LICHEE: Improving Language Model Pre-training with Multi-grained Tokenization

Weidong Guo[1*], **Mingjun Zhao**[2*], **Lusheng Zhang**[1*], **Di Niu**[2], **Jinwen Luo**[1], **Zhenhua Liu**[1], **Zhenyang Li**[1], **Jianbo Tang**[1]
[1]Platform and Content Group, Tencent
[2]University of Alberta
{weidongguo, lshzhang, jamsluo, edinliu, nickzyli, jianbotang}@tencent.com,
{zhao2, dniu}@ualberta.ca

Abstract

Language model pre-training based on large corpora has achieved tremendous success in terms of constructing enriched contextual representations and has led to significant performance gains on a diverse range of Natural Language Understanding (NLU) tasks. Despite the success, most current pre-trained language models, such as BERT, are trained based on single-grained tokenization, usually with fine-grained characters or sub-words, making it hard for them to learn the precise meaning of coarse-grained words and phrases. In this paper, we propose a simple yet effective pre-training method named *LICHEE* to efficiently incorporate multi-grained information of input text. Our method can be applied to various pre-trained language models and improve their representation capability. Extensive experiments conducted on CLUE and SuperGLUE demonstrate that our method achieves comprehensive improvements on a wide variety of NLU tasks in both Chinese and English with little extra inference cost incurred, and that our best ensemble model achieves the state-of-the-art performance on CLUE benchmark competition.

1 Introduction

Pre-trained language models (PLMs) such as GPT (Radford et al., 2018), BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) have become enormously popular and achieved great success on diverse natural language understanding tasks, such as sentiment analysis, question answering, and language inference. These models usually utilize a transformer architecture (Vaswani et al., 2017) to capture the dependencies between tokens in the input text, to model the language information, and to learn contextual representations. It is first pretrained based on large-scale unlabeled corpora, and subsequently fine-tuned based on the labeled data from downstream tasks.

In many NLU applications, tokenization often affects the performance and needs to be chosen carefully. The input tokens for pre-trained language models are usually fine-grained, e.g., words and sub-words for English and characters for Chinese. Compared with coarse-grained tokens such as phrases, the advantage of fine-grained tokens is that they form a smaller vocabulary, yielding abundant training samples per token, and thus alleviating the data sparsity issue and out-of-vocabulary (OOV) problem (Li et al., 2019). However, even trained on large corpora, it is still hard for language models pre-trained with fine-grained tokens to learn the correct attention boundaries of larger semantic units in many languages (Zhang and Li, 2020).

To obtain a more accurate model, prior studies attempt to incorporate coarse-grained information into models trained with fine-grained tokenization by masking sequences of consecutive tokens in the pre-training stage (Joshi et al., 2020; Cui et al., 2019). Zhang and Li (2020) propose AMBERT, a Siamese network based on BERT to handle multi-grained input text, and uses two encoders with shared weights to separately encode fine-grained tokens and coarse-grained tokens into two sequences of contextualized representations. Despite its effectiveness, the inference cost of AMBERT almost doubles that of the original BERT due to the dual-encoder structure, which is often unacceptable in industrial scenarios.

In this paper, we propose a novel method named *LICHEE* designed to efficiently leverage the input information at multiple levels of granularity in the pre-training stage in order to enhance the representation ability of PLMs. Unlike AMBERT that encodes the fine-grained and coarse-grained tokens with two encoders, which significantly increases the inference cost, in *LICHEE* the fusion of multi-

[*] Equal contribution.

grained information of input text happens at the embedding level, which requires no change on the original model structure of the PLM, and thus induces little extra inference cost when applied in online NLP applications. Specifically, *LICHEE* first pre-processes the input text into fine-grained and coarse-grained tokens, which are passed through two embedding layers, respectively, to derive their corresponding vector representations. Both vector representations are then merged via pooling to form the *multi-grained embedding vector*, which serves as the input to the PLM encoder. Finally, the enhanced contextual representations generated by the PLM encoder, with both fine-grained and coarse-grained information incorporated, are obtained and used for downstream tasks.

We have applied *LICHEE* to enhance multiple different pre-trained language models, including BERT (Devlin et al., 2019), ALBERT (Lan et al., 2019), and GPT (Brown et al., 2020), and conducted extensive evaluation of the resulted language models on Chinese natural language understanding (NLU) tasks evaluated by CLUE (Liang Xu, 2020) benchmarks. Results show that with *LICHEE*, the resulted pre-trained language models significantly outperform their single-grained counterparts on almost all tasks, by taking advantage of multi-grained information to effectively and efficiently produce more accurate representations.

