Z-Code++: A Pre-trained Language Model Optimized for Abstractive Summarization

Pengcheng He[1], **Baolin Peng**[2], **Song Wang**[1],
Yang Liu[1], **Ruochen Xu**[1], **Hany Hassan Awadalla**[1],**Yu Shi**[1], **Chenguang Zhu**[1],
Wayne Xiong[1], **Michael Zeng**[1], **Jianfeng Gao**[2], **Xuedong Huang**[1]
[1] Microsoft Azure AI
[2] Microsoft Research
penhe@microsoft.com

Abstract

This paper presents Z-Code++, a new pre-trained language model optimized for abstractive text summarization. The model extends the state of the art encoder-decoder model using three techniques. First, we use a two-phase pre-training process to improve model's performance on low-resource summarization tasks. The model is first pre-trained using text corpora for language understanding, and then is continually pre-trained on summarization corpora for grounded text generation. Second, we replace self-attention layers in the encoder with disentangled attention layers, where each word is represented using two vectors that encode its content and position, respectively. Third, we use fusion-in-encoder, a simple yet effective method of encoding long sequences in a hierarchical manner. Z-Code++ creates new state of the art on 9 out of 13 text summarization tasks across 5 languages. Our model is parameter-efficient in that it outperforms the 600x larger PaLM$_{540B}$ on XSum, and the finetuned 200x larger GPT3$_{175B}$ on SAMSum. In zero-shot and few-shot settings, our model substantially outperforms the competing models.

1 Introduction

Text summarization aims at producing a concise and fluent summary while preserving salient content and overall meaning of the source documents. It has been applied in a wide range of real-world applications, e.g., summarizing Web search results for interactive information retrieval (Gao et al., 2022) and generating medical summaries from doctor-patient conversation transcripts (Zhang et al., 2021).

While the *extractive* approach is the dominant approach in commercial systems due to its simplicity and effectiveness (Allahyari et al., 2017), the *abstractive* approach is getting more attention in the research community as neural language models are used (e.g., Rush et al., 2015; Nallapati et al., 2016; Chopra et al., 2016; Liu and Lapata, 2019b,a; Pasunuru et al., 2021). Compared to the extractive approach where a summary is constructed using extracted sentences, abstractive summarizers paraphrase the idea of the source documents in a new form, and have a potential of generating more concise and coherent summaries.

However, good abstractive summarizers are harder to develop since we have to deal with problems like semantic representation, inference and low-resource text generation, which are more challenging than sentence extraction. Recently, large-scale pre-trained language models (PLMs) such as PEGASUS (Zhang et al., 2020), GPT (Radford et al., 2019; Brown et al., 2020), T5 (Raffel et al., 2020), have been applied for abstractive summarization. While these models can produce surprisingly fluent text, the generated summaries often contain factual inconsistencies, caused by distorted or fabricated facts about the source documents, which is known as the *hallucination* problem (Kryściński et al., 2019; Celikyilmaz et al., 2020; Ji et al., 2022). In addition, since the amount of text in the source documents can be very large, it is expensive to train an end-to-end abstractive model (e.g., an encoder-decoder transformer model) given the memory constraints of current hardware and the latency constraints of applications such as online document summarization for interactive information retrieval. Therefore, a two-stage approach is widely used, where a subset of document sentences is coarsely selected using an extractive summarizer, and an abstractive summarizer generates the summary conditioning on the extraction (Liu and Lapata, 2019b). This approach is sub-optimal in that salient information might be missed in the extraction.

In this paper, we propose a new encoder-decoder PLM optimized for abstractive summarization, Z-Code++, which significantly extends Z-Code (Wang et al., 2020), a state-of-the-art PLM developed for machine translation, as follows.

First, Z-Code++ is pre-trained on web text using two tasks, replaced token detection (RTD) and corrupted span prediction (CSP). RTD uses a generator to generate ambiguous corruptions and a discriminator to distinguish the ambiguous tokens from the original inputs (Clark et al., 2020). RTD is proved to be more sample-efficient than the classic mask language modeling (MLM) task in learning text representations for language understanding (Bajaj et al., 2022; Hao et al., 2021). In CSP, a consecutive segment of tokens are corrupted and the model is learned to predict the corrupted spans using all the uncorrupted tokens in the original input (Raffel et al., 2020; Joshi et al., 2020). CSP can be viewed as a generalized form of gap sentences generation (GSG), a pre-training task tailored to abstractive summarization (Zhang et al., 2020), where the spans are entire sentences. CSP outperforms GSG in our experiments. In the second phase of grounded pre-training (Peng et al., 2022), the model is continually trained on summarization corpora of documents-summary pairs to better support low-resource fine-tuning to downstream summarization tasks that require the model to produce summaries grounded in source documents. We find in our experiments that grounded pre-training significantly boosts the results on downstream tasks in low-resource settings.

To handle the large input documents, we use fusion-in-encoder (FiE), a simple yet effective method of encoding long sequences in a hierarchical manner. It works by first splitting the input sequence into small chunks, applying attention on each chunk locally to get the chunk representation, and applying attention globally on the concatenated chunk representations to get the representation of the original input.

In addition, we replace the self-attention layer in the encoder with the disentangled attention (DA) layer (He et al., 2020, 2021), where each word is represented using two vectors that encode its content and position, respectively, and the attention weights among words are computed using disentangled matrices on their contents and relative positions, respectively. DA is motivated by the observation that the attention weight of a word pair depends on not only their contents but their relative positions. For example, the dependency between the words "deep" and "learning" is much stronger when they occur next to each other than when they occur in different sentences. We show in our experiments that DA leads to a more effective abstractive summarizer.

For evaluation, we have pre-trained two Z-Code++ models on English data and multi-lingual data, respectively. The English model is trained using 160G English text data and the vocabulary of DeBERTaV2 (He et al., 2020). The multi-lingual model is trained on mC4 corpus which is the same as mT5. These models are evaluated on 13 text summarization tasks across 5 languages, and create new state of the art on 9 tasks. As of May 6th, 2022, Z-Code++ sits atop of the XSum leaderboard, surpassing UL2$_{\text{20B}}$, T5$_{\text{11B}}$ and PEGASUS. It is worth noting that our models are very parameter-efficient. For example, Z-Code++ outperforms PaLM$_{\text{540B}}$, which is 600x larger in model parameters, on XSum, and outperforms a fine-tuned, 200x larger, GPT3$_{\text{175B}}$ on SAMSum. In zero-shot and few-shot settings, our models outperform more substantially the competing models.

2 Z-Code++

This section describes three modeling techniques we have exploited to optimize Z-Code++ for abstractive summarization, including two-phase pre-training, disentangled attention, and long sequence encoding.

2.1 Two-Phase Pre-Training

The two-phase pre-training, which includes the *language model pre-training* and *grounded pre-training* phases, is inspired by the GODEL recipe (Peng et al., 2022) that has been proposed to pre-train language models for grounded text generation tasks, such as dialog response generation and abstractive question-answering.

