ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations

Shizhe Diao[♡*], **Jiaxin Bai**[♡*], **Yan Song**[♠], **Tong Zhang**[♡], **Yonggang Wang**[♠]
[♡]The Hong Kong University of Science and Technology
{sdiaoaa, jbai, tongzhang}@ust.hk
[♠]Sinovation Ventures
{songyan, wangyonggang}@chuangxin.com

Abstract

The pre-training of text encoders normally processes text as a sequence of tokens corresponding to small text units, such as word pieces in English and characters in Chinese. It omits information carried by larger text granularity, and thus the encoders cannot easily adapt to certain combinations of characters. This leads to a loss of important semantic information, which is especially problematic for Chinese because the language does not have explicit word boundaries. In this paper, we propose ZEN, a BERT-based Chinese (**Z**) text encoder **E**nhanced by **N**-gram representations, where different combinations of characters are considered during training. As a result, potential word or phrase boundaries are explicitly pre-trained and fine-tuned with the character encoder (BERT). Therefore ZEN incorporates the comprehensive information of both the character sequence and words or phrases it contains. Experimental results illustrated the effectiveness of ZEN on a series of Chinese NLP tasks. We show that ZEN, using less resource than other published encoders, can achieve state-of-the-art performance on most tasks. Moreover, it is shown that reasonable performance can be obtained when ZEN is trained on a small corpus, which is important for applying pre-training techniques to scenarios with limited data. The code and pre-trained models of ZEN are available at `https://github.com/sinovation/ZEN`.

1 Introduction

Pre-trained text encoders (Peters et al., 2018b; Devlin et al., 2018; Radford et al., 2018, 2019; Yang et al., 2019) have drawn much attention in natural language processing (NLP), because state-of-the-art performance can be obtained for many NLP tasks using such encoders. In general, these encoders are implemented by training a deep neural model on large unlabeled corpora. Although the use of big data brings success to these pre-trained encoders, it is still unclear whether existing encoders have effectively leveraged all useful information in the corpus. Normally, the pre-training procedures are designed to learn on tokens corresponding to small units of texts (e.g., word pieces for English, characters for Chinese) for efficiency and simplicity. However, some important information carried by larger text units may be lost for certain languages when we use a standard encoder, such as BERT. For example, in Chinese, text semantics are greatly affected by recognizing valid n-grams[1]. This means a pre-trained encoder can potentially be improved by incorporating such boundary information of important n-grams.

Recently, there are studies adapting BERT for Chinese with word information, yet they are limited in maintaining the original BERT structure, augmented with learning from weakly supervised word information or requiring external knowledge. As an example, a representative study in Cui et al. (2019) proposed to use the whole-word masking strategy to mitigate the limitation of word information. They used an existing segmenter to produce possible words in the input sentences, and then train a standard BERT on the segmented texts by masking whole words. Sun et al. (2019a) proposed to perform both entity-level and phrase-level masking to learn knowledge and information from the pre-training corpus. However, their approaches are limited in the following senses. First, both methods rely on the word masking strategy so that the encoder can only be trained with existing word and phrase information. Second, similar to the original BERT, the masking strategy results in the mis-match of pretraining and fine-tuning, i.e., no word/phrase information is retained when

[*]Work done during the internship at Sinovation Ventures.

[1]Herein 'valid' regards to that an n-gram is a proper chunk or phrase that is frequently used in the running text.

the encoders are applied to downstream prediction tasks. Third, incorrect word segmentation or entity recognition results cause errors propagated to the pre-training process and thus may negatively affected the generalization capability of the encoder.

In this paper, we propose ZEN, a Chinese (**Z**) text encoder **E**nhanced by representing **N**-grams, which provides an alternative way to improve character based encoders (e.g., BERT) by using larger text granularity. To train our model, one uses an n-gram lexicon from any possible sources such as pre-defined dictionaries and n-gram lists extracted via unsupervised approaches. Such lexicon is then mapped to training texts, and is used to highlight possible combinations of characters that indicate likely salient contents during the training process. Our model then integrate the representations of these n-gram contexts with the character encoder. Similarly, the fine-tune process on any task-specific dataset further enhances ZEN with such n-gram representations. An important feature of our method is that while the model explicitly takes advantage of n-gram information, the model only outputs character-level encodings that is consistent with BERT. Therefore downstream tasks are not affected. ZEN extends the original BERT model and incorporate learning from large granular text explicitly into the model, which is different (and complementary) from previous methods that relied on weak supervision such as whole-word masking.[2] Our experiments follow the standard procedure, i.e., training ZEN on the Chinese Wikipedia dump and fine-tune it on several Chinese downstream NLP tasks. Experiment results demonstrate its validity and effectiveness where state-of-the-art performance is achieved on many tasks using the n-grams automatically learned from the training data other than external or prior knowledge. In particular, our method outperforms some existing encoders trained on much larger corpora on these tasks.

2 ZEN

The overall architecture of ZEN is shown in Figure 1, where the backbone model (character encoder) is BERT[3] (Devlin et al., 2018), enhanced by n-gram information represented by a multi-layer encoder. Since the basis of BERT is well explained in previous studies (Devlin et al., 2018; Yu and Jiang, 2019), in this paper, we focus on the details of ZEN, by explaining how n-grams are processed and incorporated into the character encoder.

[2]Although the character encoder may still use masking as a learning objective, the encoded n-grams are explicitly leveraged in our model.

