documents and labels. At the beginning of each document, we add a 'summarize:' prefix to guide the text-to-text model to generate keyphrases.

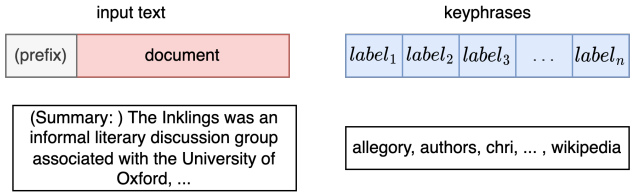

Figure 3 an example of keyphrase format for XMC dataset.

Furthermore, before the training step, we must prepare the labels corresponding to each document. So, each training record has a complete raw input text which is adhered to a prefix and a ground truth label set. After combining the training data, we feed training data into the text-to-text model for training. Otherwise, considering the length of input text is longer than the max-length of text-to-text models, we use a default configuration of truncation of text-to-text models rather than modify it for longer text to simplify more models as far as possible. Text-to-text pre-trained models can obtain contextualized more semantic information and can be easier to recall the abstract labels not in the input document but in the ground truth label set of the input text.

The other important point is the configuration of prediction. After training the text-to-text language model with our specific training data and settings, we can use the pre-trained model to generate keyphrases. The keyphrases generated by the text-to-text model contain a few labels that are the ground truth labels of input text. However, because of the feature of the text-to-text model, the model can generate redundant and very similar labels, which will influence the prediction results. We should adjust hyperparameters to avoid a mass of very similar labels in the predicted keyphrases.

3.3 Label combination

Since non-existent labels can be generated in the generating part, it is necessary to transform the keyphrases into existing labels. The non-existent labels are not useless in our results. Compared with redundant and very similar labels, the non-existent but different meanings between

labels can express abundant semantic information of input texts. With this in mind, the new method we proposed is replacing these labels with the most similar labels in the whole label set. We call the most similar label as the similarity candidate label.

For each non-existed label, we can obtain its label embedding u_{non} by BERT model. Meanwhile, the embeddings of whole labels of the dataset can be expressed as $U_L = [u_1, u_2, \dots, u_N]$, and N is the number of labels. The most similar label for one non-existent label is:

$$l_{replace} = \underset{i \in [1,N]}{\operatorname{argmax}}\left(\cos(v_{non}, u_i)\right) \quad (4)$$

Where v_{non} is the embedding of non-existent label and $l_{replace}$ is the most similar label that is used to replace a non-existent label. The cosine similarity $\cos(v_{non}, U_L)$ is the is the vector product.

Figure 4 shows the basic flow of replacement for non-existent labels. The core of combining is finding the most similar label in the whole label set to replace a non-existent label generated by the text-to-text model. Regarding the label format, we can regard a label as a short sentence that can easily use an encoder transformer model to obtain embedding vectors and compute the semantic similarity. Due to the enormous number of labels in the datasets of XMC tasks, if we compute the semantic similarity of each label with all labels in the dataset, the time cost is unacceptable. So, we choose the bi-encoder [15] as the model for computing similarity candidate labels, as shown in Figure 5.

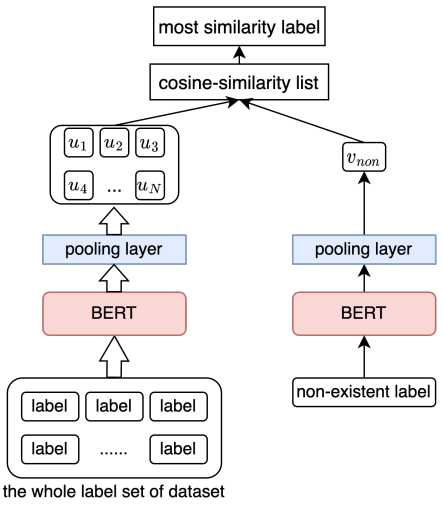

Figure 4: An example of replacement for non-existent labels. Red labels are ground truth labels of current document, and the blue label 'inkling' is non-existent label. The red label 'inklings' is the replaced label from the whole label set. Black labels are negative labels.

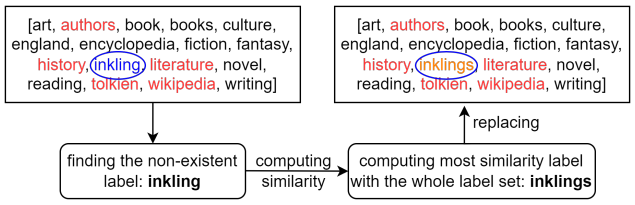

Figure 5: How to compute label cosine similarity with label set and find most similar label from the whole label set in dataset.

In the bi-encoder model, we can first compute all label embeddings of the label set and use these embeddings to calculate the cosine similarity with each non-existent label.