In the *language model pre-training* phase, Z-Code++ is pre-trained using two language modeling tasks, replaced token detection (RTD) (Clark et al., 2020) and corrupted span prediction (CSP) (Raffel et al., 2020; Joshi et al., 2020). As illustrated in Figure1 (Left), RTD uses a generator trained with MLM to generate ambiguous tokens to replace tokens in the original input $\boldsymbol{X}$, and a discriminator to determine whether a token is from $\boldsymbol{X}$ or generated by the generator. Let θ_G and θ_D be the parameters of the generator and the discriminator, respectively. The MLM loss of the generator is written as

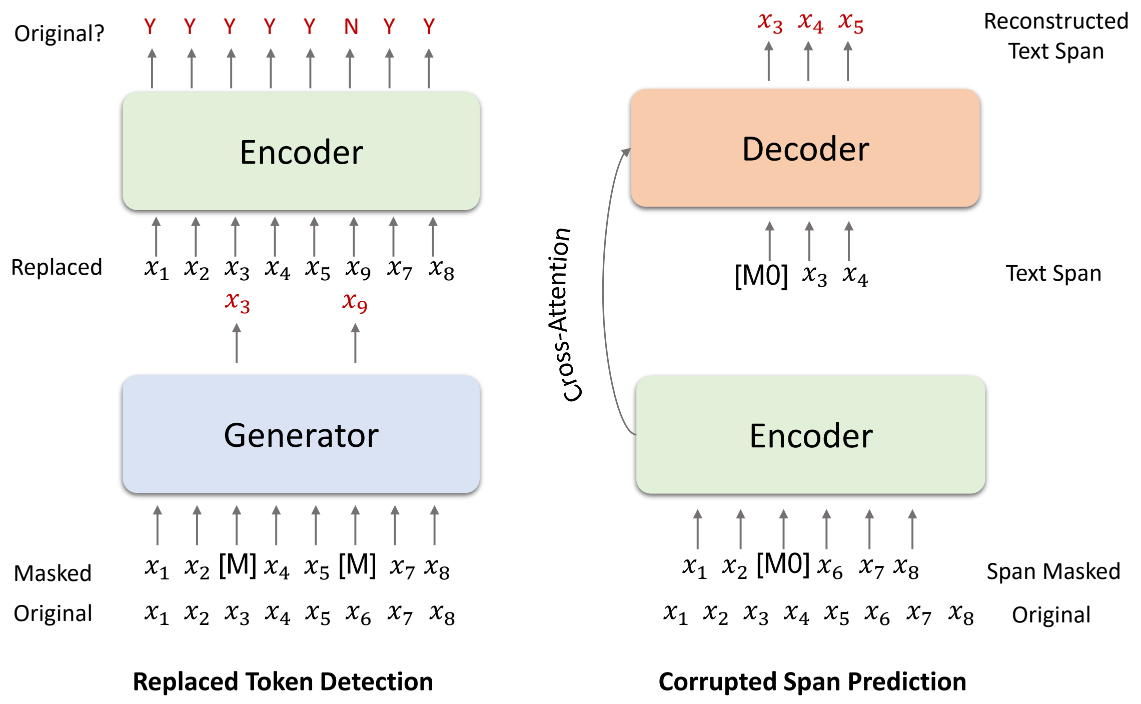
Original?
Y Y Y Y Y N Y Y
Encoder
Replaced
x_1 x_2 x_3 x_4 x_5 x_9 x_7 x_8
x_3 x_9
Generator
Masked
x_1 x_2 [M] x_4 x_5 [M] x_7 x_8
Original
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
Replaced Token Detection
x_3 x_4 x_5
Reconstructed Text Span
Decoder
Cross-Attention
[M0] x_3 x_4
Text Span
Encoder
x_1 x_2 [M0] x_6 x_7 x_8
Span Masked
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
Original
Corrupted Span Prediction

Figure 1: The two pre-training tasks, replaced token detection (RTD) and corrupted span prediction (CSP), used in the language model pre-training phase of Z-Code++. RTD task is to optimize the encoder, and CSP is to optimize the encoder-decoder. Encoders in the same color share parameters during training.

$$L_{\texttt{MLM}} = \mathbb{E}\left(-\sum_{i\in\mathcal{C}} \log p_{\theta_G}\left(\tilde{x}_{i,G} = x_i|\tilde{\boldsymbol{X}}_G\right)\right), \quad (1)$$

where $\tilde{\boldsymbol{X}}_G$ is the input to the generator by randomly masking 15% tokens in original input $\boldsymbol{X}$. The input sequence of the discriminator is constructed by replacing the masked tokens, $x_i, i \in \mathcal{C}$, with the tokens, $\tilde{x}_i$, sampled by the generator as

$$\tilde{x}_{i,D} = \begin{cases} \tilde{x}_i \sim p_{\theta_G}\left(\tilde{x}_{i,G} = x_i|\tilde{\boldsymbol{X}}_G\right), & i \in \mathcal{C} \\ x_i, & i \notin \mathcal{C}. \end{cases} \quad (2)$$

Then the discriminator is trained using the loss

$$L_{\mathbb{RTD}} = \mathbb{E}\left(-\sum_i \log p_{\theta_D}\left(\mathbb{1}\left(\tilde{x}_{i,D} = x_i\right)|\tilde{\boldsymbol{X}}_D, i\right)\right), \quad (3)$$

where $\mathbb{1}(\cdot)$ is the indicator function and $\tilde{\boldsymbol{X}}_D$ is the input to the discriminator constructed via (2). In ELECTRA (Clark et al., 2020), the discriminator and generator share token embeddings and their parameters are optimized via MLM and RTD jointly as $L = L_{\texttt{MLM}} + \lambda L_{\texttt{RTD}}$. However, as pointed out in (He et al., 2021), such embedding sharing makes training highly inefficient since MLM and RTD pull token embeddings into very different directions, creating the "tug-of-war" dynamics. MLM tries to map the tokens that are semantically similar to the embedding vectors that are close to each other. RTD, on the other hand, tries to discriminate semantically similar tokens, pulling their embeddings as far as possible to optimize the classification accuracy. Thus, we use the method of gradient-disentangled embedding sharing (He et al., 2021) by re-parameterizing the token embeddings of the discriminator as

$$\boldsymbol{E}_D = sg(\boldsymbol{E}_G) + \boldsymbol{E}_\Delta, \quad (4)$$

where $\boldsymbol{E}_D$ and $\boldsymbol{E}_G$ are the embedding parameters of the discriminator and generator, respectively, sg is the stop gradient operator which only allows gradients propagation through $\boldsymbol{E}_\Delta$. $\boldsymbol{E}_\Delta$ is initialized as a zero matrix. In each training pass, we first run a forward pass of the generator to generate inputs for the discriminator, and then a backward pass to update $\boldsymbol{E}_G$ with respect to MLM. After that, we run a forward pass for the discriminator using the inputs produced by the generator and run a backward pass with respect to the RTD loss to update $\boldsymbol{E}_D$ by propagating gradients only through $\boldsymbol{E}_\Delta$. After model training, $\boldsymbol{E}_\Delta$ is added to $\boldsymbol{E}_G$ and the sum is saved as $\boldsymbol{E}_D$ in the discriminator, as Equation 4.