In addition, we also participated in the CLUE benchmark competition with our best ensemble model built upon a collection of *LICHEE*-enhanced BERT-large models, and achieved the state-of-the-art performance of an average score of 80.42 (as of January 8, 2021) over 9 different Chinese NLU tasks, as well as the best scores on two individual tasks: IFLYTEK and CSL.

Moreover, we have also conducted English natural language understanding experiments based on SuperGLUE (Wang et al., 2019a) benchmarks. Significant improvements are observed when *LICHEE* is employed in the pre-training stage, which demonstrates that the proposed pre-training method is generally effective in different language settings.

2 Related Work

In this section, we give a brief overview of some popular pre-trained language models and studies on the training techniques related to tokenization.

Pre-trained language models are pre-trained on large unsupervised corpora and aim to produce meaningful representations for each input token not only considering the meaning of itself, but also with its surrounding contexts anticipated. ELMo (Peters et al., 2018) is one of the first pre-trained language models based on bidirectional LSTMs which produces the contextual representation of each token by concatenating its left-to-right and right-to-left representations. GPTs (Radford et al., 2018, 2019; Brown et al., 2020) leverage the powerful Transformer (Vaswani et al., 2017) to build an auto-regressive language model predicting the next token given its history context. BERT (Devlin et al., 2019) is a bidirectional auto-encoding language model also based on transformer. It consists of two pre-training objectives: masked language model (MLM) and next sentence prediction (NSP). Yang et al. (2019) point out the discrepancy of the pre-training and fine-tuning stage of BERT due to the masking symbol, and propose a permutation language model called XLNet (Yang et al., 2019).

The great popularity of BERT draws many researchers to make improvements on the architecture. RoBERTa (Liu et al., 2019) improves several training details of BERT including dynamic masking and the removal of the NSP pre-training task. ALBERT (Lan et al., 2019) reduces the model parameters with cross-layer weight sharing and accelerates the training process. ELECTRA (Clark et al., 2019) proposes a new token detection task and adopts a generator-discriminator framework to pre-train the language model.

Although most pre-trained language models are built on fine-grained tokenization, coarse-grained information proves to be helpful to the model performance. Cui et al. (2019) propose a masking scheme called "whole word masking" (WWM) for Chinese BERT, where the consecutive characters belonging to the same word are masked together. In ERNIE (Sun et al., 2019), knowledge graphs are added to enhance the model, and entity level masking is used during the pre-training, which is beneficial for language understanding tasks. SpanBERT (Joshi et al., 2020) proposes to mask random spans instead of random tokens, and adopts a new span boundary objective task to replace the next sentence prediction task in the pre-training. Instead of focusing on the masking scheme, AMBERT (Zhang and Li, 2020) proposes to adopt two encoders with shared parameters to learn the representations of fine-grained and coarse-grained tokens in parallel. However, even that the weight sharing setting reduces the number of model parameters, the dual-encoder structure of AMBERT induces twice the inference cost, which remains a huge issue when deployed in online applications.

Different from AMBERT, our work merges the fine-grained and coarse-grained tokenization at embedding level, and achieves significant performance gains with little additional computation costs.

3 Methodology

In this section, we present *LICHEE*, the general multi-grained framework for language model pre-training, and its detailed implementation, including the pre-training methods for both auto-regressive and auto-encoding tasks and fine-tuning details.

3.1 Model Architecture

Figure 1 gives an overview of *LICHEE* where the input information from multiple granularities is leveraged to enhance the representation ability for many pre-trained language models.

The framework takes in text sequences as input which are tokenized into token sequences. In this paper, we keep two vocabularies and use two tokenizers to perform fine-grained and coarse-grained tokenizations, where items in vocabularies are selected based on their token frequencies in pre-training corpora. Also, the definitions of "fine grain" and "coarse grain" vary across languages. For example, in English, words and phrases are often used as the fine-grained and coarse-grained tokens respectively. And in Chinese, characters and words are used instead. Officially, for a given input text sequence T, we use t_i^f to denote the i-th fine-grained token and $t_{j\text{-}k}^c$ to denote a coarse-grained token that is composed of fine-grained tokens $\{t_j^f, ..., t_k^f\}$ between j and k. For example, in figure 1, the coarse-grained token "New York Times" is composed of the first, sencond, and third fine-grained tokens, and is denoted as $t_{1\text{-}3}^c$.