Furthermore, in a group of generated labels from one input text, more than one predicted and combined labels can have the same similar candidate label. We need to find the second semantically similar label for these labels to avoid the conflict of label combining.

3.4 Label Ranking

In the XMC tasks, we need to recall as many ground truth labels as possible. So, the accuracy of the labels that are recalled is crucial. Our investigation found that the sentence-transformer model's capability of embedding and semantic similarity can also be used in label ranking. Maybe this part is close to the original embedding method with a pre-trained language model. For each document, we first embed the document and the combined labels. Then we calculate the cosine similarity between the document embedding and each combined label and rank these labels by the cosine similarity score. This process is similar to the label combing part and can be expressed by:

$$\hat{L}_{rank} = sort\left(\cos\left(v_{doc}, [\hat{l}_1, \hat{l}_2, \dots, \hat{l}_N]\right)\right) \quad (5)$$

where $[\hat{l}_1, \hat{l}_2, \dots, \hat{l}_N]$ are the combined labels that have been replaced all non-existent labels, $\hat{L}_{rank}$ is ranked by cosine similarity scores between document and combined labels. Figure 6 shows an example of the ranking part of our work.

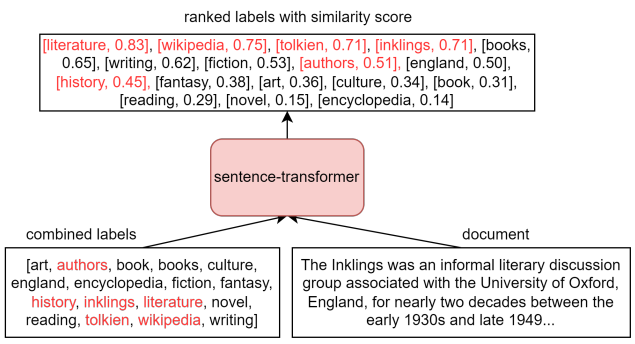

Figure 6: A example of ranking part. Each combine labels are computed the semantic similarity with input document by sentence transformer. Red means this label is the ground truth label for this document, black labels are negative labels.

Training data for the ranking model has three parts: 1. Raw input document, 2. combined labels, and 3. each score of combined labels. Instead of using labels from keyphrase generation, we use the labels from combined labels because the labels of the generating part have many kinds of noise, such as replicative labels and non-existent labels, which will disturb the accuracy of the sentence-transformer model.

Unlike the combining part, we can train the sentence-transformer with negative samples. As for the training part, the simplest method is assigning the ground truth label 1 score and the negative sample label score 0.

4. Experiments

4.1 Evaluation Metrics

Regard to XMC tasks, the P@k (Precision at k) is a very explicit value for evaluating model performance. In our works, the P@k value can be defined as:

$$P@k = \frac{1}{k} \sum_{l \in rank_k(\hat{y})}^{k} y_l \quad (6)$$

where $y \in \{0,1\}^L$ is the true labels value, $\hat{y}$ is the final output of a model. Also, l is the index in the top-k output labels.

4.2 Datasets

Since we need to compute the semantic similarity of labels with each other, it is necessary to obtain the original raw label texts and document texts, rather than the label index and BOW features given in XMC datasets. The raw input documents of XMC datasets can be found easily, but the raw label texts are hard to be found. Almost all of raw documents of XMC datasets can be found, but a significant portion of XMC datasets provide raw documents and numerical label indexes instead of real labels. Fortunately, some datasets, like EURLex-4K [4], Wiki10-31K [5], AmazonCat-13K [10], and Wikipedia-500K [10], provide the raw label texts. Therefore, our main experiments will be performed on these four datasets. Table 1 shows the statistics of the datasets from AttentionXML [17].

Table 1: The statistics of four datasets. N_{train} and N_{test} mean the number of instances of training set and testing set. N_{label} refers to the total number of labels in dataset. D_{tfidf}: the dimension of TF-IDF feature.

	N_{train}	N_{test}	N_{label}	D_{tfidf}
EUR-Lex	15,449	3,865	3,956	186,104
Wiki10-31K	14,146	6,616	30,938	101,938
AmazonCat-13K	1,186,239	306,782	13,330	203,882
WIki500K	1,779,881	769,421	501,008	2,381,304

4.3 Experiment Settings

We try more than one text-to-text models for XMC tasks in the keyphrase generation part. However, depending on different datasets, various text-to-text models have different performances. We utilize BART-base [14], T5[3], and Pegasus [12] as the text-to-text language model with 5e-5 learning rate, 2(large size) or 4(base size) training batch size, and five training epochs for all datasets. As for the combining part, we chose a bi-encoder as the model, which is used to compute the embedding of labels. Regarding the ranking part, because of the requirement of supervised data for training bi-encoder, we must use labels that are predicted by the pre-trained text-to-text model with training documents. We choose all-MiniLM-L6-v2 as the bi-encoder model with 64 batch size. We also choose cross-encoder/stsb-RoBERTa-base as the cross-encoder model with 24 batch sizes in our experiments. Both two models are trained with four epochs.