The CSP is widely used to optimize the encoder-decoder PLMs such as T5 (Raffel et al., 2020). As illustrated in Figure 1 (Right), given input string $\boldsymbol{X}$, we first select a continuous span $\boldsymbol{Y}_i$ by first randomly selecting a start position in $\boldsymbol{X}$ and a span with an average length of 3. Then we replace the selected span $\boldsymbol{Y}_i$ with a sentinel token $[\mathtt{M}_i]$. We repeat the process until the replaced tokens amount to 15% of all tokens in $\boldsymbol{X}$. Then, we feed the corrupted input $\tilde{\boldsymbol{X}}_{CSR}$ to the encoder. The encoder-decoder model is then trained to recover the $\boldsymbol{Y}_i$ from the context. The CSP loss is written as

$$L_{\mathtt{CSP}} = \mathbb{E}\left(-\sum_{i=1}^{|\boldsymbol{Y}|} \log p_\theta\left(\boldsymbol{Y}_i|\tilde{\boldsymbol{X}}_{\mathtt{CSP}}, \boldsymbol{Y}_{<i}\right)\right) \quad (5)$$

If we restrict the corrupted span $\boldsymbol{Y}_i$ to a complete sentence, CSP is equivalent to the GSG task which simulates the process of extractive summarization and is shown to be effective for training abstractive summarizers (Zhang et al., 2020). In this study, we find the that CSP, as a more general form of GSG, works better across many natural language understanding and generation tasks, including summarization, as to be discussed in Section 3.

Combining the pre-training tasks of MLM, RTD and CSP, in the language model pre-training phase, Z-Code++ is optimized using the joint loss as $L = \lambda_1 L_{\mathtt{MLM}} + \lambda_2 L_{\mathtt{RTD}} + \lambda_3 L_{\mathtt{CSP}}$, where we set $\lambda_1 = 1, \lambda_2 = 30, \lambda_3 = 1$ in our experiment.

In the second phase of *grounded pre-training*, Z-Code++ is continually pre-trained on a collection of summarization datasets, as shown in Table 1, which consist of documents-summary pairs $(\boldsymbol{X}, \boldsymbol{Y})$, to better support low-resource finetuning for downstream summarization tasks that require the model to generate target summaries $\boldsymbol{Y}$ grounded in source documents $\boldsymbol{X}$, as

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{n=1}^{N} p(y_n|y_1, \cdots, y_{n-1}, \boldsymbol{X}) \quad (6)$$

Following T0 (Wei et al., 2021), FLAN (Sanh et al., 2022), and GODEL (Peng et al., 2022), we add for each training pair $(\boldsymbol{X}, \boldsymbol{Y})$ a natural language instruction of the summarization task, as illustrated in the below example and in Table 1. In our experiment, we only apply *grounded pre-training* for low-resource summarizations. Unless specified, we apply the first phase Z-Code++ to downstream task adaptation.

2.2 Disentangled Attention

Disentangled Attention (DA) is first used in DeBERTa (He et al., 2020, 2021). DA is an extension of the classic self-attention (SA) mechanism in that

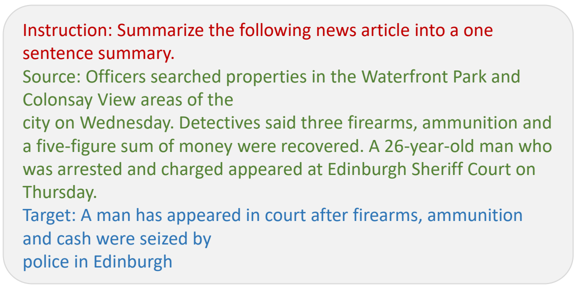
Instruction: Summarize the following news article into a one sentence summary.
Source: Officers searched properties in the Waterfront Park and Colonsay View areas of the
city on Wednesday. Detectives said three firearms, ammunition and a five-figure sum of money were recovered. A 26-year-old man who was arrested and charged appeared at Edinburgh Sheriff Court on Thursday.
Target: A man has appeared in court after firearms, ammunition and cash were seized by
police in Edinburgh

Figure 2: Examples of instructions used for grounded pre-training.

DA represents each input word using two separate vectors: one for the content and the other for the position. Meanwhile, its attention weights among words are computed via disentangled matrices on both their contents and relative positions. The experiments of DeBERTa shows that DA is more efficient than SA to encode the positional dependency in Transformer models. Z-Code++ adopts DA in modeling. Our experiments show that DA leads to a more effective abstractive summarizer.

2.3 Long Sequence Encoding

It is challenging to encode long sequence given the $O(N^2)$ memory and computation complexity of self-attention and DA. Various sparse attention mechanisms have been proposed to alleviate the problem. However, sparse attention often hurts performance on short sequences due to the decrease of attention precision. Inspired by fusion-in-decoder (Izacard and Grave, 2020) and hierarchical transformer (Liu and Lapata, 2019a), we propose fusion-in-encoder (FiE), a simple but effective mechanism to encode long sequences while retaining high attention precision on short sequences. FiE works by separating the L encoder layers of Z-Code++ into m local layers and n global layers. In each local layer, the hidden states of input sequence are split into small chunk of size l (e.g. 256 or 512), and self-attention (or DA) is only applied to those small chunks locally with a complexity of $O(l^2)$. After local layer, the hidden states of those small chunks are concatenated together to form the representation of the long sequence. Global layers are the same as original self-attention (or DA) layers in encoder to fuse the local states of small chunks. With FiE, the complexity of encoder is reduced from $O(LN^2)$ to $O(mNl+nN^2)$. Both the local layers and fusion layers are initialized with the corresponding weights of encoder

Task	Genre	Instructions
MediaSum	Interview	Summarize the following interview script into a two sentences summary.
	-	How can the following interview script be rephrased into a few sentences summary.
MultiNews	News	Summarize the news article into a one sentence summary.
	-	Rephrase the news article with a few sentences.
NewsRoom	News	Summarize the news article into a one sentence summary.
	-	Rephrase the news article concisely with a few sentences.
WikiHow	Wiki	Summarize the paragraph into a one sentence summary.
	-	Summarize the paragraph with a few words.

Table 1: Grounded pre-training summarization datasets and examples of instructions.

layers of Z-Code++. Please check Appendix A.3 for a graphic illustration of FiE. In experiment, we show that compared with LongT5 (Guo et al., 2021) which applies sparse attention that is specifically optimized for summarization, Z-Code++ achieves similar or better performance on long document summarization tasks.

3 Experiment

3.1 Experiment Setups

Datasets We validate the effectiveness of Z-Code++ on 11 representative summarization tasks, which are detailed in Table 2. Among these datasets, XSum (Narayan et al., 2018), CNNDM (See et al., 2017), NewsRoom (Grusky et al., 2018), and MultiNews (Fabbri et al., 2019) are news article summarizations, while SAMSum (Gliwa et al., 2019), MediaSum (Zhu et al., 2021), and Reddit TIFU (Kim et al., 2018) are conversation-like summarization tasks. Following LongT5, we use MultiNews, MediaSum, arXiv (Cohan et al., 2018) and PubMed (Cohan et al., 2018) to assess the long document summarization capability. In addition, WikiLingua (Ladhak et al., 2020) and MLsum (Scialom et al., 2020) are used to evaluate the capacity of Z-Code++ on multilingual summarization.