[3]The two terms, 'BERT' and 'character encoder' are used interchangeably in this paper.

2.1 N-gram Extraction

Pre-training ZEN requires n-gram extraction in the first place before training starts, where two different steps are performed. The first one is to prepare an n-gram lexicon, $\mathcal{L}$, from which one can use any unsupervised method to extract n-grams for later processing. The second step of n-gram extraction is performed during pre-training, where some n-grams in $\mathcal{L}$ are selected according to each training instance $\mathbf{c} = (c_1, c_2, ..., c_i, ..., c_{k_c})$ with k_c characters. Once these n-grams are extracted, we use an n-gram matching matrix, $\mathcal{M}$, to record the positions of the extracted n-grams in each training instance. $\mathcal{M}$ is thus an $k_c \times k_n$ matrix, where each element is represented by

$$m_{ij} = \begin{cases} 1 & c_i \in n_j \\ 0 & c_i \notin n_j \end{cases},$$

where k_n is the number of extracted n-grams from $\mathbf{c}$, and n_j denotes the j-th extracted n-gram. A sample $\mathcal{M}$ with respect to an input text is shown in the bottom part of Figure 1.

2.2 Encoding N-grams

As shown in the right part of Figure 1 (dashed box marked as 'B'), ZEN requires a multi-layer encoder to represent all n-grams, whose information are thus encoded in different levels matching the correspondent layers in BERT. We adopt Transformer (Vaswani et al., 2017) as the encoder, which is a multi-layer encoder that can model the interactions among all n-grams through their representations in each layer. This modeling power is of high importance for ZEN because for certain context, salient n-grams are more useful than random others, and such salient n-grams are expected to be emphasized in pre-training. This effect can be achieved by multi-head self-attention (MhA) mechanism in Transformer (Clark et al., 2019). In detail, the transformer for n-grams is the same as its original version for sequence modeling, except that it does not encode n-gram positions because all n-grams are treated equally without a sequential order. Formally, denote the j-th

