3 Pre-trained Language Models

Given the large impact that pre-trained LMs have had on NLP in the pre-train and fine-tune paradigm, there are already a number of high-quality surveys that interested readers where interested readers can learn more (Raffel et al., 2020; Qiu et al., 2020; Xu et al., 2021; Doddapaneni et al., 2021). Nonetheless, in this chapter we present a systematic view of various pre-trained LMs which (i) organizes them along various axes in a more systematic way, (ii) particularly focuses on aspects salient to prompting methods. Below, we will detail them through the lens of *main training objective*, *type of text noising*, *auxiliary training objective*, *attention mask*, *typical architecture*, and *preferred application scenarios*. We describe each of these objectives below, and also summarize a number of pre-trained LMs along each of these axes in Tab. 13 in the appendix.

3.1 Training Objectives

The main training objective of a pre-trained LM almost invariably consists of some sort of objective predicting the probability of text $\boldsymbol{x}$.

Standard Language Model (SLM) objectives do precisely this, training the model to optimize the probability $P(\boldsymbol{x})$ of text from a training corpus (Radford et al., 2019). In these cases, the text is generally predicted in an *autoregressive* fashion, predicting the tokens in the sequence one at a time. This is usually done from left to right (as detailed below), but can be done in other orders as well.

A popular alternative to standard LM objectives are *denoising* objectives, which apply some noising function $\tilde{\boldsymbol{x}} = f_{\text{noise}}(\boldsymbol{x})$ to the input sentence (details in the following subsection), then try to predict the original input sentence given this noised text $P(\boldsymbol{x}|\tilde{\boldsymbol{x}})$. There are two common flavors of these objectives:

Corrupted Text Reconstruction (CTR) These objectives restore the processed text to its uncorrupted state by calculating loss over *only* the noised parts of the input sentence.

Full Text Reconstruction (FTR) These objectives reconstruct the text by calculating the loss over the *entirety* of the input texts whether it has been noised or not (Lewis et al., 2020a).

The main training objective of the pre-trained LMs plays an important role in determining its applicability to particular prompting tasks. For example, left-to-right autoregressive LMs may be particularly suitable for prefix prompts, whereas reconstruction objectives may be more suitable for cloze prompts. In addition, models trained with standard LM and FTR objectives may be more suitable for tasks regarding text generation, whereas other tasks such as classification can be formulated using models trained with any of these objectives.

In addition to the main training objectives above, a number of *auxiliary objectives* have been engineered to further improve models' ability to perform certain varieties of downstream tasks. We list some commonly-used auxiliary objectives in Appendix A.2.

3.2 Noising Functions

In training objectives based on reconstruction, the specific type of corruption applied to obtain the noised text $\tilde{\boldsymbol{x}}$ has an effect on the efficacy of the learning algorithm. In addition, prior knowledge can be incorporated by controlling the type of noise, e.g. the noise could focus on entities of a sentence, which allows us to learn a pre-trained model with particularly high predictive performance for entities. In the following, we introduce several types of noising functions, and give detailed examples in Tab. 4.

Operation	**Element**	**Original Text**	**Corrupted Text**
Mask	one token	Jane will move to New York .	Jane will [Z] to New York .
	two tokens	Jane will move to New York .	Jane will [Z] [Z] New York .
	one entity	Jane will move to New York .	Jane will move to [Z] .
Replace	one token	Jane will move to New York .	Jane will move [X] New York .
	two tokens	Jane will move to New York .	Jane will move [X] [Y] York .
	one entity	Jane will move to New York .	Jane will move to [X] .
Delete	one token	Jane ~~will~~ move to New York .	Jane move to New York .
	two token	Jane ~~will move~~ to New York .	Jane to New York .
Permute	token	Jane will move to New York .	New York . Jane will move to
Rotate	none	Jane will move to New York .	to New York . Jane will move
Concatenation	two languages	Jane will move to New York .	Jane will move to New York . [/s] 简将搬到纽约。

Table 4: Detailed examples for different noising operations.