After tokenization, two separate embedding layers are used to map the tokenized tokens to their vector representations. Specifically, each fine-grained token t_i^f is passed into a fine-grained embedding layer to produce the fine-grained embedding vector $\vec{e}_i^f \in \mathcal{R}^d$ of the token, where d denotes the dimension of the fine-grained embedding. Similarly, the coarse-grained embedding $\vec{e}_{j\text{-}k}^c \in \mathcal{R}^d$ is derived with the same dimension d by feeding token $t_{j\text{-}k}^c$ to the coarse-grained embedding layer, shown as:

$$\begin{aligned} \vec{e}_i^f &= embedding_{fine}(t_i^f), \\ \vec{e}_{j\text{-}k}^c &= embedding_{coarse}(t_{j\text{-}k}^c). \end{aligned} \tag{1}$$

For each token t_i^f, we construct its multi-grained embedding vector $\vec{e}_i \in \mathcal{R}^d$ by performing a max-pooling operation on the derived fine-grained embedding $\vec{e}_i^f$ and the coarse-grained embedding $\vec{e}_{j\text{-}k}^c$ of its corresponding coarse-grained token $t_{j\text{-}k}^c$:

$$\vec{e}_i = \textit{max-pool}(\vec{e}_i^f, \vec{e}_{j\text{-}k}^c), \tag{2}$$

where $j \leq i \leq k$. Note that d is equal to the original embedding dimension of the single-grained PLM, to prove that the performance gain is contributed to the introduction of multi-grained information other than modified model structure.

Finally, the combined embedding vectors $\vec{\boldsymbol{e}}$ are fed into the PLM encoder to construct the final contextualized representations $\vec{\boldsymbol{h}}$ enhanced with multi-grained information:

$$\vec{\boldsymbol{h}} = encode(\vec{\boldsymbol{e}}). \tag{3}$$

3.2 Pre-training

We have applied *LICHEE* on both auto-regressive and auto-encoding PLMs, such as GPT and BERT.

For ***auto-regressive PLMs***, the pre-training task is Next Token Prediction which aims to predict the next token t_i based on its previous context $t_{<i}$, by optimizing the following objective function

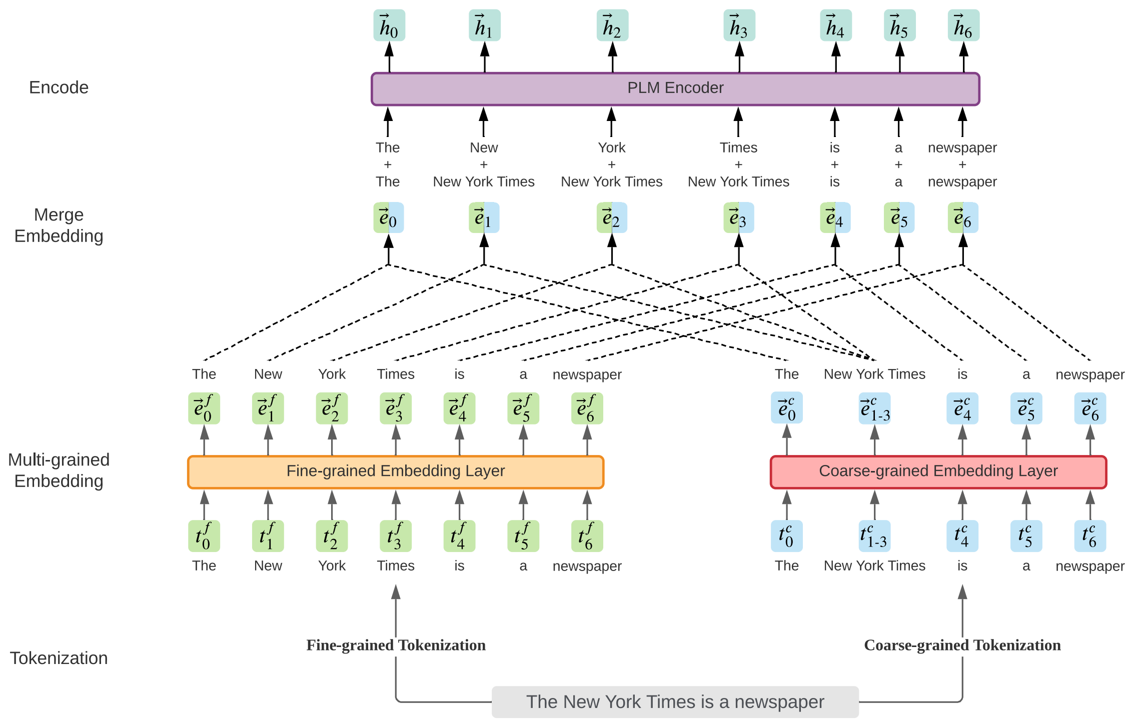
$$\min_\theta - \sum_i \log p_\theta(t_i | t_{<i}), \tag{4}$$