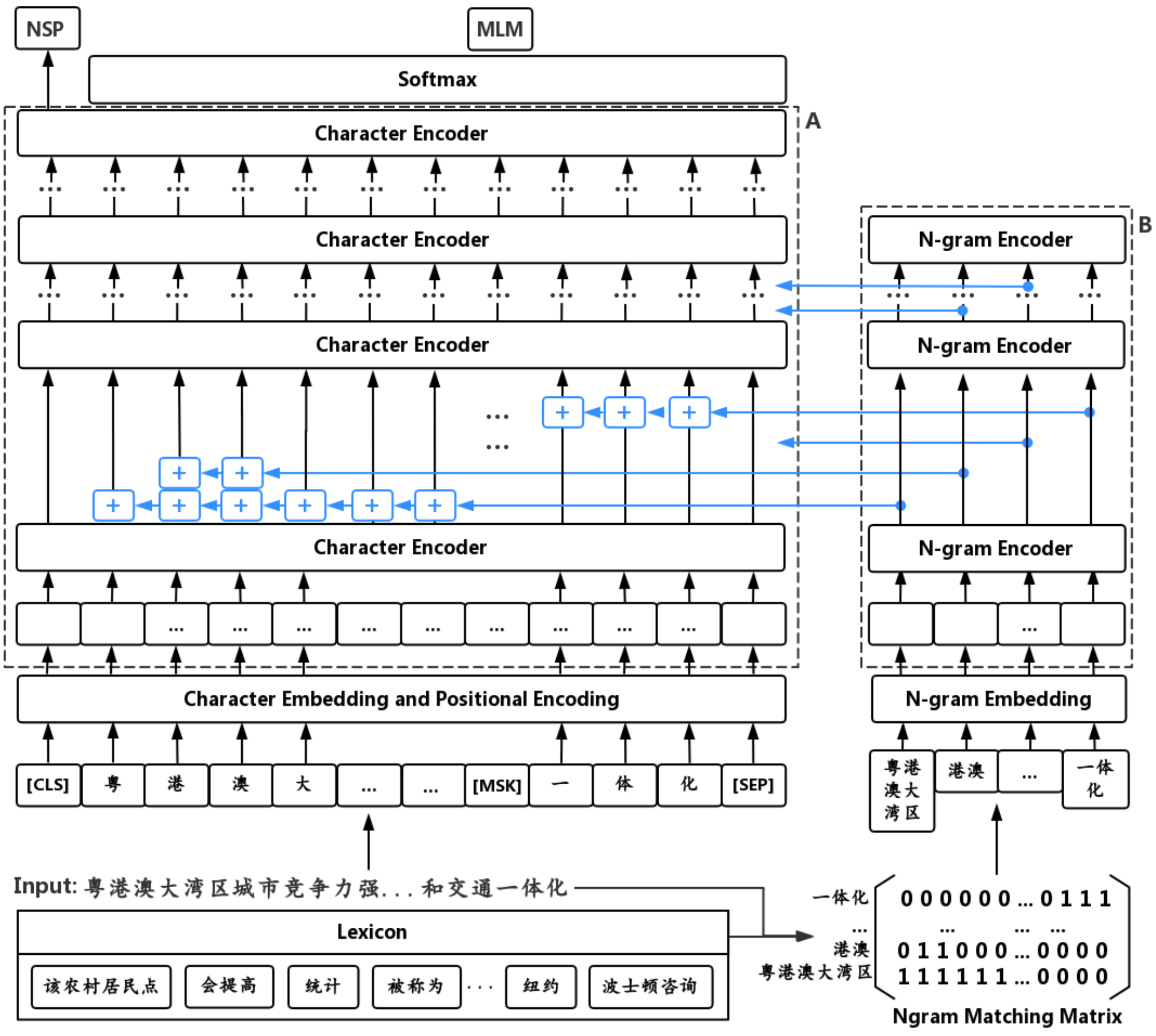

NSP
MLM
Softmax
Character Encoder
A
Character Encoder
N-gram Encoder
B
Character Encoder
N-gram Encoder
Character Encoder
N-gram Encoder
Character Embedding and Positional Encoding
N-gram Embedding
[CLS] 粤 港 澳 大 ... ... [MSK] 一 体 化 [SEP]
粤港澳大湾区 港澳 ... 一体化
Input: 粤港澳大湾区城市竞争力强...和交通一体化
Lexicon
该农村居民点 会提高 统计 被称为 ... 纽约 波士顿咨询
一体化 0 0 0 0 0 0 ... 0 1 1 1
港澳 0 1 1 0 0 0 ... 0 0 0 0
粤港澳大湾区 1 1 1 1 1 1 ... 0 0 0 0
Ngram Matching Matrix

Figure 1: The overall architecture of ZEN, where the area marked by dashed box 'A' presents the character encoder (BERT, in Transformer structure); and the area marked by dashed box 'B' is the n-gram encoder. [NSP] and [MLM] refer to two BERT objectives: next sentence prediction and masked language model, respectively. [MSK] is the masked token. The incorporation of n-grams into the character encoder is illustrated by the addition operation presented in blue color. The bottom part presents n-gram extraction and preparation for the given input instance.

n-gram in layer l by $\mu_j^{(l)}$, the n-gram encoder represents each of them by MhA via

$$\mu_j^{(l+1)} = MhA(Q = \mu_j^{(l)}, K = V = \mathcal{U}^{(l)}) \quad (1)$$

where $\mu_j^{(l)}$ is used as the query (Q) vector to calculate the attentions over all other input n-grams from the same layer, and $\mathcal{U}^{(l)}$ refers to the matrix that stacks all n-gram representations in the layer l that servers as the key (K) and value (V) in MhA. This encoding process is repeated layer-by-layer along with the character encoder.

2.3 Representing N-grams in Pre-training

With the n-gram encoder, ZEN combine the representations of each character and its associated n-grams to train the backbone model, as shown in the left upper part of Figure 1 (dashed box marked as 'A'). In detail, let $\upsilon_i^{(l)}$ and $\mu_{i,k}^{(l)}$ represent embeddings for the i-th character and the k-th n-gram associated to this character at layer l, the enhanced representation for this character is computed by

$$\upsilon_i^{(l)*} = \upsilon_i^{(l)} + \sum_k \mu_{i,k}^{(l)} \quad (2)$$

where $\upsilon_i^{(l)*}$ is the resulting embedding sent to the next layer. Herein $+$ and $\sum$ refer to the element-wise addition operation. Therefore, $\upsilon_i^{(l)*} = \upsilon_i^{(l)}$ when no n-gram covers this character. For the entire layer l, this enhancement can be formulated by

$$\mathcal{V}^{(l)*} = \mathcal{V}^{(l)} + \mathcal{M} \times \mathcal{U}^{(l)} \quad (3)$$

where $\mathcal{V}^{(l)}$ is the embedding matrix for all characters, and its combination with $\mathcal{U}^{(l)}$ can be directly

TASK	CWS		POS		NER		DC		SA		SPM		NLI	
DATASET	MSR		CTB5		MSRA		NEWS		CSC		LCQMC		XNLI	
	S#	C#	S#	C#	S#	C#	D#	C#	D#	C#	SP#	C#	SP#	C#
TRAIN	87K	4M	18K	720K	45K	2M	50K	41M	10K	927K	239K	5M	393K	23M
DEV	-	-	350	10K	-	-	5K	4M	1K	115K	9K	209K	3K	136K
TEST	4K	173K	348	13K	3K	153K	10K	9M	1K	114K	13K	233K	3K	273K

Table 1: The statistics of task datasets used in our experiments. S#, C#, D# and SP# refer to numbers of sentences, characters, documents and sentence pairs, respectively.

done through $\mathcal{M}$. This process is repeated for each layer in the backbone BERT excecept for the last one. The final output of all character embeddings from the last layer is sent to optimize BERT objectives, i.e., mask recovery and next sentence prediction. Note that, since there is masking in BERT training, when a character is masked, n-grams that cover this character are not considered.

3 Experiment Settings

3.1 Tasks and Datasets

For pre-training, following previous studies (Devlin et al., 2018; Cui et al., 2019), we use Chinese Wikipedia dump[4] as the base corpus to learn different encoders including ZEN. To clean the base corpus, we remove useless symbols and translate all traditional characters into simplified ones, and lowercase all English letters. The resulted corpus contains 474M tokens and 23K unique characters. For fine-tuning, we choose seven NLP tasks and their corresponding benchmark datasets in our experiments, many of them have been used in previous studies (Cui et al., 2019; Sun et al., 2019a,b). These tasks and datasets are described as follows.