Masking (e.g. Devlin et al. (2019)) The text will be masked in different levels, replacing a token or multi-token span with a special token such as `[MASK]`. Notably, masking can either be random from some distribution or specifically designed to introduce prior knowledge, such as the above-mentioned example of masking entities to encourage the model to be good at predicting entities.

Replacement (e.g. Raffel et al. (2020)) Replacement is similar to masking, except that the token or multi-token span is not replaced with a `[MASK]` but rather another token or piece of information (e.g., an image region (Su et al., 2020)).

Deletion (e.g. Lewis et al. (2020a)) Tokens or multi-token spans will be deleted from a text without the addition of `[MASK]` or any other token. This operation is usually used together with the FTR loss.

Permutation (e.g. Liu et al. (2020a)) The text is first divided into different spans (tokens, sub-sentential spans, or sentences), and then these spans are be permuted into a new text.

3.3 Directionality of Representations

A final important factor that should be considered in understanding pre-trained LMs and the difference between them is the directionality of the calculation of representations. In general, there are two widely used ways to calculate such representations:

Left-to-Right The representation of each word is calculated based on the word itself and all previous words in the sentence. For example, if we have a sentence "This is a good movie", the representation of the word "good" would be calculated based on previous words. This variety of factorization is particularly widely used when calculating standard LM objectives or when calculating the output side of an FTR objective, as we discuss in more detail below.

Bidirectional The representation of each word is calculated based on all words in the sentence, including words to the left of the current word. In the example above, "good" would be influenced by all words in the sentence, even the following "movie".

In addition to the two most common directionalities above, it is also possible to mix the two strategies together in a single model (Dong et al., 2019; Bao et al., 2020), or perform conditioning of the representations in a randomly permuted order (Yang et al., 2019), although these strategies are less widely used. Notably, when implementing these strategies within a neural model, this conditioning is generally implemented through *attention masking*, which masks out the values in an attentional model (Bahdanau et al., 2014), such as the popular Transformer architecture (Vaswani et al., 2017). Some examples of such attention masks are shown in Figure 2.

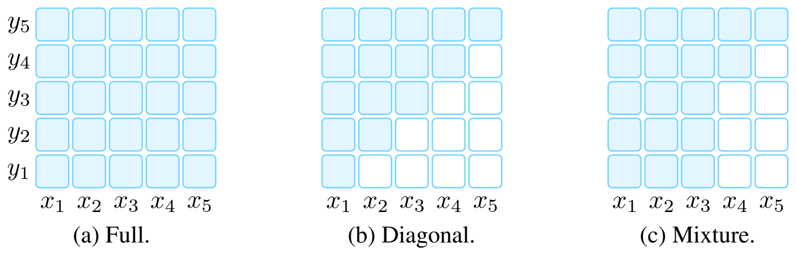

y_5
y_4
y_3
y_2
y_1
x_1 x_2 x_3 x_4 x_5
(a) Full.
x_1 x_2 x_3 x_4 x_5
(b) Diagonal.
x_1 x_2 x_3 x_4 x_5
(c) Mixture.

Figure 2: Three popular attention mask patterns, where the subscript t indicates the t-th timestep. A shaded box at (i, j) indicates that the attention mechanism is allowed to attend to the input element i at output time step j. A white box indicates that the attention mechanism is not allowed to attend to the corresponding i and j combination.