Implementation Details We have built our models following the same setting as T5. For Z-Code++$_{\texttt{LARGE}}$, there are 24 layers for the encoder and 24 layers for the decoder with 1024 hidden dimension sizes and 16 self-attention heads. Following DeBERTaV3 (He et al., 2021), a 6-layer generator with the same structure as the encoder is employed during the pre-training stage. Z-Code++$_{\texttt{LARGE}}$ is trained on 160G data with a vocabulary of size 128k. Our code is implemented based on open sourced pytorch[1] and DeBERTa[2].

We pre-train Z-Code++$_{\texttt{LARGE}}$ for 1M steps with a batch size of 2048 in Azure Machine Learning cluster[3] with 128 A-100 GPUS for 20 days. AdamW is used as the optimizer in all experiments. For tasks with an input length of more than 10k words, i.e., arXiv and PubMed, Fusion-in-Encoder is used to encode the document as described in 2.3. For the other standard summarization tasks with moderate input length (i.e., less than 4k words) we directly feed the input document to the encoder.

For multilingual summarization, we have built Z-Code++$_{\texttt{LARGE}}$ with the same architecture but different training data and vocabulary. Specifically, Z-Code++$_{\texttt{LARGE}}$ is trained with mC4 data and a vocabulary of size 250k, which are the same as mT5 (Xue et al., 2021). Following XLM (Lample and Conneau, 2019), CCMatrix (Schwenk et al., 2019) and CCAligned (El-Kishky et al., 2019), parallel data is used to enhance the cross-lingual summarization of Z-Code++$_{\texttt{LARGE}}$. Due to the limited computational resource, Z-Code++$_{\texttt{LARGE}}$ is trained with only 500B tokens instead of 1T tokens as that for mT5 training.

We use grid search to choose the grounded training and fine-tuning hyper-parameters based on validation set, the parameter search range are listed in appendix A.1.

3.2 Experiment Results

3.2.1 Results on Standard English Summarization Tasks

We first conduct experiments to compare the performance of Z-Code++$_{\texttt{LARGE}}$ with SOTA and PEGASUS$_{\texttt{LARGE}}$ on 7 representative standard public English summarization datasets with moderate document length, including AESLC, SAMSum, XSUM, WikiHow, NewsRoom, CNN/DailyMail(CNNDM), and Reddit TIFU. Following (Chowdhery et al., 2022; Gehrmann et al., 2022), for each dataset we re-

[1]https://pytorch.org/

[2]https://github.com/microsoft/DeBERTa

[3]https://ml.azure.com

Dataset	# Docs.	# Input Tokens Avg/95%	# Summary Tokens Avg/95%	Genre
Standard Document Summarization				
AESLC	14K	152/440	5/13	Business/Personal
SAMSum	15k	132/331	24/52	Dialog
XSUM	227K	458/1,139	25/35	News
WikiHow	168K	623/1,878	90/226	Wiki
NewsRoom	1.3M	715/1,704	43/152	News
CNNDM	311K	827/1,682	74/127	News
Reddit TIFU	41K	470/1,096	24/51	Forum
Long Document Summarization				
MediaSum	463K	1,554/5,323	14/52	Interview
MultiNews	459K	2,103/6,642	264/407	News
PubMed	133K	3,224/8,210	214/401	Scientific
arXiv	215K	6,913/19,560	293/576	Scientific
Multilingual Summarization				
WikiLingua (ru → en)	37K	661/1,468	49/102	Wiki
WikiLingua (vi → en)	13K	1,140/2,570	48/96	Wiki
WikiLingua (es → en)	79K	676/1,454	50/105	Wiki
WikiLingua (tr → en)	3k	549/1,294	50/100	Wiki
MLSum (de)	221k	907/1,712	50/81	News
MLSum (es)	266k	1,195/2,402	31/50	News

Table 2: Statistics of the datasets used for evaluation including the total number of documents, the average length of input tokens and summary tokens, and the genres of each dataset.

Dataset	Prior SOTA	$\text{PEGASUS}_{\text{LARGE}}$ 470M	$\text{Z-Code++}_{\text{LARGE}}$ 710M
XSum	27.1[a]	24.6	**24.6**
CNNDM	22.6[b]	21.4	**22.2** [4]
NewsRoom	33.5	**33.5**	33.1
WikiHow	18.5	18.5	**22.1**
SAMSum	29.8[c]	26.3	**30.3**
Reddit TIFU	11.3[d]	9.0	**11.6**
AESLC	21.2	21.2	**22.5**
Average	23.4	22.1	**23.8**

Table 3: Results on Common English Summarization tasks. Best numbers are in **Bold**. [a]ST-$\text{MOE}_{268\text{B}}$ (Zoph et al., 2022), [b]$\text{T5}_{11\text{B}}$ (Rothe et al., 2021), [c]$\text{GPT3}_{175\text{B+LoRA}}$ (Hu et al., 2021), [d]MAPPET+$\text{BART}_{\text{LARGE}}$ (Aghajanyan et al., 2021).

port the average F-measure ROUGE-2 score of 5 runs. Detailed F-measure of ROUGE-1/ROUGE-2/ROUGE-L scores can be found in Appendix ??.

As listed in Table 3 , $\text{Z-Code++}_{\text{LARGE}}$ achieves substantial improvements over $\text{PEGASUS}_{\text{LARGE}}$, which is a PLM optimized for abstractive summarization, on 6 out of 7 tasks in terms of ROUGE-2 F-measure score. [5] Specifically, on SAMSum, a critical dialog summarization task, $\text{Z-Code++}_{\text{LARGE}}$ outperforms $\text{GPT-3}_{175\text{B}}$ that is extensively fine-tuned with LoRA(Hu et al., 2021) even though $\text{Z-Code++}_{\text{LARGE}}$ has less than 1/175 parameters of $\text{GPT-3}_{175\text{B}}$. Furthermore, $\text{Z-Code++}_{\text{LARGE}}$ lifts SOTAs by 0.36 points on average. These results demonstrate the effectiveness of Z-Code++ on English document summarization tasks. Additionally, we observe that $\text{Z-Code++}_{\text{LARGE}}$ outperforms $\text{PEGASUS}_{\text{LARGE}}$ on WikiHow, SAMSum, Reddit TIFU, and AESLC by a much larger margin ($> 1\%$) than it does on XSum, CNNDM, and NewsRoom. We speculate that PEGASUS is biased to news-like tasks since it is heavily pre-trained on large amounts of news data. In contrast, Z-Code++ is pre-trained on diverse web data and thus is more adaptable for general-domain summarization tasks.

3.2.2 Results on Long Document Summarization

We compare Z-Code++ to PEGASUS and LongT5, which is optimized for long document summarization. Results in Table 4 show that $\text{Z-Code++}_{\text{LARGE}}$ exceeds all the strong competitors on all long document summarization datasets and lifts SOTA by 0.35 point on average. For FiE, which is used to generate summaries for arXiv and PubMed, we choose the chunk size $l = 256$, and choose the last

[5]The computation cost of the embedding layer is not factored in, so we only display the primary model parameters in the table, excluding those from the embedding layer. This approach is consistent across all subsequent experiments for comparison purposes.

[5]We have achieved 24.1 R2 score on CNNDM using exposure debiasing to address the mismatch between teacher forcing and student forcing learning, as we will describe in detail in a future publication.