- **Chinese word segmentation (CWS)**: MSR dataset from SIGHAN2005 Chinese word segmentation Bakeoff (Emerson, 2005).
- **Part-of-speech (POS) tagging**: CTB5 (Xue et al., 2005) dataset with standard splits.
- **Named entity recognition (NER)**: MSRA dataset from international Chinese language processing Bakeoff 2006[5].
- **Document classification (DC)**: THUCNews (News) dataset (Sun et al., 2016) from Sina news with 10 evenly distributed classes.
- **Sentiment analysis (SA)**: The ChnSentiCorp[6] (CSC) dataset with 12,000 documents from three domains, i.e., book, computer and hotel.
- **Sentence pair matching (SPM)**: The LCQMC (a large-scale Chinese question matching corpus) proposed by Liu et al. (2018), where each instance is a pair of two sentences with a label indicating whether their intent is matched.
- **Natural language inference (NLI)**: The Chinese part of the XNLI (Conneau et al., 2018).

The statistics of these datasets with respect to their splits are reported in Table 1. For CWS, POS, we fine-tune and test according to their standard split of training and test sets. For the other tasks, we follow the settings of Cui et al. (2019) to process those datasets in our experiments.

3.2 Implementation

N-grams to build the lexicon $\mathcal{L}$ are extracted from the same training corpus, i.e., Chinese Wikipedia dump, and prepared by sorting them (except for unigrams) according to their frequencies. We try the cut-off threshold between 5 and 40 where all those n-grams with frequency lower than the threshold are not included in $\mathcal{L}$. As a result, the sizes of $\mathcal{L}$ with respect to different threshold range from 179K to 64K n-grams in them.[7] The embeddings of the n-grams are randomly initialized.

For the backbone BERT in ZEN, we use the same structure as that in previous work (Devlin et al., 2018; Sun et al., 2019a; Cui et al., 2019), i.e., 12 layers with 12 self-attention heads, 768 dimensions for hidden states and 512 for max input length, etc. The pre-training tasks also employ the same masking strategy and next sentence prediction as in Devlin et al. (2018), so that ZEN can be compared with BERT on a fair basis. We use the same parameter setting for the n-gram encoder as in BERT, except that we only use 6 layers and set 128 as the max length of n-grams[8]. The resulting ZEN requires only 20% additional inference time (averaged by testing on the seven tasks) over

[4]https://dumps.wikimedia.org/zhwiki/
[5]http://sighan.cs.uchicago.edu/bakeoff2006/
[6]https://github.com/pengming617/bert_classification

[7]Our main experiments are conducted on cut-off=15, resulting in 104K n-grams in the lexicon.
[8]That is, we extract up to 128 n-grams per instance.

	CWS	POS		NER	DC		SA		SPM		NLI	
	TEST	DEV	TEST	TEST	DEV	TEST	DEV	TEST	DEV	TEST	DEV	TEST
BERT (R)	97.20	95.72	95.43	93.12	96.90	96.71	94.00	94.10	87.22	85.13	75.67	75.01
BERT (P)	97.95	96.30	96.10	94.78	97.60	97.50	94.53	94.67	88.50	86.59	77.40	77.52
BERT-WWM	-	-	-	95.10	97.60	97.60	94.50	95.00	89.20	86.80	78.40	78.00
ERNIE 1.0	-	-	-	95.10	97.30	97.30	95.20	95.40	89.70	87.40	79.90	78.40
ERNIE 2.0 (B)	-	-	-	-	-	-	**95.70**	95.50	**90.90**	87.90	81.20	**79.70**
NEZHA (B)	-	-	-	-	-	-	94.74	95.17	89.98	87.41	**81.37**	79.32
NEZHA-WWM (B)	-	-	-	-	-	-	94.75	95.84	89.85	87.10	81.25	79.11
ERNIE 2.0 (L)	-	-	-	-	-	-	*96.10*	*95.80*	*90.90*	*87.90*	*82.60*	*81.00*
NEZHA (L)	-	-	-	-	-	-	*95.92*	*95.83*	*90.18*	*87.20*	*81.53*	*80.44*
NEZHA-wwm (L)	-	-	-	-	-	-	*95.75*	*96.00*	*90.87*	*87.94*	*82.21*	*81.17*
ZEN (R)	97.89	96.12	95.82	93.24	97.20	96.87	94.87	94.42	88.10	85.27	77.11	77.03
ZEN (P)	**98.35**	**97.43**	**96.64**	**95.25**	**97.66**	**97.64**	95.66	**96.08**	90.20	**87.95**	80.48	79.20

Table 2: The overall performance of ZEN and the comparison against existing models on seven NLP tasks, where R denotes that pre-training starts from random initialization and P is that model parameters are initialized from Google's released Chinese BERT base model. B and L refer to each backbone model uses BERT base or large model, respectively. Since ZEN uses BERT base model, encoders using BERT large model and their performance are listed as references in italic fonts. The bold numbers are the best results from all base models in each column.

the original BERT base model. We adopt mixed precision training (Micikevicius et al., 2017) by the Apex library[9] to speed up the training process. Each ZEN model is trained simultaneously on 4 NVIDIA Tesla V100 GPUs with 16GB memory.

Our task-specific fine-tuning uses similar hyper-parameters reported in Cui et al. (2019), with slightly different settings on max input sequence length and batch size for better utilization of computational resources. Specifically, we set max length to 256 for CWS and POS, and 96 for their batch size. For NER, SPM and NLI, we set both the max length and batch size to 128. For the other two tasks, DC and SA, we set the max length and batch size to 512 and 32, respectively.