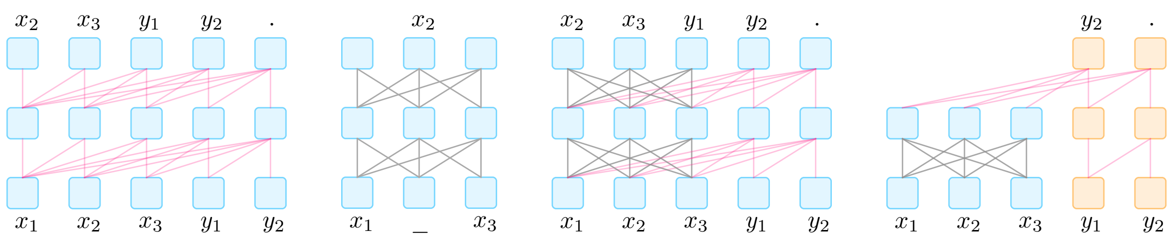
3.4 Typical Pre-training Methods

With the above concepts in mind, we introduce four popular pre-training methods, resulting from diverse combinations of objective, noising function, and directionality. These are described below, and summarized in Fig. 3 and Tab. 5.

3.4.1 Left-to-Right Language Model

Left-to-right LMs (L2R LMs), a variety of *auto-regressive LM*, predict the upcoming words or assign a probability $P(\boldsymbol{x})$ to a sequence of words $\boldsymbol{x} = x_1, \cdots, x_n$ (Jurafsky and Martin, 2021). The probability is commonly broken down using the chain rule in a left-to-right fashion: $P(\boldsymbol{x}) = P(x_1) \times \cdots P(x_n|x_1 \cdots x_{n-1})$.[3]

[3]Similarly, a right-to-left LM can predict preceding words based on the future context, such as $P(x_i|x_{i+1}, \cdots, x_n)$.


x_2 x_3 y_1 y_2 .
x_1 x_2 x_3 y_1 y_2
x_2
x_1 _ x_3
x_2 x_3 y_1 y_2 .
x_1 x_2 x_3 y_1 y_2
y_2 .
x_1 x_2 x_3 y_1 y_2

(a) Left-to-right LM. (b) Masked LM. (c) Prefix LM. (d) Encoder-Decoder.

Figure 3: Typical paradigms of pre-trained LMs.

Example & Applicable Scenario

Left-to-right LMs have been standard since their proposal by Markov in 1913 (Markov, 2006), and have been used continuously since then in both count-based (Goodman, 2001) and neural forms (Bengio et al., 2003; Mikolov et al., 2010; Radford and Narasimhan, 2018). Representative examples of modern pre-trained left-to-right LMs include GPT-3 (Brown et al., 2020), and GPT-Neo (Black et al., 2021).
L2R pre-trained LMs are also the popular backbone that many prompting methods adopt (Radford et al., 2019; Brown et al., 2020) . One practical reason for this is that many such models are large (PanGu-α (Zeng et al., 2021), Ernie-3 (Sun et al., 2021)) and ponderous to train, or not even available publicly. Thus using these models in the pre-train and fine-tune regimen is often not possible.

LMs	$\boldsymbol{x}$			$\boldsymbol{y}$			**Application**
	Mask	**Noise**	**Main Obj.**	**Mask**	**Noise**	**Main Obj.**	
L2R	Diagonal	None	SLM	-	-	-	NLU & NLG
Mask	Full	Mask	CTR	-	-	-	NLU
Prefix	Full	Any	CTR	Diagonal	None	SLM	NLU & NLG
En-De	Full	Any	None†	Diagonal	None	FTR/CRT	NLU & NLG

Table 5: Typical architectures for pre-trained LMs. $\boldsymbol{x}$ and $\boldsymbol{y}$ represent text to be encoded and decoded, respectively. **SLM**: Standard language model. **CTR**: Corrupted text reconstruction. **FTR**: Full text reconstruction. †: Encoder-decoder architectures usually apply objective functions to the decoder only.

3.4.2 Masked Language Models

While autoregressive language models provide a powerful tool for modeling the probability of text, they also have disadvantages such as requiring representations be calculated from left-to-right. When the focus is shifted to generating the optimal representations for down-stream tasks such as classification, many other options become possible, and often preferable. One popular bidirectional objective function used widely in representation learning is the *masked language model* (MLM; Devlin et al. (2019)), which aims to predict masked text pieces based on surrounded context. For example, $P(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ represents the probability of the word x_i given the surrounding context.