Dataset	Prior SOTA	LongT5$_{\text{XLARGE}}$ 3B	LongT5$_{\text{LARGE}}$ 705M	PEGASUS$_{\text{LARGE}}$ 470M	Z-Code++$_{\text{LARGE}}$ 710M
MediaSum	19.7	19.7	19.0	-	**20.2**
MultiNews	21.1[a]	19.4	18.4	18.7	**21.6**
arXiv	21.9[b]	21.9	20.6	17.2	**22.5**
PubMed	24.8	24.8	24.7	19.6	**24.9**
Average	21.9	21.5	20.7	18.5	**22.2**

Table 4: Comparison results on long input summarization tasks. Best numbers are in **Bold**. [a]PRIMER (Xiao et al., 2021), [b]Top-Down Transformer (Pang et al., 2022)

Model	Conciseness	Fluency	No-hallucinations	Informativeness	**Overall**
UL2$_{\text{20B}}$	**0.53**	**0.52**	0.54	0.49	0.50
BART$_{\text{LARGE}}$	0.50	0.50	0.52	0.49	0.49
PEGASUS$_{\text{LARGE}}$	0.52	0.49	0.49	0.49	0.49
T5$_{\text{11B}}$	0.49	0.50	0.49	0.48	0.47
Z-Code++$_{\text{LARGE}}$	0.50	0.51	**0.55**	**0.49**	**0.51**

Table 5: Human evaluation results on the XSum leaderboard.

Dataset	PaLM$_{\text{540B}}$ 540B	mT5$_{\text{XLARGE}}$ 3B	mT5$_{\text{LARGE}}$ 705M	Z-Code++$_{\text{LARGE}}$ 710M
#Training Tokens	500B	1T	1T	500B
Cross-lingual summarization				
WikiLingua (ru → en)	18.6	14.6	11.2	**15.9**
WikiLingua (vi → en)	19.1	14.9	10.9	**16.7**
WikiLingua (es → en)	20.9	17.2	12.6	**17.7**
WikiLingua (tr → en)	23.1	18.3	14.5	**22.9**
Average	20.4	16.3	12.3	**18.3**
Multilingual summarization				
MLSum (de)	33.1	36.2	35.4	**36.8**
MLSum (es)	12.0	13.8	12.3	**14.8**
Average	22.6	25.0	23.9	**25.8**

Table 6: Evaluation results on multi-lingual summarization tasks. Best numbers excluding PaLM$_{\text{540B}}$ are in **Bold**.

layer of encoder as fusion layer based on the experiment results. Specifically, Z-Code++$_{\text{LARGE}}$ outperforms LongT5$_{\text{3B}}$ with less than 1/3 of parameters. These results demonstrate both the effectiveness and flexibility of Z-Code++ by using Disentangled-Attention to encode word dependencies.

3.2.3 Human Evaluation

As human evaluation is the most reliable measurement of the quality of natural language generation models, we submit the test results of XSum to the leaderboard (Khashabi et al., 2021) which requires human raters to compare the generated summaries side by side with human written references. Please check the paper of the leaderboad (Khashabi et al., 2021) to get more details of human evaluation process including instructions, dataset preparing, payments and demographics of the raters. We list the human evaluation results in Table5. Z-Code++ outperforms all the other models, e.g., BART$_{\text{LARGE}}$, PEGASUS$_{\text{LARGE}}$, T5$_{\text{11B}}$, UL2$_{\text{20B}}$(Tay et al., 2022), on the leaderboard in terms of human-overall score. As the human evaluation score is an average of side-by-side preference comparison scores, a score of 0.51 indicates that the annotators prefer the output of Z-Code++ to the human written references. Further more, while hallucination is one of the most critical problems for abstractive summarization, Z-Code++ does not suffer much, i.e., 0.55, among the leaderbard. The human evaluation results validate that Z-Code++ produces higher quality summaries than other models.

3.2.4 Results on Multilingual Summarization

Following GEM-benchmark (Gehrmann et al., 2021), we evaluate the performance of Z-Code++$_{\text{LARGE}}$ [6] on multilingual summarization with WikiLingua and MLSum. We compare Z-Code++$_{\text{LARGE}}$ with mT5$_{\text{LARGE}}$ and mT5$_{\text{XLARGE}}$. The results of PaLM$_{\text{540B}}$, a state of the art PLM, are also listed in Table 6. Compared with mT5$_{\text{XLARGE}}$, Z-Code++$_{\text{LARGE}}$ achieves substantially better performance across all the tasks with only 1/3 parameters and half training data. In addition, we observe a significant performance gap between Z-Code++$_{\text{LARGE}}$ and PaLM$_{\text{540B}}$ on WikiLingua, which is not surprising due to the sharp difference in model size and capacity. However, Z-Code++$_{\text{LARGE}}$ surpasses

[6]Note that Z-Code++$_{\text{LARGE}}$ for multilingual summarization is differently trained. Refer to 3.1 for more details.

PaLM$_{540B}$ on MLSum by a large margin, i.e., 3.7% on MLSum(de), 2.8% on MLSum(es), albeit Z-Code++$_{\text{LARGE}}$ has less than 1/500 parameters. We believe that by scaling up Z-Code++ to a moderate size (e.g., 10B), the performance gap on WikiLingua would be mitigated. We leave it to future work.

3.2.5 Results on Low-Resource Summarization

We explore how well knowledge learned in different pre-training stages can generalize to low-resource summarization scenarios, i.e. zero/few-shot evaluation. For the grounded pre-training phase, we choose to include MediaSum, Multi-News, NewsRoom, and WikiHow datasets. Corresponding instructions are listed in Table 1. We reckon that incorporating diverse datasets and instructions is beneficial, which we leave it to future work. For the fine-tuning stage, following the setting in Zhang et al. (2020), we randomly select the number of training data to 0, 10, 100, and 1000, and sample examples from XSUM, CNNDM, and SAMSum, and then fine-tune Z-Code++ until no significant improvement on the validation set is observed. Note that 0 denotes zero-shot evaluation. Table 11 presents the results. By fine-tuning first-phase pre-trained model, Z-Code++$_{\text{LARGE}}$ outperforms T5$_{\text{LARGE}}$ by more than 3 points on average. PEGASUS$_{\text{LARGE}}$ exceeds Z-Code++$_{\text{LARGE}}$ when the number of training examples is less than 100, which is foreseeable as PEGASUS$_{\text{LARGE}}$ is pre-trained with a pseudo summarization objective. However, Z-Code++$_{\text{LARGE}}$ performs significantly better than them when it is trained with more than 100 examples, showing strong generalization in the few-shot setting. More importantly, with grounded pre-training, Z-Code++$_{\text{LARGE}}$ beats all the competing models by a large margin in both zero and few-shot settings, outperforming PEGASUS$_{\text{LARGE}}$ by 5.7/1.5/3.3 points on average. This suggests that instructions-grounded pre-training enables effective knowledge transfer to downstream low-resource tasks.