4 Experimental Results

4.1 Overall Performance

The first experiment is to compare ZEN and BERT with respect to their performance on the aforementioned NLP tasks. In this experiment, ZEN and BERT use two settings, i.e., training from (R): randomly initialized parameters and (P): pre-trained model, which is the Google released Chinese BERT base model. The results are reported in Table 2, with the evaluation metrics for each task denoted in the second row. Overall, in both R and P settings, ZEN outperforms BERT in all seven tasks, which clearly indicates the advantage of introducing n-grams into the encoding of character sequences. This observation is similar to that from Dos Santos and Gatti (2014); Lample et al. (2016); Bojanowski et al. (2017); Liu et al. (2019a). In detail, when compare R and P settings, the performance gap between ZEN (P) and BERT (P) is larger than that in their R setting, which illustrates that learning an encoder with reliable initialization is more important and integrating n-gram information contributes a better enhancement on well-learned encoders. For two types of tasks, it is noticed that token-level tasks, i.e., CWS, POS and NER, demonstrate a bigger improvement of ZEN over BERT than that of sentence-level tasks. where the potential boundary information presented by n-grams are essential to provide a better guidance to label each character. Particularly for CWS and NER, these boundary information are directly related to the outputs. Similarly, sequence-level tasks show a roughly same trend on the improvement of ZEN over BERT, which also shows the capability of combining both character and n-gram information in a text encoder. The reason behind this improvement is that in token-level tasks, where high-frequent n-grams[10] in many cases are valid chunks in a sentence that carry key semantic information.

We also compare ZEN (P) and existing pre-trained encoders on the aforementioned NLP tasks, with their results listed in the middle part of Table 2.[11] Such encoders include BERT-wwm (Cui et al., 2019), ERNIE 1.0 (Sun et al., 2019a), ERNIE 2.0 (B) (Sun et al., 2019b), ERNIE 2.0

[9]https://github.com/NVIDIA/apex

[10]Such as fixed expressions and common phrases, which may have less varied meanings than other ordinary combinations of characters and random character sequences.

[11]We only report the performance on their conducted tasks.

	CWS	POS		NER	DC		SA		SPM		NLI	
	TEST	DEV	TEST	TEST	DEV	TEST	DEV	TEST	DEV	TEST	DEV	TEST
BERT (R)	95.14	93.64	93.23	87.11	96.02	95.77	93.41	92.33	85.62	85.53	72.12	71.44
ZEN (R)	**96.05**	**93.79**	**93.37**	**88.39**	**96.11**	**96.05**	**93.92**	**93.51**	**86.12**	**85.78**	**72.66**	**72.31**

Table 3: The performance of BERT and ZEN on seven NLP tasks when they are trained on a small corpus.

(L) (Sun et al., 2019b), NEZHA (B) and (L) (Wei et al., 2019) where B and L denote the base and large model of BERT, respectively. Note that although there are other pre-trained encoders with exploiting entity knowledge or multi-model signals, they are not compared in this paper because external information are required in their work. In fact, even though without using such external information, ZEN still achieves the state-of-the-art performance on many of the tasks experimented.

In general, the results clearly indicate the effectiveness of ZEN. In detail, for the comparison between ZEN and BERT-wwm, it shows that, when starting from pre-trained BERT, ZEN outperforms BERT-wwm on all tasks that BERT-wwm has results reported. This observation suggests that explicitly representing n-grams and integrating them into BERT has its advantage over using masking strategy, and using n-grams rather than word may have better tolerance on error propagation since word segmentation is unreliable in many cases. The comparison between ZEN and ERNIE encoders also illustrates the superiority of enhancing BERT with n-grams. For example, ZEN shows a consistent improvement over ERNIE 1.0 even though significantly larger non-public datasets were utilized in their pre-training. Compared to ERNIE 2.0, which used many more pre-training tasks and significantly more non-public training data, ZEN is still competitive on SA, SPM and NLI tasks. Particularly, ZEN outperforms ERNIE 2.0 (B) on SA (TEST) and SPM (TEST), which indicates that n-gram enhanced character-based encoders of ZEN can achieve performance comparable to approaches using significantly more resources. Since the two approaches are complementary to each other, one might be able to combine them to achieve higher performance. Moreover, ZEN and ERNIE 2.0 (L) have comparable performance on some certain tasks (e.g., SA and SPM), which further confirms the power of ZEN even though the model of ERNIE 2.0 is significantly larger. Similar observation is also drawn from the comparison between ZEN and NEZHA, where ZEN illustrates its effectiveness again when compared to a model that learning with larger model and more data, as well as more tricks applied in pre-training. However, for NLI task, ZEN's performance is not as good as ERNIE 2.0 and NEZHA (B & L), which further indicates that their model are good at inference task owing to their larger model setting and large-scale corpora have much more prior knowledge.