Example & Applicable Scenario

Representative pre-trained models using MLMs include: BERT (Devlin et al., 2019), ERNIE (Zhang et al., 2019; Sun et al., 2019b) and many variants. In prompting methods, MLMs are generally most suitable for natural language understanding or analysis tasks (e.g., text classification, natural language inference , and extractive question answering). These tasks are often relatively easy to be reformulated into cloze problems, which are consistent with the training objectives of the MLM. Additionally, MLMs have been a pre-trained model of choice when exploring methods that combine prompting with fine-tuning, elaborated further in §7.

3.4.3 Prefix and Encoder-Decoder

For conditional text generation tasks such as translation and summarization where an input text $\boldsymbol{x} = x_1, \cdots, x_n$ is given and the goal is to generate target text $\boldsymbol{y}$, we need a pre-trained model that is both capable of encoding the input text and generating the output text. There are two popular architectures for this purpose that share a common thread of (1) using an encoder with fully-connected mask to encode the source $\boldsymbol{x}$ first and then (2) decode the target $\boldsymbol{y}$ auto-regressively (from the left to right).

Prefix Language Model The prefix LM is a left-to-right LM that decodes $\boldsymbol{y}$ conditioned on a prefixed sequence $\boldsymbol{x}$, which is encoded by the *same* model parameters but with a fully-connected mask. Notably, to encourage the prefix LM to learn better representations of the input, a corrupted text reconstruction objective is usually applied over $\boldsymbol{x}$, in addition to a standard conditional language modeling objective over $\boldsymbol{y}$.

Encoder-decoder The encoder-decoder model is a model that uses a left-to-right LM to decode $\boldsymbol{y}$ conditioned on a *separate* encoder for text $\boldsymbol{x}$ with a fully-connected mask; the parameters of the encoder and decoder are not shared. Similarly to the prefix LM, diverse types of noising can be applied to the input $\boldsymbol{x}$.

Example & Applicable Scenario

Prefix LMs have been explored in UniLM 1-2 (Dong et al., 2019; Bao et al., 2020) and ERNIE-M (Ouyang et al., 2020) while encoder-decoder models are widely used in pre-trained models such as T5 (Raffel et al., 2020), BART (Lewis et al., 2020a), MASS (Song et al., 2019) and their variants.
Pre-trained models with prefix LMs and encoder-decoder paradigms can be naturally used to text generation tasks with (Dou et al., 2021) or without (Yuan et al., 2021a; Liu and Liu, 2021) prompting using input texts. However, recent studies reveal that other non-generation tasks, such as information extraction (Cui et al., 2021), question answering (Khashabi et al., 2020) , and text generation evaluation (Yuan et al., 2021b) can be reformulated a generation problems by providing appropriate prompts. Therefore, prompting methods (i) broaden the applicability of these generation-oriented pre-trained models. For example, pre-trained models like BART are less used in NER while prompting methods make BART applicable, and (ii) breaks the difficulty of unified modelling among different tasks (Khashabi et al., 2020).

4 Prompt Engineering

Prompt engineering is the process of creating a prompting function $f_{\text{prompt}}(\boldsymbol{x})$ that results in the most effective performance on the downstream task. In many previous works, this has involved *prompt template engineering*, where a human engineer or algorithm searches for the best template for each task the model is expected to perform. As shown in the "Prompt Engineering" section of Fig.1, one must first consider the *prompt shape*, and then decide whether to take a *manual* or *automated* approach to create prompts of the desired shape, as detailed below.