Considering the time cost, for Wiki500K and AmazonCat-13K, we do not use all training data to train the text-to-text model and ranking model. We randomly select 5% train data of Wiki500K and 10% train data of AmazonCat-13K to train the text-to-text models and ranking models.

5. Results and Analysis

We evaluate our model on four public datasets with raw documents and labels: EUR-Lex 4K, Wiki10-31K, AmazonCat-13K, and Wiki500K. Our raw document texts come from AttentionXML [17]. The raw label texts of AmzaconCat-13K, Wiki500K, and Wiki10-31K are also from Attention XML. The raw label texts of EUR-Lex are from (Johannes et al. 2010) [5].

The performance metric is P@k (Precision@k), a simple and valuable evaluation metric used in XMC tasks. Table 2 shows the performance of our method in XMC tasks and the comparisons with other methods.

Since there is no previous method based on a text-to-text model of keyphrase generation for XMC tasks, we compare our method with DiSMEC, AnnexML, Parabel, XML-CNN, AttentionXML, and LightXML. The results of all the baselines are from AttentionXML and LightXML. Considering the fair comparison, all the results on our method in table 2 are by facebook/bart-base model except Wiki500(using Pegasus), because BART with bi-encoder obtained the high score in Wiki10-31K, and AmazconCat-13K.

We also compared with an up-to-date XMC research XRR

[11], which is a complex system and uses semantic and statistics information for XMC tasks. Although XRR method provide overwhelming performance, our method shows a highest score of P@1 in wiki10-31K.

Our method outperformed the almost traditional one-vs-all and word embedding methods in EUR-Lex, Wiki10-31K, and Wiki500K. Moreover, our method shows the great performance in P@3 of Wiki500K rank only second to XRR.

Table 2: The comparison with the results of all baseline and our method.

EUR-Lex			
	P@1	P@3	P@5
AnnexML	79.66	64.94	53.52
DiSMEC	83.21	70.39	58.73
Parabel	82.12	68.91	57.89
XT	79.17	66.80	56.09
Bonsai	82.30	69.55	58.35
XML-CNN	75.32	60.14	49.21
AttentionXML	87.12	73.99	61.92
LightXML	87.63	75.89	63.36
XRR	**87.96**	**78.88**	**68.52**
our method	84.09	69.31	53.07
Wiki10-31K			
	P@1	P@3	P@5
AnnexML	86.46	74.28	64.20
DiSMEC	84.13	74.72	65.94
Parabel	84.19	72.46	63.37
XT	83.66	73.28	64.51
Bonsai	84.52	73.76	64.69
XML-CNN	81.41	66.23	56.11
AttentionXML	87.47	78.48	69.37
LightXML	89.45	78.96	69.85
XRR	89.54	**85.38**	**81.34**
our method	**89.62**	79.73	70.12
AmazonCat-13K			
	P@1	P@3	P@5
AnnexML	93.54	78.36	63.30
DiSMEC	93.81	79.08	64.06
Parabel	93.02	79.14	64.51
XT	92.50	78.12	63.51
Bonsai	92.98	79.13	64.46
XML-CNN	93.26	77.06	61.40
AttentionXML	95.92	82.41	67.31
LightXML	96.77	84.02	68.70
XRR	**97.01**	**84.70**	**69.39**
our method	91.61	74.93	56.04
Wiki500K			
	P@1	P@3	P@5
AnnexML	64.22	43.15	32.79
DiSMEC	70.21	50.57	39.68
Parabel	68.70	49.57	38.64
XT	65.17	46.32	36.15
Bonsai	69.26	49.80	38.83
XML-CNN	-	-	-
AttentionXML	76.95	58.42	46.14
LightXML	77.78	58.85	45.57
XRR	**84.58**	**66.41**	**54.32**
our method	75.45	59.16	45.86

However, the performance of AmazonCat-13K could be better because of its different categories. Simultaneously, our method is close to the new deep-learning-based methods like AttentionXML and LightXML.

5.1 Difference combination of keyphrases generation and label ranking.

In relative work, we have shown the difference between BART, T5, and Pegasus. We also run different models, consisting of different text-to-text models and sentence-transformers in EUR-Lex 4K. The results are given in Table 3. Pegasus with cross-encoder obtained the highest score of P@1, but the P@3 and P@5 are not the best scores. T5 with cross-encoder and Pegasus obtained the highest score of P@3 and P@5 separately. However, Pegasus (with 568M parameters) has a longer training time and larger model size compared with BART-base and T5-base.