4.2 Pre-training with Small Corpus

Pre-trained models usually require a large corpus to perform its training. However, in many applications in specialized domains, a large corpus may not be available. For such applications with limited training data, ZEN, with n-gram enhancement, is expected to encode text much more effectively. Therefore, to further illustrate the advantage of ZEN, we conduct an experiment that uses a small corpus to pre-train BERT and ZEN. In detail, we prepare a corpus with $1/10$ size of the entire Chinese Wikipedia by randomly selecting sentences from it. Then all encoders are pre-trained on it with random initialization and tested on the same NLP tasks in the previous experiment. The results are reported in Table 3. In general, same trend is shown in this experiment when compared with that in the previous one, where ZEN constantly outperform BERT in all task. This observation confirms that representing n-grams provides stable enhancement when our model is trained on corpora with different sizes. In detail, these results also reveals that n-gram information helps more on some tasks, e.g., CWS, NER, NLI, over the others. The reason is not surprising since that boundary information carried by n-grams can play a pivotal role in these tasks. Overall, this experiment simulates the situation of pretraining a text encoder with limited data, which could be a decisive barrier in pre-training a text encoder in the cold-start scenario, and thus demonstrates that ZEN has its potential to perform well in this situation.

5 Analyses

We analyze ZEN with several factors affecting its performance. Details are illustrated in this section.

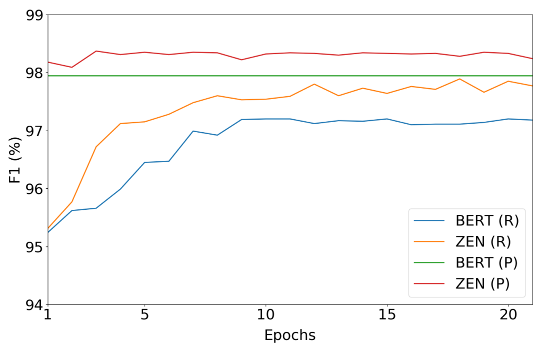


Figure 2: CWS performance against training epochs of BERT and ZEN with different parameter initialization.

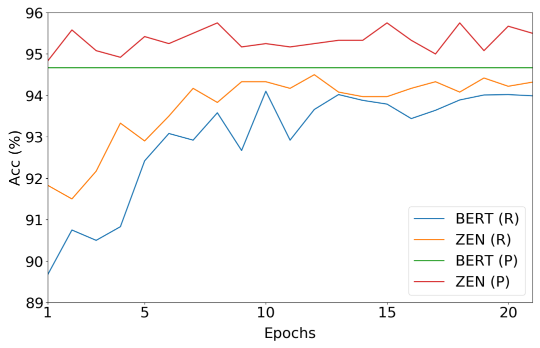


Figure 3: SA performance against training epochs of BERT and ZEN with different parameter initialization.

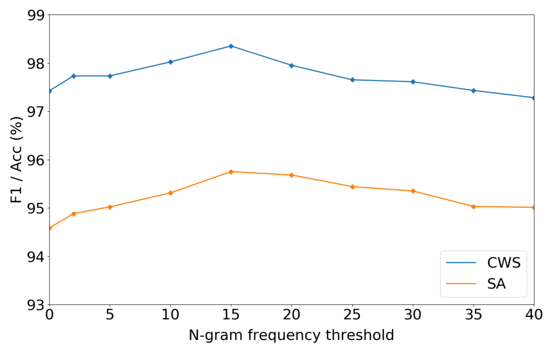


Figure 4: CWS and SA performance of ZEN against frequency threshold of constructing n-gram lexicons.

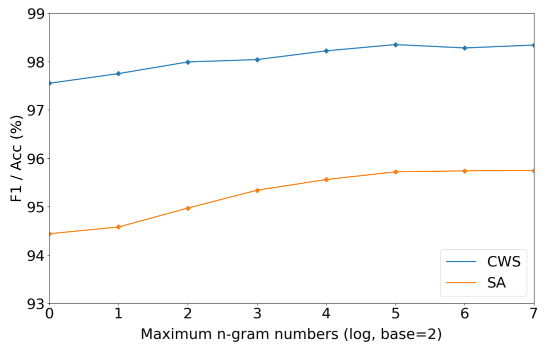


Figure 5: CWS and SA performance of ZEN against maximum n-gram numbers for training each instance.

5.1 Effects of Pre-training Epochs

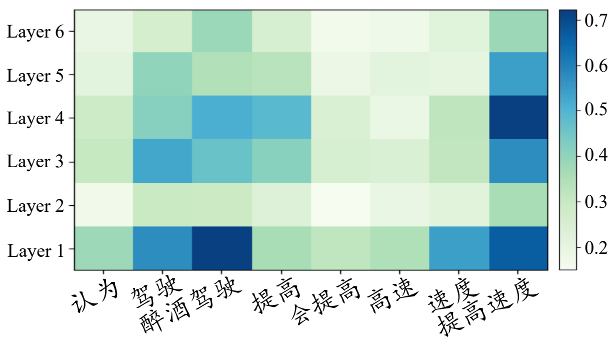
The number of pretraining epochs is another factor affecting the performance of pre-trained encoders. In this analysis, we use CWS and SA as two probing tasks to test the performance of different encoders (BERT and ZEN) against the number of pretraining epochs. The pretrained models at certain epochs are fine-tuned on these tasks, and the results are illustrated in Figure 2 and 3. We have the following observations. First, for both P and R models, ZEN shows better curves than those of BERT in both tasks, which indicates that ZEN achieves higher performance at comparable pretraining stages. Second, for R settings, ZEN shows a noticeable faster convergence than BERT, especially during the first few epochs of pretraining. This demonstrates that n-gram information improves the encoder's performance when pretraining starts from random initialization.