4.1 Prompt Shape

As noted above, there are two main varieties of prompts: *cloze prompts* (Petroni et al., 2019; Cui et al., 2021), which fill in the blanks of a textual string, and *prefix prompts* (Li and Liang, 2021; Lester et al., 2021), which continue a string prefix. Which one is chosen will depend both on the task and the model that is being used to solve the task. In general, for tasks regarding generation, or tasks being solved using a standard auto-regressive LM, prefix prompts tend to be more conducive, as they mesh well with the left-to-right nature of the model. For tasks that are solved using masked LMs, cloze prompts are a good fit, as they very closely match the form of the pre-training task. Full text reconstruction models are more versatile, and can be used with either cloze or prefix prompts. Finally, for some tasks regarding multiple inputs such as *text pair classification*, prompt templates must contain space for two inputs, `[X1]` and `[X2]`, or more.

4.2 Manual Template Engineering

Perhaps the most natural way to create prompts is to manually create intuitive templates based on human introspection. For example, the seminal LAMA dataset (Petroni et al., 2019) provides manually created cloze templates to probe knowledge in LMs. Brown et al. (2020) create manually crafted prefix prompts to handle a wide variety of tasks, including question answering, translation, and probing tasks for common sense reasoning. Schick and Schütze (2020, 2021a,b) use pre-defined templates in a few-shot learning setting on text classification and conditional text generation tasks.

4.3 Automated Template Learning

While the strategy of manually crafting templates is intuitive and does allow solving various tasks with some degree of accuracy, there are also several issues with this approach: (1) creating and experimenting with these prompts is an art that takes time and experience, particularly for some complicated tasks such as semantic parsing (Shin et al., 2021); (2) even experienced prompt designers may fail to manually discover optimal prompts (Jiang et al., 2020c).

To address these problems, a number of methods have been proposed to automate the template design process. In particular, the automatically induced prompts can be further separated into *discrete prompts*, where the prompt is an

A Appendix on Pre-trained LMs

In this appendix we present some auxiliary information on pre-trained LMs that may be useful to the readers to better understand the current lay of the land with respect to this dynamic research area.

A.1 Evolution of Pre-trained LM Parameters

Fig. 7 lists several popular pre-trained models' statistics of parameters, ranging from 0 to 200 billion. GPT3, CPM2, and PanGu-α are the top three largest models with parameters greater than 150 billion.

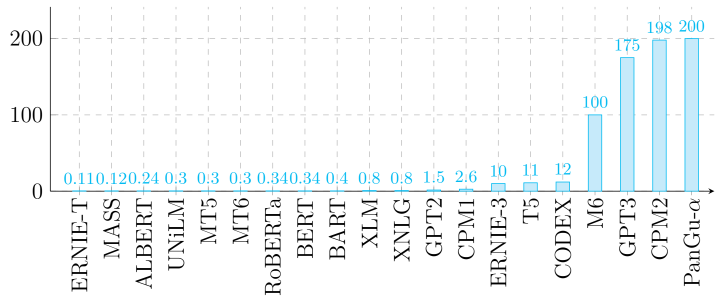


Figure 7: Comparison of the size of existing popular pre-trained language models.

A.2 Auxiliary Objective

In this subsection, more auxiliary objectives for pre-training language models have been listed.