4 Conclusions

We present Z-Code++, an efficient and effective pre-trained language model optimized for abstractive text summarization. The model extends the encoder-decoder model using three techniques. The first is a two-phase pre-training process, where the model is first pre-trained using text corpora for language understanding, and then is continually pre-trained on summarization corpora for grounded text generation. The second technique is the use of the disentangled attention mechanism, where each word is represented using two vectors that encode its content and position, respectively. The third is the fusion-in-encoder method for encoding long sequence inputs. We present a comprehensive empirical study to validate the effectiveness of Z-Code++. The model creates new state of the art on 9 out of 13 text summarization tasks across 5 languages. In addition, we show that our model is parameter-efficient in that it outperforms the 600x larger PaLM$_{540B}$ on XSum, and the finetuned 200x larger GPT3$_{175B}$ on SAMSum. Z-Code++ also generalizes well to low-resource downstream tasks. For example, in zero-shot and few-shot settings, our model outperforms more substantially the competing models.

However, evaluation (Liang et al., 2022) and hallucinations are still two long-standing problems of summarizations that we do not touch with in this work, in the future we will 1) explore evaluation metrics that correlate well with human experience, 2) learn to summarize to better align with human preferences (Stiennon et al., 2020; Ouyang et al., 2022), and 3) ground summarization models on

Model	0	10	100	1000	**Average**
			XSUM		
T5$_{\text{LARGE}}$	2.3	2.5	5.5	9.4	4.9
PEGASUS$_{\text{LARGE}}$	3.0	3.5	16.4	18.2	10.3
Z-Code++$^{\dagger}_{\text{LARGE}}$	0.1	2.1	12.3	17.3	8.0
Z-Code++$_{\text{LARGE}}$	**13.7**	**14.0**	**17.5**	**18.9**	**16.0**
			CNNDM		
T5$_{\text{LARGE}}$	4.9	5.1	7.7	11.2	2.7
PEGASUS$_{\text{LARGE}}$	13.3	15.8	18.2	19.4	16.7
Z-Code++$^{\dagger}_{\text{LARGE}}$	0.1	1.5	15.0	18.3	8.7
Z-Code++$_{\text{LARGE}}$	**17.3***	**17.3***	**18.4**	**19.6**	**18.2**
			SAMSum		
T5$_{\text{LARGE}}$	1.3	4.0	10.4	17.8	8.4
PEGASUS$_{\text{LARGE}}$	6.4	11.7	19.8	24.4	15.6
Z-Code++$^{\dagger}_{\text{LARGE}}$	0.1	2.6	20.2	26.3	12.3
Z-Code++$_{\text{LARGE}}$	**7.9**	**17.4**	**22.3**	**28.1**	**18.9**

Table 7: ROUGE-2 score in different summarization datasets. Results are shown on their full test sets using 10, 100, and 1000 training examples. 0 denotes zero-shot results. Results marked with * mean that unfine-tuned checkpoints perform the best, i.e., zero-shot performance is better than the fine-tuned one. Z-Code++$^{\dagger}_{\text{LARGE}}$ refers to fine-tuning from phase 1 pre-trained model. Z-Code++$_{\text{LARGE}}$ fine-tuned from two-phase pre-trained model.

world knowledge to largely reduce hallucinations (LeCun, 2022; Hafner et al., 2023).

Limitations

In this paper, we introduce Z-Code++, a robust pre-trained model tailored for summarization tasks. However, it should be noted that there are certain limitations to our model. Firstly, the model is not versatile enough as it is specifically designed for summarization. It is unclear whether it performs well on other natural language tasks. Secondly, while FiE can handle document summarization, there are still significant potential for improving cost efficiency. Lastly, the evaluation of multilingual summarization is not thorough enough due to the limitations of available datasets. We intend to address these limitations in our future work.

Ethics Statement

The same as all existing generative language models, the generated text of Z-Code++ raises various ethical considerations. One crucial consideration is the issue of potential hallucinations in the summaries generated by the model. The summaries produced by a generative model may not necessarily be faithful to the original article or entirely factual which may mislead the users to make incorrect decisions based on the summary without additional knowledge. In addition, another important consideration is the potential for bias in generated summaries, such as bias based on gender, race, and other factors.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*.

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*.

Payal Bajaj, Chenyan Xiong, Guolin Ke, Xiaodong Liu, Di He, Saurabh Tiwary, Tie-Yan Liu, Paul Bennett, Xia Song, and Jianfeng Gao. 2022. Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals. *arXiv preprint arXiv:2204.06644*.

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*.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. *arXiv preprint arXiv:1804.05685*.

Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzmán, and Philipp Koehn. 2019. Ccaligned: A massive collection of cross-lingual web-document pairs. *arXiv preprint arXiv:1911.06154*.

Alexander R Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*.

Jianfeng Gao, Chenyan Xiong, Paul Bennett, and Nick Craswell. 2022. Neural approaches to conversational information retrieval. *arXiv preprint arXiv:2201.05176*.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D Dhole, et al. 2021. The gem benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:2102.01672*.

Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. 2022. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *arXiv preprint arXiv:2202.06935*.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *arXiv preprint arXiv:1804.11283*.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2021. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

Yaru Hao, Li Dong, Hangbo Bao, Ke Xu, and Furu Wei. 2021. Learning to sample replacements for electra pre-training. *arXiv preprint arXiv:2106.13715*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *arXiv preprint arXiv:2202.03629*.

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.

Daniel Khashabi, Gabriel Stanovsky, Jonathan Bragg, Nicholas Lourie, Jungo Kasai, Yejin Choi, Noah A Smith, and Daniel S Weld. 2021. Genie: A leaderboard for human-in-the-loop evaluation of text generation. *arXiv preprint arXiv:2101.06561*.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2018. Abstractive summarization of reddit posts with multi-level memory networks. *arXiv preprint arXiv:1811.00783*.

Wojciech Kryściński, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551.

Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. 2020. Wikilingua: A new benchmark dataset for cross-lingual abstractive summarization. *arXiv preprint arXiv:2010.03093*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *NeurIPS*.

Yann LeCun. 2022. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Yang Liu and Mirella Lapata. 2019a. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081.

Yang Liu and Mirella Lapata. 2019b. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Bo Pang, Erik Nijkamp, Wojciech Kryściński, Silvio Savarese, Yingbo Zhou, and Caiming Xiong. 2022. Long document summarization with top-down and bottom-up inference. *arXiv preprint arXiv:2203.07586*.

Ramakanth Pasunuru, Asli Celikyilmaz, Michel Galley, Chenyan Xiong, Yizhe Zhang, Mohit Bansal, and Jianfeng Gao. 2021. Data augmentation for abstractive query-focused multi-document summarization.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13666–13674.

Baolin Peng, Michel Galley, Pengcheng He, Chris Brockett, Lars Liden, Elnaz Nouri, Zhou Yu, Bill Dolan, and Jianfeng Gao. 2022. Large-scale pre-training for goal-directed dialogue. Technical report, Microsoft Technical Report.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Sascha Rothe, Joshua Maynez, and Shashi Narayan. 2021. A thorough evaluation of task-specific pre-training for summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 140–145.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, and Armand Joulin. 2019. Ccmatrix: Mining billions of high-quality parallel sentences on the web. *arXiv preprint arXiv:1911.04944*.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. Mlsum: The multilingual summarization corpus. *arXiv preprint arXiv:2004.14900*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Neil Houlsby, and Donald Metzler. 2022. Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*.