where the conditional probability p_θ is modeled with a network with parameter θ.

In our framework, we adjust the objective function to include both fine-grained context $t_{<i}^f$ and coarse-grained context $t_{<i}^c$, shown as:

$$\min_\theta - \sum_i \log p_\theta(t_i | t_{<i}^f, t_{<i}^c). \tag{5}$$

Note that when making predictions on any token within a coarse-grained span $t_i \in t_{j\text{-}k}^c$, the token embedding $\vec{e}_i$ will cause information leakage as it involves the coarse-grained token embedding $\vec{e}_{j\text{-}k}^c$ which contains information beyond the history context. For example, in the case illustrated in figure 1, the prediction on token "York" should not rely



Encode
PLM Encoder
Merge Embedding
Multi-grained Embedding
Fine-grained Embedding Layer
Coarse-grained Embedding Layer
Tokenization
Fine-grained Tokenization
Coarse-grained Tokenization
The New York Times is a newspaper
The
New
York
Times
is
a
newspaper
New York Times

Figure 1: The overall structure of our proposed pre-training framework *LICHEE*. Fine-grained and coarse-grained tokens are first derived from the input text by tokenization, and separately passed into two individual embedding layers. The multi-grained embedding vectors are acquired by taking a max-pooling on the fine-grained and coarse-grained embedding vectors, and are fed into the PLM encoder to extract the final contextualized representations.

on token "New" and its embedding $\vec{e}_1$ as it discloses the entire information of the coarse-grained token of "New York Times" by the coarse-grained embedding $\vec{e}^c_{1\text{-}3}$. Therefore, we can only exploit the context before the start position of the coarse-grained token to make predictions, illustrated as:

$$\min_{\theta} - \sum_{j \le i \le k} \log p_\theta(t_i | t^f_{<j}, t^c_{<j}), \qquad (6)$$

where j and k are the start and end positions of the coarse-grained token.

For ***auto-encoding PLMs***, we only include Masked Language Modeling (MLM) task in the pre-training process, as Next Sentence Prediction (NSP) task is shown to have no benefits indicated in many recent studies (Lan et al., 2019; Liu et al., 2019; Zhang and Li, 2020). In MLM, 15% of the tokens are randomly selected and substituted with a set of tokens, in which 80% are replaced with *[MASK]* token, 10% are replaced with random tokens, and 10% stay unchanged.

The objective is to recover the masked tokens $T^m \subset T$ from the altered text input sequence $\tilde{T}$:

$$\min_{\theta} - \sum_{t^m \in T^m} \log p_\theta(t^m | \tilde{T}). \qquad (7)$$

In our framework, we propose to exploit the multi-grained information of the input in the MLM task, shown as:

$$\min_{\theta} - \sum_{t^m \in T^m} \log p_\theta(t^m | \tilde{T}^f, \tilde{T}^c), \qquad (8)$$

where $\tilde{T}^f$ and $\tilde{T}^c$ stand for the fine-grained and coarse-grained altered input text.

Similar to the strategy deployed in auto-regressive PLMs, we apply a masking strategy that when a fine-grained token t^f_i is to be masked, its corresponding coarse-grained token $t^c_{j\text{-}k}$ and all the fine-grained tokens $t^f_j, ..., t^f_k$ belonging to it are also masked, in order to avoid information leakage from the multi-grained embeddings.

3.3 Fine-tuning

In fine-tuning of downstream tasks, we append the special tokens (*[CLS]*, *[SEP]*) to both fine-grained and coarse-grained vocabularies. In sentence-level classification tasks, *[CLS]* is attached to the start of input sequences in auto-encoding PLMs like BERT, and to the end of the input in auto-regressive PLMs like GPT. Its multi-grained contextualized representation $\vec{h}_{[CLS]}$ is used to represent the whole input sequence and is passed into a projecting layer for the final prediction.