- **Next Sentence Prediction (NSP)** (Devlin et al., 2019): A binary classification loss predicting whether two segments appear consecutively within a larger document, or are random unrelated sentences.
- **Sentence Order Prediction (SOP)** (Lan et al., 2020): A binary classification loss for predicting whether two sentences are in a natural or swapped order.
- **Capital Word Prediction (CWP)** (Liu et al., 2020b): A binary classification objective calculated over each word, predicting whether whether each word is capitalized or not.
- **Sentence Deshuffling (SDS)** (Liu et al., 2020b): A multi-class classification task to reorganize permuted segments.
- **Sentence distance prediction (SDP)** (Liu et al., 2020b) : A three-class classification task, predicting the positional relationship between two sentences (adjacent in the same document, not adjacent but in the same document, in different documents).
- **Masked Column Prediction (MCP)** (Yin et al., 2020): Given a table, recover the names and data types of masked columns.
- **Linguistic-Visual Alignment (LVA)** (Lu et al., 2019): A binary classification to Predict whether the text content can be aligned to visual content.
- **Image Region prediction (IRP)** (Su et al., 2020): Given an image whose partial features are masked (zeroed out), predict the masked regions.
- **Replaced Token Detection (RTD)** (Xiao et al., 2021): A binary classification loss predicting whether each token in corrupted input was replaced by a generative sample or not.
- **Discourse Relation Prediction (DRP)** (Sun et al., 2020): Predict the semantic or rhetorical relation between two sentences.
- **Translation Language Modeling (TLM)** (Lample and Conneau, 2019): Consider parallel sentences and mask words randomly in both source and target sentences.
- **Information Retrieval Relevance (IRR)** (Sun et al., 2020): Predict the information retrieval relevance of two sentences.
- **Token-Passage Prediction (TPP)** (Liu et al., 2020b): Identify the keywords of a passage appearing in the segment.
- **Universal Knowledge-Text Prediction (UKTP)** (Sun et al., 2021): Incorporate knowledge into one pre-trained language model.
- **Machine Translation (MT)** (Chi et al., 2021a) : Translate a sentence from the source language into the target language.
- **Translation Pair Span Corruption (TPSC)** (Chi et al., 2021a) : Predict the masked spans from a translation pair.
- **Translation Span Corruption (TSC)** (Chi et al., 2021a) : Unlike TPSC, TSC only masks and predicts the spans in one language.

- **Multilingual Replaced Token Detection (MRTD)** (Chi et al., 2021b): Distinguish real input tokens from corrupted multilingual sentences by a Generative Adversarial Network, where both the generator and the discriminator are shared across languages.
- **Translation Replaced Token Detection (TRTD)** (Chi et al., 2021b): Distinguish the real tokens and masked tokens in the translation pair by the Generative Adversarial Network.
- **Knowledge Embedding (KE)** (Wang et al., 2021): Encode entities and relations in knowledge graphs (KGs) as distributed representations
- **Image-to-text transfer (ITT)** (Wang et al., 2021): Is similar to the image caption that generates a corresponding description for the input image.
- **Multimodality-to-text transfer (MTT)** (Wang et al., 2021): Generate the target text based on both the visual information and the noised linguistic information.

A.3 Pre-trained Language Model Families

The increasing number of models makes it difficult for people to clearly grasp the differences between them. Based on this, we cluster the current mainstream pre-training models and characterize them from diverse dimensions.