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

Yiren Wang, ChengXiang Zhai, and Hany Hassan. 2020. Multi-task learning for multilingual neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1022–1034.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2021. Primer: Pyramid-based masked sentence pre-training for multi-document summarization. *arXiv preprint arXiv:2110.08499*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Longxiang Zhang, Renato Negrinho, Arindam Ghosh, Vasudevan Jagannathan, Hamid Reza Hassanzadeh, Thomas Schaaf, and Matthew R Gormley. 2021. Leveraging pretrained models for automatic summarization of doctor-patient conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3693–3712.

Chenguang Zhu, Yang Liu, Jie Mei, and Michael Zeng. 2021. Mediasum: A large-scale media interview dataset for dialogue summarization. *arXiv preprint arXiv:2103.06410*.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*.

A Appendix

A.1 Hyper parameters

Hyper-parameter	Z-Code++$_{\text{LARGE}}$
Warmup Steps	{50,100,500,1000,1500}
Learning Rates	{5e-6, 8e-6, 9e-6, 1e-5}
Batch Size	{16,32,64}
Weight Decay	0.01
Maximun Training Epochs	{10,20}
Learning Rate Decay	Linear
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.999
Gradient Clipping	1.0
Beam search size	{2,4,5,8}
Length penalty	{0.5-1.2}
Repeated nGram blocking	{0,3}

Table 8: Hyper-parameters for fine-tuning Z-Code++ on summarization tasks.

Hyper-parameter	Z-Code++$_{\text{LARGE}}$
Warmup Steps	{1500}
Learning Rates	{5e-6, 1e-5, 2e-6}
Batch Size	{64}
Weight Decay	0.01
Maximun Training Epochs	{10,20}
Learning Rate Decay	Linear
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.999
Gradient Clipping	1.0
Beam search size	{5,8}
Length penalty	{0.5-1.2}
Repeated nGram blocking	{0,3}

Table 9: Hyper-parameters for Z-Code++ grounded training.

A.2 Rouge scores of summarization tasks

We list the rouge scores of summarizaiton tasks in table10

A.3 Fusion-in-Encoder structure

In figure 3, we show the architecture of FiE.

A.4 Ablation study

We conducted a comprehensive experiment to explore what is important for the encoder's language understanding ability. Specifically, we experiment on the natural language inference task, e.g., MNLI (Williams et al., 2018), the question answering task,

Task	Eval	Metrics		
English Summarization				
XSum	test	47.7	24.6	39.7
CNNDM		44.9	22.2	41.8
NewsRoom		45.5	33.3	41.5
WikiHow		46.4	22.1	45.2
SAMSum		54.6	30.3	46.1
Reddit TIFU		31.0	11.6	25.3
AESLC		38.9	22.5	37.7
MediaSum		36.9	20.2	33.5
MultiNews		47.9	36.8	43.9
arXiv		50.0	22.5	44.9
PubMed		51.1	24.9	46.9
Multi-Lingual Summarization				
WikiLingua(ru → en)	test	38.8	15.9	32.7
WikiLingua(vi → en)		39.3	16.7	33.2
WikiLingua(es → en)		41.5	17.7	34.5
WikiLingua(tr → en)		46.5	22.9	40.2
MLSum(de)	test	47.9	36.8	43.9
MLSum(es)		32.9	14.8	26.5

Table 10: ROUGE-1/ROUGE-2/ROUGE-L results on summarization tasks.

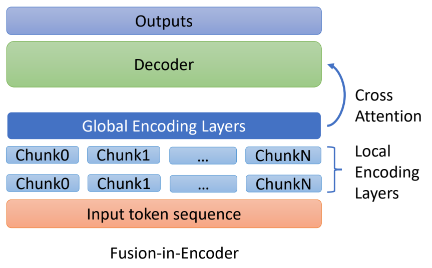

Figure 3: The structure of Fusion-in-Encoder.

e.g., SQuAD (Rajpurkar et al., 2016), the summarization tasks, e.g., XSum (Narayan et al., 2018) and CNNDM (See et al., 2017). The results in Table 12 show that using disentangled attention improves MNLI-matched/mismatched accuracy by 0.9%/1.2%, indicating an improvement in the encoder's language understanding ability. This improvement is also reflected in the performance of two summarization tasks, which see an improvement in R2 scores by 0.39% and 0.22%. Removing RTD significantly decreased performance, indicating that it is essential for improving the model's NLU capability.

A.5 Evaluate on NLU tasks

In order to assess the model's effectiveness on natural language understanding (NLU) tasks, we conducted experiments using the eight NLU tasks from

Model	0	10	100	1000
		XSUM		
$T5_{LARGE}$	12.8/2.3/9.8	13.2/2.5/10.0	21.5/5.5/17.0	31.2/9.4/23.8
$PEGASUS_{LARGE}$	19.3/3.0/12.7	19.4/3.5/14.02	39.07/16.4/31.3	41.6/18.2/33.3
Z-Code++$^{\dagger}_{LARGE}$	3.6/0.1/3.7	16.7/2.1/12.6	35.3/12.3/27.5	40.9/17.3/32.8
Z-Code++$_{LARGE}$	36.6/13.7/28.6	37.4/14.0/29.1	40.6/17.5/30.0	41.9/18.9/33.6
		CNNDM		
$T5_{LARGE}$	18.5/4.9/13.3	19.0/5.1/13.6	24.2/7.7/17.5	31.9/11.2/21.4
$PEGASUS_{LARGE}$	32.9/13.3/29.4	37.6/15.8/33.5	40.3/18.2/37.0	41.7/19.4/38.3
Z-Code++$^{\dagger}_{LARGE}$	3.5/0.1/3.1	11.9/1.5/8.7	37.3/15.0/25.5	40.7/18.3/28.3
Z-Code++$_{LARGE}$	40.0/17.3/25.3*	40.0/17.3/25.3*	41.1/18.4/27.5	42.0/19.6/28.9
		SAMSum		
$T5_{LARGE}$	9.4/1.3/8.2	14.0/4.0/12.0	29.6/10.4/23.5	41.4/17.8/32.8
$PEGASUS_{LARGE}$	26.3/6.4/20.5	37.0/11.7/28.1	45.0/19.8/36.1	49.3/24.4/40.6
Z-Code++$^{\dagger}_{LARGE}$	6.0/0.1/5.4	13.6/2.6/11.0	44.7/20.2/36.7	50.9/26.3/42.3
Z-Code++$_{LARGE}$	26.5/7.9/20.5	40.27/17.4/33.7	47.6/22.3/38.7	52.2/28.1/43.9

Table 11: ROUGE-1/ROUGE-2/ROUGE-L scores in different summarization datasets. Results are shown on their full test sets using 10, 100, and 1000 training examples. 0 denotes zero-shot results. Results marked with * mean that unfine-tuned checkpoints perform the best, i.e., zero-shot performance is better than the fine-tuned one. Z-Code++$^{\dagger}_{LARGE}$ refers to fine-tuning from phase 1 pre-trained model. Z-Code++$_{LARGE}$ fine-tuned from two-phase pre-trained model.