5.2 Effects of N-gram Extraction Threshold

To explore how n-gram extraction cutoff threshold affects the performance of ZEN, we test it with different thresholds for n-gram lexicon extraction. Similar to the previous experiment, we also use CWS and SA as the probe tasks in this analysis.

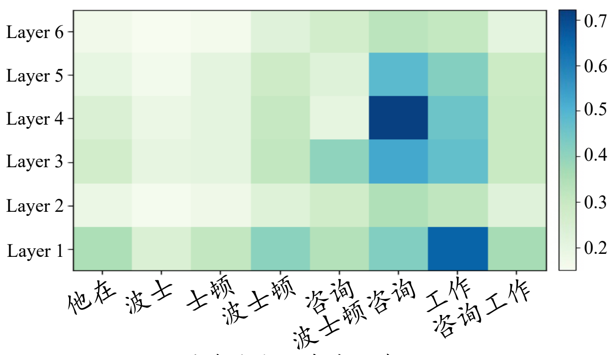
The first analysis on threshold-performance relations is demonstrated in Figure 4, where we set the threshold ranging from 0 to 40 and use the max number of 128 n-grams in pre-training. In doing so, we observe that the best performed ZEN on both tasks is obtained when the threshold is set to 15, where increasing the threshold value under 15 causes improved performance of ZEN and vice versa when it gets over 15. This observation confirms that either too many (lower threshold) or too few (higher threshold) n-grams in the lexicon are less helpful in enhancing ZEN's performance, since there exists a balance between introducing enough knowledge and noise.

For the second analysis, when an optimal threshold is given (i.e., 15), one wants to know the performance of ZEN with different maximum number of n-grams in pre-training for each input sequence. In this analysis we test such number ranging from 0 (no n-grams encoded in ZEN) to 128, with the results shown in Figure 5 (X-axis is in log view with base 2). It shows that the number 32 (2^5) gives a good tradeoff between performance and computation, although there is a small gain by



Input: 你是否认为醉酒驾驶会提高速度？
(*Do you think drunk driving will speed up?*)

Figure 6: The heatmap of n-grams encoded by ZEN across different layers for an example sentence.



Input: 他在波士顿咨询工作。
(*He works in Boston Consulting Group.*)

Figure 7: The heatmap of n-grams encoded by ZEN across different layers for an example sentence.

using more n-grams. This analysis illustrates that ZEN only requires a small numbers of n-grams to achieve good performance.

5.3 Visualization of N-gram Representations

In addition to quantitative analysis, we also conduct case studies on some certain instances to further illustrate the effectiveness of n-gram representations in pre-training ZEN. Figure 6 and 7 visualize the weights of extracted n-grams from two input instances when they are encoded by ZEN across different layers. In general, 'valid' n-grams are more favored than others, e.g., 提高 (*improve*), 波士顿 (*Boston*) have higher weights than 会提高 (*will improve*) and 士顿 (*Ston*), especially those ones that have cross ambiguities in the context, e.g., 高速 (*high speed*) should not be considered in the first instance so that 速度 (*speed*) has a higher weight than it. This observation illustrates that ZEN is able to not only distinguish those phrasal n-grams to others but also select appropriate ones according to the context. Interestingly, for different layers, long (and valid) n-grams, e.g., 提高速度 (*speed up*) and 波士顿咨询 (*Boston consulting group*), tend to receive more intensive weights at higher layers, which implicitly indicates that such n-grams contain more semantic rather than morphological information. We note that information encoded in BERT follows a similar layer-wise order as what is suggested in Jawahar et al. (2019). The observations from this case study, as well as the overall performance in previous experiments, suggest that integration of n-grams in ZEN not only enhances the representation power of character-based encoders, but also provides a potential solution to some text analyzing tasks, e.g., chunking and keyphrase extraction.

6 Related Work

Representation learning of text attracts much attention in recent years, with the rise of deep learning in NLP (Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014). There are considerable interests in representing text with contextualized information (Ling et al., 2015; Melamud et al., 2016; Bojanowski et al., 2017; Song et al., 2017, 2018; Peters et al., 2018a; Song and Shi, 2018). Following this paradigm, pre-trained models have been proposed and are proven useful in many NLP tasks (Devlin et al., 2018; Radford et al., 2018, 2019; Yang et al., 2019). In detail, such models can be categorized into two types: autoregressive and autoencoding encoders. The former models behave like normal language models that predict the probability distributions of text units following observed texts. These models, such as GPT (Radford et al., 2018) and GPT2 (Radford et al., 2019), are trained to encode a unidirectional context. Differently, the autoencoding models, such as BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019), leverage bidirectional context, and encode text by reconstructing the masked tokens in each text instance according to their context from both sides.

Because words carry important linguistic information in Chinese, many enhanced pre-train models are proposed specifically for Chinese that can utilize word-level information in one way or another. For example, ERNIE 1.0 (Sun et al., 2019a) adopted a multi-level masking strategy performed on different level of texts; its improved version, ERNIE 2.0 (Sun et al., 2019b) used continual pre-training strategy which is benefited from multi-task learning with more parameters in the model. Recently, BERT-wwm (Cui et al., 2019) enhanced

Chinese BERT with a simple masking of whole-words. In addition, there are other recent studies that enhanced BERT for Chinese language processing, such as optimizing training via special optimization techniques (Wei et al., 2019) or from prior knowledge (Liu et al., 2019b). All the studies revealed that processing on larger granularity of text is helpful in Chinese, which is consistent with previous findings in many Chinese NLP tasks (Song et al., 2009; Song and Xia, 2012; Wu et al., 2015; Chang et al., 2018; Higashiyama et al., 2019). However, previous approach are limited to the use of weak supervision, i.e., masking, to incorporate word/phrase information. ZEN thus provides an alternative solution that explicitly encodes n-grams into character-based encoding, which is effective for downstream NLP tasks.