Similarly, for tasks that include token-level span detection, such as Question Answering, the contextual representation $\vec{h}_i$ for each token t_i is extracted and utilized in the task.

4 Experiments

We have carried out extensive experiments on various natural language understanding tasks on both Chinese and English datasets. In the following section, we will first introduce the pre-training datasets used in our evaluation and provide the implementation details of our framework. And we demonstrate the effectiveness of *LICHEE* by conducting comprehensive experiments on various Chinese NLU datasets with multiple different PLMs, and compare our method with other baseline methods. Next, we perform a thorough ablation study to evaluate different approaches of integrating input text information from multiple granularities. Finally, we adopt *LICHEE* to an English BERT to verify its efficacy on English NLU tasks.

4.1 Pre-Training Datasets

For Chinese language, there is no commonly used corpus for pre-training language models. We utilize a large corpus consisting of 450G text from a wide range of popular Chinese applications including Kandian, Zhihu, Wechat, and Weibo, in various fields of news, wiki, and blogs.

Similar to most Chinese PLMs, characters are used as fine-grained tokens due to the language nature of Chinese. For coarse-grained tokens, We use QQSeg which is a segmentation tool with an open API to perform segmentation on text, and the segmented words are treated as coarse-grained tokens. For the construction of vocabularies, we follow Google's Chinese BERT and include $21,128$ tokens in the fine-grained vocabulary. And in the coarse-grained vocabulary, we calculate the token frequencies and trimmed out tokens with frequency lower than 8, resulting in $210,946$ tokens. Note that in order to alleviate the out-of-vocabulary (OOV) problem, all tokens in the fine-tuned vocabulary are also included in coarse-grained vocabulary.

For English, a corpus with 6.2 million documents (18.9G compressed text) from Wikipedia is leveraged to pre-train the model. We first perform sub-word tokenization with BPE algorithm (Sennrich et al., 2015) on the English text, where the produced words and sub-words constitute the fine-grained vocabulary of $28,996$ tokens. In the coarse-grained vocabulary, we treat high-frequency words as coarse-grained tokens, resulting in $136,630$ tokens in total, which also include all tokens in the fine-grained vocabulary for the OOV concern.

4.2 Benchmarks

The evaluation of the pre-trained models is conducted on various downstream NLU tasks. In our experiments, all the Chinese PLMs are evaluated on Chinese Language Understanding Evaluation (CLUE) (Liang Xu, 2020) which is a comprehensive language understanding benchmark developed for Chinese containing 9 natural language understanding tasks. Within the 9 tasks, there are two single-sentence classification tasks that are TNEWS and IFLYTEK, four sentence-pair classification tasks that are AFQMC, OCNLI, CLUEWSC and CSL, and three question answering tasks that are CMRC2018, CHID, and C3. Note that OCNLI has replaced CMNLI since Oct 22, 2020. We compare the model performance by reporting the performance score of each task and the average score of all tasks.

For English tasks, we use the SuperGLUE benchmarks (Wang et al., 2019a) which is an extension of GLUE (Wang et al., 2019b) consisting of a collection of 8 NLU tasks of higher difficulty for comprehensively evaluating the performance of English PLMs. SuperGLEU contains a word sense disambiguation task (WiC), two textual entailment tasks (CB and RTE), two reasoning tasks (COPA and WSC), and three question answering tasks (BoolQ, MultiRC, and ReCoRD).

4.3 Experiment Setup

In order to demonstrate the general applicability and effectiveness of our framework, we have implemented three different pre-trained language models with our method including BERT, ALBERT and GPT, and compare the performances with their corresponding single-grained baseline methods.

For BERT and ALBERT, we follow the "base" structure in (Devlin et al., 2019) with an encoder of 12 layers. And the GPT model in our experiment is also made up of a 12-layer transformer decoder.