A.3 Pre-trained Language Model Families

Family	Models	LM	Pre-training Tasks			Corruption				Application
			Main	Auxiliary	Parallel	Mask	Replace	Delete	Permute	
GPT	GPT [139]	L2R	SLM	-	✗	-	-	-	-	NLG
	GPT-2 [140]	L2R	SLM	-	✗	-	-	-	-	NLG
	GPT-3 [16]	L2R	SLM	-	✗	-	-	-	-	NLG
	Codex [20]	L2R	SLM	-	✗	-	-	-	-	NLG
ELMo	ELMo [130]	L2R	SLM	-	✗	-	-	-	-	NLU, NLG
BERT	BERT [32]	Mask	CTR	NSP	✗	Tok	-	-	-	NLU
	RoBERTa [105]	Mask	CTR	-	✗	Tok	-	-	-	NLU
	SpanBERT [70]	Mask	CTR	-	✗	Span	-	-	-	NLU
	DeBERTa [60]	Mask	CTR	-	✗	Tok	-	-	-	NLU
	SciBERT [7]	Mask	CTR	NSP	✗	Tok	-	-	-	Sci-NLU
	BioBERT [89]	Mask	CTR	NSP	✗	Tok	-	-	-	Bio-NLU
	ALBERT [87]	Mask	CTR	SOP	✗	Tok	-	-	-	mSent
	FinBERT [108]	Mask	CTR	CWP, SDS, SDP, TPP	✗	Span	-	-	Sent	Fin-NLU
	VLBERT [164]	Mask	CTR	IRP	✓	Tok, Region	-	-	-	VLU
	ViLBERT [110]	Mask	CTR	IRP, LVA	✓	Tok, Region	-	-	-	VLU
	BEIT [5]	Mask	CTR,FTR	-	✗	Visual “Tok”[7]	-	-	-	VLU
	VideoBERT [166]	Mask	CTR	LVA	✓	Tok, Frame	-	-	-	VLU
	TaBERT [189]	Mask	CTR	MCP	✓	Tok, Column	-	-	-	Tab2Text
	mBERT [32]	Mask	CTR	NSP	✗	Tok	-	-	-	XLU
	TinyBERT [69]	Mask	CTR	NSP	✗	Tok	-	-	-	XLU
ERNIE	ERNIE-T [199]	Mask	CTR	NSP	✗	Tok, Entity	-	-	-	NLU
	ERNIE-B [169]	Mask	CTR	-	✗	Tok,Entity, Phrase	-	-	-	NLU
	ERNIE-NG [183]	Mask	CTR	RTD	✗	N-gram	Tok	-	-	NLU
	ERNIE-B2 [168]	Mask	CTR	CWP,SDS,SOP, SDP,DRP,IRR	✗	Entity, Phrase	-	-	Sent	NLU
	ERNIE-M [126]	LPM	CTR	-	✓	Tok	-	-		XLU, XLG
	ERNIE-B3 [167]	Mask	CTR	SOP,SDP,UKTP	✗	Entity, Phrase	-	-	-	NLU
BART	BART [94]	En-De	FTR	-	✗	Tok	Span	Tok	Sent,Doc	NLU, NLG
	mBART [104]	En-De	FTR	-	✗	Span	-	-	Sent	NLG
UniLM	UniLM1 [35]	LPM	SLM,CTR	NSP	✗	Tok	-	-	-	NLU, NLG
	UniLM2 [6]	LPM	SLM,CTR	-	✗	Tok	-	-	Tok	NLU, NLG
T5	T5 [141]	En-De	CTR	-	✗	-	Span	-	-	NLU, NLG
	mT5 [186]	En-De	CTR	-	✗	-	Span	-	-	XLU, XLG
	mT6 [22]	En-De	CTR	MT,TPSC,TSC	✓	-	Span	-	-	XLU, XLG
	ByT5 [185]	En-De	CTR	-	✗	-	byte-span	-	-	XLU, XLG
XLM	XLM [86]	LPM	CTR	TLM	✓	Tok	-	-	-	XLU, XLG
	XLM-R [28]	Mask	CTR	-	✗	Tok	-	-	-	XLU
	XLM-E [23]	Mask	CTR	MRTD,TRTD	✗	-	Tok	-	-	XLU, XLG
CPM	CPM [200]	L2R	SLM	-	✗	-	-	-	-	NLG
	CPM-2 [198]	En-De	CTR	-	✗	Span	-	-	-	NLU,NLG
Other	XLNet [188]	L2R	SLM	-	✗	-	-	-	Tok	NLU
	PanGu-α [194]	L2R	SLM	-	✗	-	-	-	-	NLG
	ELECTRA [26]	Mask	CTR	RTD	✗	Tok	Tok	-	-	NLU,NLG
	MASS [162]	En-De	CTR	-	✗	Span	-	-	-	NLG
	PEGASUS [195]	En-De	CTR	-	✗	Tok, Sent	-	-	-	Summarization
	M6 [179]	En-De	CTR	ITT,MTT	✗	Span	-	-	-	NLG

Table 13: A detailed illustration of different pre-trained models characterized by the four aspects. “**Parallel**” represents if parallel data have been used for pre-training. `Sci`, `Bio`, `Fin`, `K` represent scientific, biomedical, financial, and knowledge, respectively. `Tok`, `Sent`, `Doc` denote token, sentence and document, respectively. `Region`, `Frame` denote basic units of images and video respectively.