7 Conclusion

In this paper, we proposed ZEN, a pre-trained Chinese text encoder enhanced by n-gram representations, where different combinations of characters are extracted, encoded and integrated in training a backbone model, i.e., BERT. In ZEN, given a sequence of Chinese characters, n-grams are extracted and their information are effectively incorporated into the character encoder. Different from previous work, ZEN provides an alternative way of learning larger granular text for pre-trained models, where the structure of BERT is extended by another Transformer-style encoder to represent the extracted n-grams for each input text instance.

Experiments on several NLP tasks demonstrated the validity and effectiveness of ZEN. Particularly, state-of-the-art results were obtained while ZEN only uses BERT base model requiring less training data and no knowledge from external sources compared to other existing Chinese text encoders. Further analyses of ZEN are conducted, showing that ZEN is efficient and able to learn with limited data. We note that ZEN employs a different method to incorporate word information that is complementary to some other previous approaches. Therefore it is potentially beneficial to combine it with previous approaches suggested by other researchers, as well as to other languages.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Chieh-Teng Chang, Chi-Chia Huang, and Jane Yung-jen Hsu. 2018. A Hybrid Word-Character Model for Abstractive Summarization. *arXiv preprint arXiv:1802.09968*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What Does BERT Look At? An Analysis of BERT's Attention. *arXiv preprint arXiv:1906.04341*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating Cross-lingual Sentence Representations. *arXiv preprint arXiv:1809.05053*.

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. 2018. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Cicero Dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Thomas Emerson. 2005. The Second International Chinese Word Segmentation Bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*.

Shohei Higashiyama, Masao Utiyama, Eiichiro Sumita, Masao Ideuchi, Yoshiaki Oida, Yohei Sakamoto, and Isaac Okada. 2019. Incorporating Word Attention into Character-Based Word Segmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2699–2709.

Ganesh Jawahar, Benoît Sagot, Djamé Seddah, Samuel Unicomb, Gerardo Iñiguez, Márton Karsai, Yannick Léo, Márton Karsai, Carlos Sarraute, Éric Fleury, et al. 2019. What Does BERT Learn about the Structure of Language? In *57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. *arXiv preprint arXiv:1603.01360*.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/Too Simple Adaptations of Word2Vec for Syntax Problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado.

Wei Liu, Tongge Xu, Qinghua Xu, Jiayu Song, and Yueran Zu. 2019a. An Encoding Strategy Based Word-Character LSTM for Chinese NER. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2379–2389.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2019b. K-BERT: Enabling Language Representation with Knowledge Graph. *arXiv preprint arXiv:1909.07606*.

Xin Liu, Qingcai Chen, Chong Deng, Huajun Zeng, Jing Chen, Dongfang Li, and Buzhou Tang. 2018. LCQMC: A Large-scale Chinese Question Matching Corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1952–1962.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany.

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*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, New Orleans, Louisiana.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep Contextualized Word Representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8).

Yan Song, Chunyu Kit, and Xiao Chen. 2009. Transliteration of Name Entity via Improved Statistical Translation on Character Sequences. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 57–60, Suntec, Singapore.

Yan Song, Chia-Jung Lee, and Fei Xia. 2017. Learning Word Representations with Regularization from Prior Knowledge. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 143–152, Vancouver, Canada.

Yan Song and Shuming Shi. 2018. Complementary Learning of Word Embeddings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4368–4374.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional Skip-Gram: Explicitly Distinguishing Left and Right Context for Word Embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 175–180, New Orleans, Louisiana.

Yan Song and Fei Xia. 2012. Using a Goodness Measurement for Domain Adaptation: A Case Study on Chinese Word Segmentation. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3853–3860, Istanbul, Turkey.

M Sun, J Li, Z Guo, Z Yu, Y Zheng, X Si, and Z Liu. 2016. THUCTC: An Efficient Chinese Text Classifier. *GitHub Repository, https://github. com/thunlp/THUCTC*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019a. ERNIE: Enhanced Representation through Knowledge Integration. *arXiv preprint arXiv:1904.09223*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019b. ERNIE 2.0: A Continual Pre-training Framework for Language Understanding. *arXiv preprint arXiv:1907.12412*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in neural information processing systems*, pages 5998–6008.

Junqiu Wei, Xiaozhe Ren, Xiaoguang Li, Wenyong Huang, Yi Liao, Yasheng Wang, Jiashu Lin, Xin Jiang, Xiao Chen, and Qun Liu. 2019. NEZHA: Neural Contextualized Representation for Chinese Language Understanding. *arXiv preprint arXiv:1909.00204*.

Yonghui Wu, Min Jiang, Jianbo Lei, and Hua Xu. 2015. Named Entity Recognition in Chinese Clinical Text Using Deep Neural Network. *Studies in health technology and informatics*, 216:624.

Naiwen Xue, Fei Xia, Fu Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11(2):207–238.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237*.

Jianfei Yu and Jing Jiang. 2019. Adapting bert for target-oriented multimodal sentiment classification. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5408–5414.