Model	Avg.	TNEWS	IFLYTEK	AFQMC	OCNLI	CLUEWSC	CSL	CMRC2018	CHID	C3
	-	acc.	acc.	acc.	acc.	acc.	acc.	EM.	acc.	acc.
BERT	71.12	66.62	60.64	71.74	73.45	72.92	84.01	73.08	75.52	62.08
BERT-*LICHEE*	**73.92**	**67.94**	**60.94**	**73.65**	**75.85**	**81.03**	**84.51**	**75.84**	**77.65**	**67.84**
ALBERT	67.27	64.45	57.54	**71.35**	69.19	68.80	83.00	68.06	68.97	54.04
ALBERT-*LICHEE*	**69.30**	**66.31**	**58.29**	70.95	**71.05**	**70.39**	**83.31**	**72.87**	**71.93**	**58.65**
GPT	67.41	67.52	60.84	69.83	70.91	63.76	83.12	62.53	73.31	54.84
GPT-*LICHEE*	**68.73**	**68.40**	**61.06**	**70.00**	**72.01**	**66.01**	**83.23**	**64.57**	**74.02**	**59.27**

Table 1: Comparison of the model performances on the CLUE tasks. BERT-*LICHEE*, ALBERT-*LICHEE* and GPT-*LICHEE* stand for the multi-grained version of the model with our method incorporated. The average score of the nine CLUE tasks are also given.

Model	Avg.	TNEWS	IFLYTEK	AFQMC	OCNLI	CLUEWSC	CSL	CMRC2018	CHID	C3
	-	acc.	acc.	acc.	acc.	acc.	acc.	EM.	acc.	acc.
Archer-24E-SINGLE	79.19	69.54	62.27	**77.26**	**83.57**	90.00	85.73	75.65	85.66	**83.04**
roberta_selfrun	79.46	69.10	63.92	76.09	80.40	**93.10**	87.27	79.20	88.80	77.29
UER-ensemble	79.64	**72.20**	64.00	76.82	80.80	90.35	85.83	79.15	86.03	81.60
BERTs	79.66	69.94	63.92	76.77	82.09	88.97	86.77	**80.50**	**89.51**	78.44
LICHEE-ensemble	**80.06**	70.50	**64.15**	76.98	81.30	90.69	**87.40**	79.80	87.51	82.22

Table 2: Top-5 models on the CLUE benchmark leaderboard where our ensemble model achieves the state-of-the-art performance on the averaged CLUE score. These results are grabbed from the official CLUE website[1] on Jan 8, 2021.

Then, we apply the following training setting to the training process of all three models. For better scalability in large batch, we adopt LAMB (You et al., 2019) to replace Adam (Kingma and Ba, 2014) as the optimizer with a batch size of 768 and a learning rate of $2e-4$. We first train the model for 1M steps using 128 as the maximum sequence length, and increase the maximum length to 512 for another 100k steps, for better capturing the long distance dependencies. To enhance the training efficiency, we adopt mix-precision training technique (Micikevicius et al., 2017) during pre-training, which are performed on 4 Nvidia V100 gpus.

We have also implemented a *LICHEE*-enhanced ensemble model based on BERT-large to participate in the CLUE benchmark competition. During training, we adapt the batch size to $1,024$ and the maximum sequence lengths at the first and second stage are set to 256 and 512. And 64 Nvidia V100 gpus are used to train the model.

For the evaluation of each task, we derive 6 results with different random seeds and report the average performance in this paper.

4.4 Main Results

In table 1, we adopt our multi-grained pre-training method on three pre-trained language models: BERT, ALBERT, and GPT, and compare them with their single-grained baselines on CLUE benchmark. From the results, we can see that our method achieves significant performance gains by exploiting the multi-grained information of the text input. The averaged CLUE scores of our multi-grained BERT-*LICHEE*, ALBERT-*LICHEE* and GPT-*LICHEE* are 73.92, 69.30 and 68.73 respectively, producing significant absolute improvements of 2.80, 2.03, and 1.32 compared to their single-grained baseline models. Aside from the improvement on the averaged CLUE score, it is also worth to mention that our multi-grained BERT-*LICHEE* and GPT-*LICHEE* outperforms their single-grained baselines on all 9 NLU tasks in CLUE, while the ALBERT-*LICHEE* model also beat the single-grained ALBERT in 8 out of 9 tasks, which provides strong evidence that the benefits of our method are generally applicable to different pre-trained language models and diverse NLU tasks.