Model	#Traing Tokens	MNLI-m/mm Acc	SQuAD v1.1 F1/EM	XSum R1/R2/RL	CNNDM R1/R2/RL
$T5_{BASE}$	1T	87.1/86.2	92.1/85.4	42.96/20.38/35.10	42.05/20.34/39.40
		Our Implementations			
ZCode++$_{BASE}$	130B	**89.6/89.1**	**92.4/85.6**	**44.04/21.05/36.00**	**43.45/20.71/40.31**
- DA		88.4/88.2	91.5/84.4	43.58/20.66/35.83	43.24/20.49/40.09
- DA - RTD		87.3/86.9	90.5/83.5	43.31/20.28/35.32	43.10/20.35/39.93

Table 12: Ablation study of the impact of encoder performance on generation tasks.

the GLUE dataset (Wang et al., 2019). These tasks are commonly used to evaluate sentence classification performance in machine learning. Our model, Z-Code++, was tested using two approaches: adapting only the encoder and fine-tuning with a classification head, similar to BERT, or adapting the encoder-decoder and treating the task as a generation task, similar to T5. We compared Z-Code++ to other encoder-based PLMs with similar structures, including BERT, RoBERTa, ELECTRA, DeBERTa, and DeBERTaV3, as well as T5 for the encoder-decoder comparison.

The results, shown in Table 13, demonstrate that Z-Code++ performs comparably or better than the other models on all tasks. In particular, Z-Code++ outperformed the other encoder PLMs by an average of more than 1% and outperformed T5 on all tasks with an average improvement of 1.98% in test scores. These results demonstrate Z-Code++ as a strong universal language model with excellent performance on generation tasks and superior performance on NLU tasks.

A.6 Evaluate on NLG tasks

We evaluated the language generation performance of Z-Code++ on a range of English tasks, including abstractive document summarization tasks (XSum, CNNDM, Wikilingual-en), a conversa-

Model	Eval	CoLA	QQP	MNLI-m/mm	SST-2	STS-B	QNLI	RTE	MRPC	Avg.
		Mcc	Acc	Acc	Acc	Corr	Acc	Acc	Acc	
#Train		8.5k	364k	393k	67k	7k	108k	2.5k	3.7k	
Encoder-Only										
BERT$_{\texttt{LARGE}}$	Dev	60.6	91.3	86.6/-	93.2	90.0	92.3	70.4	88.0	84.05
RoBERTa$_{\texttt{LARGE}}$		68.0	92.2	90.2/90.2	96.4	92.4	93.9	86.6	90.9	88.82
ELECTRA$_{\texttt{LARGE}}$		69.1	92.4	90.9/-	96.9	92.6	95.0	88.0	90.8	89.46
DeBERTa$_{\texttt{LARGE}}$		70.5	92.3	91.1/91.1	96.8	92.8	95.3	88.3	91.9	90.00
DeBERTaV3$_{\texttt{LARGE}}$		75.3	**93.0**	**91.8/91.9**	**96.9**	93.0	**96.0**	**92.7**	92.2	**91.37**
Z-Code++		**75.5**	92.8	91.7/91.5	96.3	**93.1**	95.8	92.4	**92.4**	91.23
Encoder-Decoder										
T5$_{\texttt{LARGE}}$	Test	61.2	89.9	89.9/89.6	96.3	89.9	94.8	87.2	**89.9**	87.35
Z-Code++	Test	**69.2**	**90.0**	**91.0/90.9**	**97.9**	**91.2**	**95.1**	**90.7**	89.6	**89.33**
Z-Code++	Dev	86.2	92.4	91.4/91.4	96.5	92.5	95.2	92.1	91.2	92.19

Table 13: Comparison results on the GLUE development set. To make a fair comparison, following previous work on encoder models, we evaluate Z-Code++ with development set. For Encoder-Decoder model we follow T5 to fine-tune all tasks jointly and submit result on test set to GLUE evaluation server.

tional summarization task (SAMSum), data-to-text tasks (WebNLG-en, E2ENLG) and a question answering task (SQuAD v1.1). We compared the performance of the Z-Code++ model to other state-of-the-art models with similar architectures and parameters, as shown in Table 14.

Results show that Z-Code++ outperforms all of the other models' scores by a large margin in terms of ROUGE and BLEU scores. For example, Z-Code++ significantly outperformed T5$_{\texttt{XLARGE}}$ on CNNDM by 1% in terms of ROUGE-2 score, on the WebNLG-en task by 6.9%, and about 1% BLEU score on dialog response generation tasks. Even though it has less than 1/3 the parameters of T5$_{\texttt{XLARGE}}$, Z-Code++ outperformed PEGASUS on SAMSum task by 4% in terms of ROUGE-2 score. We conjecture that PEGASUS is a model specifically optimized for summarization using 1500GB of news data, which may have introduced a domain mismatch with the conversational summarization task. We also compared Z-Code++ to other state-of-the-art models with extremely large parameters, including PaLM, GPT3, and UL2. Z-Code++ outperformed PaLM on three out of four tasks by a large margin, even though it has less than 1/600 the parameters of PaLM. Z-Code++ also outperformed UL2$_{\texttt{20B}}$ on four out of five tasks, even though it has less than 1/20 the parameters of UL2$_{\texttt{20B}}$. These results demonstrate the efficiency of the Z-Code++ model.

Dataset	Metric	BART$_{\text{LARGE}}$ 400M	PEGASUS$_{\text{LARGE}}$ 500M	T5$_{\text{LARGE}}$ 800M	T5$_{\text{XLARGE}}$ 3B	PaLM 540B	GPT3 175B	UL2 20B	Z-Code++ 800M
XSum	R1/R2/RL	45.1/22.3/37.3	47.2/24.6/39.4	44.3/22.0/36.7	-	-/21.2/-	-	**-/26.6/-**	**47.7/24.7/39.7**
CNNDM	R1/R2/RL	44.2/21.3/40.9	44.2/21.5/41.1	43.6/21.4/40.6	42.7/21.0/39.9	-	-	-/21.9/-	**44.9/22.0/41.8**
SAMSum	R1/R2/RL	53.4/28.7/44.2	50.2/26.3/46.2	51.0/27.0/**46.6**	-	-	53.8/29.8/45.9	-/29.6/-	**54.6/30.3**/46.1
WebNLG-en	R1/R2/RL	-	-	67.1/39.6/51.8	75.4/49.4/59.5	-/49.3/-	-	-/55.4/-	**79.0/56.3/64.6**
E2E NLG	R1/R2/RL	-	-	70.8/41.7/49.5	70.8/41.7/49.7	-/45.3/-	-	-/46.5/-	**74.8/46.9/54.0**

Table 14: Comparison results on English NLG tasks.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☑ A2. Did you discuss any potential risks of your work?
Limitations

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Instruction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Section 3, experiments

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 3.1, experiment setup

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 3, experiments and appendix A 1

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 3, experiments

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3, experiments

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Section 3, experiments

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. We quote our results from public benchmark https://leaderboard.allenai.org/genie-mt/submissions/public which run human evaluation from their backend.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. We quote our results from public benchmark https://leaderboard.allenai.org/genie-mt/submissions/public which run human evaluation from their backend.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. We quote our results from public benchmark https://leaderboard.allenai.org/genie-mt/submissions/public which run human evaluation from their backend.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. We quote our results from public benchmark https://leaderboard.allenai.org/genie-mt/submissions/public which run human evaluation from their backend.