Table 3: The comparison with the results of all baseline and our method. Cross-encoder: using cross-encoder to replace bi-encoder ranking part.

EUR-Lex	P@1	P@3	P@5
BART	84.09	69.31	53.07
BART+cross-encoder	84.11	69.84	53.16
T5	83.54	70.22	55.19
T5+cross-encoder	84.91	**71.40**	55.32
Pegasus	83.70	69.14	**56.67**
Pegasus+cross-encoder	**85.25**	70.77	55.92

6. Conclusion and Future Work

We proposed a brand-new method for extreme multi-label classification tasks in this paper. Compared with existing methods, there are two innovative points in our method. The first is using keyphrase generation to replace classical label recalling. The second is utilizing the semantic representation to evaluate the relevance between labels and the original document based on raw label texts. We evaluate our method with various combinations in four public datasets of the XMC task. Our method's results outperformed most of traditional XMC methods and were close to SOTA. Compared with SOTA, our method utilizes a different way of label recalling, and the subsequent processing combining and ranking parts are also based on semantical similarities.

In future work, we can try more modified and customized text-to-text models for keyphrase generation, to enhance the performance of keyphrase generation. On the other hand, the ranking performance still has an ample room for

improvement.

7. Acknowledgment

Our work got help from Hao Shen, who have graduated from Iwaihara Lab, Waseda University and provided a partial idea and prototype of our work.

References

[1] Diya, A. and Mizuho, I., 2022. Keyphrase generation by utilizing BART finetuning and BERT-based ranking. In *DEIM Forum*.

[2] Zubiaga, A., 2012. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv:1202.5469*.

[3] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, *21*(140), pp.1-67.

[4] Loza Mencía, E. and Fürnkranz, J., 2008, September. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 50-65). Springer, Berlin, Heidelberg.

[5] Loza Mencía, E. and Fürnkranz, J., 2010. Efficient multilabel classification algorithms for large-scale problems in the legal domain. In *Semantic Processing of Legal Texts* (pp. 192-215). Springer, Berlin, Heidelberg.

[6] Yen, I.E.H., Huang, X., Ravikumar, P., Zhong, K. and Dhillon, I., 2016, June. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International conference on machine learning* (pp. 3069-3077). PMLR.

[7] Devlin, J., Ming-Wei C., Kenton L., and Kristina T. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[8] Hartigan, J.A. and Wong, M.A., 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, *28*(1), pp.100-108.

[9] Liu, J., Chang, W.C., Wu, Y. and Yang, Y., 2017, August. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval* (pp. 115-124).

[10] McAuley, J. and Leskovec, J., 2013, October. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems* (pp. 165-172).

[11] Xiong, J., Yu, L., Niu, X. and Leng, Y., 2023. XRR: Extreme multi-label text classification with candidate retrieving and deep ranking. *Information Sciences*, *622*, pp.115-132.

[12] Zhang, J., Zhao, Y., Saleh, M. and Liu, P., 2020, November. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning* (pp. 11328-11339). PMLR.

[13] Bhatia, K., Jain, H., Kar, P., Varma, M. and Jain, P., 2015. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems*, *28*.

[14] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L., 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

[15] Reimers, N. and Gurevych, I., 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

[16] Dekel, O. and Shamir, O., 2010, March. Multiclass-multilabel classification with more classes than examples. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (pp. 137-144). JMLR Workshop and Conference Proceedings.

[17] You, R., Zhang, Z., Wang, Z., Dai, S., Mamitsuka, H. and Zhu, S., 2019. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in Neural Information Processing Systems*, *32*.

[18] Babbar, R. and Schölkopf, B., 2017, February. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the tenth ACM international conference on web search and data mining* (pp. 721-729).

[19] Jiang, T., Wang, D., Sun, L., Yang, H., Zhao, Z. and Zhuang, F., 2021, May. Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 9, pp. 7987-7994).

[20] Siblini, W., Kuntz, P. and Meyer, F., 2018, July. Craftml, an efficient clustering-based random forest for extreme multi-label learning. In *International Conference on Machine Learning* (pp. 4664-4673). PMLR.

[21] Ju, Y. and Iwaihara, M., 2022. Unsupervised Keyphrase Generation by Utilizing Masked Words Prediction and Pseudo-label BART Finetuning. In *International Conference on Asian Digital Libraries* (pp. 21-34). Springer, Cham.

[22] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[23] Prabhu, Y. and Varma, M., 2014, August. 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* (pp. 263-272).

[24] Prabhu, Y., Kag, A., Harsola, S., Agrawal, R. and Varma, M., 2018, April. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference* (pp. 993-1002).

[25] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R. and Le, Q.V., 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.