In order to further investigate the potential of *LICHEE*, we apply it on an ensemble model based on BERT-large and participate in the CLUE benchmark competition. As demonstrated in table 2, our method outperforms all other candidates on the average score of 9 CLUE tasks by a significant margin, and also achieves the state-of-the-art performance on two individual NLU tasks of IFLY-

[1]https://www.cluebenchmarks.com/rank.html

Model (BERT)	**Avg.**	**TNEWS**	**IFLYTEK**	**AFQMC**	**OCNLI**	**CLUEWSC**	**CSL**	**CMRC2018**	**CHID**	**C3**
	-	acc.	acc.	acc.	acc.	acc.	acc.	EM.	acc.	acc.
SG	71.12	66.62	60.64	71.74	73.45	72.92	84.01	73.08	75.52	62.08
SG (WWM)	72.24	66.87	60.55	72.62	74.41	74.07	84.12	75.22	77.74	64.60
MG (CAT 384+384)	72.86	**68.11**	61.09	72.33	75.08	75.26	84.48	75.35	77.84	66.17
MG (CAT 256+512)	72.94	67.63	**61.55**	71.96	74.97	76.54	84.16	75.31	78.17	66.15
MG (CAT 512+256)	73.08	67.88	61.06	73.07	75.84	74.45	**84.74**	74.44	**78.29**	**67.91**
MG (MEAN)	73.22	67.85	60.99	73.44	**75.97**	76.31	84.52	75.54	77.84	66.53
LICHEE	**73.92**	67.94	60.94	**73.65**	75.85	**81.03**	84.51	**75.84**	77.65	67.84

Table 3: Ablation study of different pre-training strategies with BERT model on CLUE dataset. Two single-grained (SG) baselines and five multi-grained (MG) methods (*LICHEE* and its variants) with different ways of integrating the fine-grained and coarse-grained representations are evaluated.

TEK and CSL. This results further proves that our multi-grained pre-training method is able to bring significant improvements on the representation ability of language models and is generally effective to a wide range of downstream NLU tasks.

The reason of *LICHEE*'s success is that we adopt a multi-grained pre-training strategy to model the contextual information of the input text to leverage the advantages from both granularities, where fine-grained token representations are easier to learn considering the sufficient training samples, and coarse-grained tokens are more complete as lexical units and provide more accurate contextual information. Furthermore, in our framework, the combination of the multi-grained information is realized on the embedding level so that we can keep the model structure unaltered, showing that the benefits are achieved entirely through the information gains caused by multi-grained pre-training other than model-level modifications.

4.5 Ablation Analysis

We have conducted ablation analysis on CLUE benchmarks with BERT, to evaluate the impact of our multi-grained design, as well as perform a comprehensive study on the different methods of integrating the multi-grained embedding. Table 3 lists the performance of model variants with different training strategies, including two single-grained methods and five multi-grained methods.

The original single-grained BERT whose masking scheme is solely based on fine-grained tokens gives an average CLUE score of 71.12. The Whole Word Masking (WWM) technique (Cui et al., 2019) performs masking operations on continuous fine-grained tokens that form a coarse-grained token and improves the performance to 72.24. Note that although WWM utilizes coarse-grained token boundary information during the masking operations, it does not explicitly train representations for coarse-grained tokens. Therefore, we treat WWM also as a single-grained pre-training method.

For multi-grained pre-training methods, we have conducted experiments to explore five different approaches of combining embedding representations of fine-grained and coarse-grained tokens, including concatenating the embedding vectors with different dimension settings, and integrating them with mean-pooling and max-pooling. For the concatenation approaches, we keep the dimension of the concatenated multi-grained embeddings to 768 to align with the baseline models, and apply three settings to adjust the dimensions of fine-grained and coarse-grained embedding correspondingly to (384, 384), (256, 512) and (512, 256). Empirically, we discover that the three concatenation settings achieve similar performances, while having larger embedding vectors for fine-grained tokens and smaller embedding vectors for coarse-grained tokens produces a slightly better performance of 73.08 average CLUE score.

Exploiting mean-pooling to integrate the multi-grained information gives more performance gains compared with concatenation methods and reaches 73.22 average CLUE score, which may be attributed to the greater number of embedding parameters, as pooling methods do not require a shrink on the embedding dimension and allow both fine-grained and coarse-grained embedding dimension to stay 768. Finally, *LICHEE* with the max-pooling incorporated outperforms all the fore-mentioned approaches, attains an overall score of 73.92, and achieves the best score on 3 out of 9 CLUE tasks, due to its capability of extracting more representative features. Especially for the task of CLUEWSC, *LICHEE* acquires an accuracy of 81.03 while the second best method only reaches 76.54. We believe this is because the small training set of CLUEWSC

Model	Avg.	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC
	-	acc.	acc.	acc.	EM.	EM.	acc.	acc.	acc.
BERT-WWM	63.64	77.13	79.76	62.83	24.88	65.20	70.88	64.50	63.94
BERT-*LICHEE*	**65.53**	**77.98**	**88.21**	**63.00**	**25.41**	**67.50**	**71.91**	**65.45**	**64.81**

Table 4: Comparison between our multi-grained BERT-*LICHEE* and the single-grained BERT-WWM on SuperGLUE tasks.

Model	FLOPs	Speedup
BERT	43.5B	1.0x
AMBERT	87.0B	0.5x
LICHEE	43.5B	1.0x

Table 5: Comparison of FLOPs and speedup among the single-grained BERT, AMBERT, and our method.

with only 532 examples makes it more dependent on powerful pre-trained representations, so that the advantage of the max-pooling method is amplified.

Overall, we can see from table 3 that all multi-grained pre-training methods outperform the single-grained baselines by a significant margin, which again proves that our idea of incorporating multi-grained information during the pre-training phase is efficacious and can benefit model performance considerably.

4.6 Inference Speed Analysis

We have also studied the inference speed of *LICHEE* and compare it with the original single-grained BERT and another multi-grained method AMBERT.

Table 5 gives a brief comparison in terms of FLOPs and speedup, tested on a binary classification task with 512 sequence length. FLOPs indicates the number of floating-point operations that the model performs for a single process, where generally speaking, the higher the model's FLOPs is, the slower the inference speed will be.

We can see that the FLOPs of the AMBERT is 87.0 billion, twice the number of the single-grained BERT. It means the inference time of AMBERT is almost doubled, which can cost a lot more time and resources, and often can be unacceptable for real-world applications. Meanwhile, our multi-grained method produces a model with 43.5 billion FLOPs with a negligible increase compared with the single-grained baseline, because the additional operations only include an embedding lookup operation for coarse-grained tokens and a max-pooling operation to integrate the fine-grained and coarse-grained embedding vectors. In summary, *LICHEE* can produce significant performance gains with negligible extra inference time needed.

4.7 English Tasks

We have also conducted experiments on SuperGLUE benchmarks to evaluate *LICHEE* on English language tasks, and compared it with the single-grained baseline: BERT-WWM (Cui et al., 2019).

As shown in table 4, the BERT model pre-trained with our multi-grained method outperforms the single-grained BERT-WWM on all 8 SuperGLUE tasks, and attains an average score of 65.53 surpassing the baseline by 1.89. This improvement over BERT-WWM demonstrates that the effectiveness of *LICHEE* is attributed greatly to the information gain of its multi-grained representations, more than just token boundary information. We also notice that, similar to the CLUEWSC task, a huge increase of 8.45 on accuracy is achieved for the CB dataset of 250 training samples, because our pre-training method leverages the information gains of multi-grained tokens and produces more accurate representations, which is especially effective on tasks with small training data.

This result evidently illustrates that *LICHEE* is not only effective on tasks of character based language like Chinese that highly relies on correct tokenizations, but can also produce significant improvements on languages that are naturally tokenized such as English.

5 Conclusion

In this paper, we have proposed a novel multi-grained method for language model pre-training named *LICHEE*, which can be applied to both auto-regressive and auto-encoding PLMs. In our method, the fine-grained embeddings and the coarse-grained embeddings are separately learned and integrated as the multi-grained embeddings, which is then passed into the encoder of the language model. Experiments show that *LICHEE* can significantly enhance the model performance by a great margin on downstream tasks of both Chinese and English, and significantly improve the inference speed compared to the prior multi-grained method.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Xiaoya Li, Yuxian Meng, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. 2019. Is word segmentation necessary for deep learning of chinese representations? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3242–3252.

Lu Li Hai Hu Chenjie Cao Weitang Liu Junyi Li Yudong Li Kai Sun Yechen Xu Yiming Cui Cong Yu Qianqian Dong Yin Tian Dian Yu Bo Shi Jun Zeng Rongzhao Wang Weijian Xie Yanting Li Yina Patterson Zuoyu Tian Yiwen Zhang He Zhou Shaoweihua Liu Qipeng Zhao Cong Yue Xinrui Zhang Zhengliang Yang Zhenzhong Lan Liang Xu, Xuanwei Zhang. 2020. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*.

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*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in neural information processing systems*, pages 3266–3280.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019b. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*.

Xinsong Zhang and Hang Li. 2020. Ambert: A pre-trained language model with multi-grained tokenization. *arXiv preprint arXiv:2008.11869*.