arXiv:2201.05273v4 [cs.CL] 13 May 2022

Pre-trained Language Models for Text Generation: A Survey

JUNYI LI*, Renmin University of China, China and Université de Montréal, Canada
TIANYI TANG*, Renmin University of China, China
WAYNE XIN ZHAO†, Renmin University of China, China
JIAN-YUN NIE, Université de Montréal, Canada
JI-RONG WEN, Renmin University of China, China

Text Generation aims to produce plausible and readable text in a human language from input data. The resurgence of deep learning has greatly advanced this field, in particular, with the help of neural generation models based on pre-trained language models (PLMs). Text generation based on PLMs is viewed as a promising approach in both academia and industry. In this paper, we provide a survey on the utilization of PLMs in text generation. We begin with introducing three key aspects of applying PLMs to text generation: 1) how to encode the input into representations preserving input semantics which can be fused into PLMs; 2) how to design an effective PLM to serve as the generation model; and 3) how to effectively optimize PLMs given the reference text and to ensure that the generated texts satisfy special text properties. Then, we show the major challenges arisen in these aspects, as well as possible solutions for them. We also include a summary of various useful resources and typical text generation applications based on PLMs. Finally, we highlight the future research directions which will further improve these PLMs for text generation. This comprehensive survey is intended to help researchers interested in text generation problems to learn the core concepts, the main techniques and the latest developments in this area based on PLMs.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computing methodologies** → **Natural language generation**.

Additional Key Words and Phrases: Pre-trained Language Models, Natural Language Processing

ACM Reference Format:
Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2022. Pre-trained Language Models for Text Generation: A Survey. 1, 1 (May 2022), 35 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Text generation, also known as *natural language generation*, has been one of the most important sub-fields in natural language processing (NLP). It aims to produce plausible and readable text in a human language, from the input data in various forms including text, image, table and knowledge base. In the last decades, text generation techniques have been extensively applied to a wide range of applications. For example, they have been used in dialog systems to generate responses to user

*Equal Contribution
†Corresponding author

Authors' addresses: Junyi Li, Renmin University of China, Beijing, China and Université de Montréal, Montréal, Canada, lijunyi@ruc.edu.cn; Tianyi Tang, Renmin University of China, Beijing, China, steventianyitang@outlook.com; Wayne Xin Zhao, Renmin University of China, Beijing, China, batmanfly@gmail.com; Jian-Yun Nie, Université de Montréal, Montréal, Canada; Ji-Rong Wen, Renmin University of China, Beijing, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Association for Computing Machinery.
XXXX-XXXX/2022/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

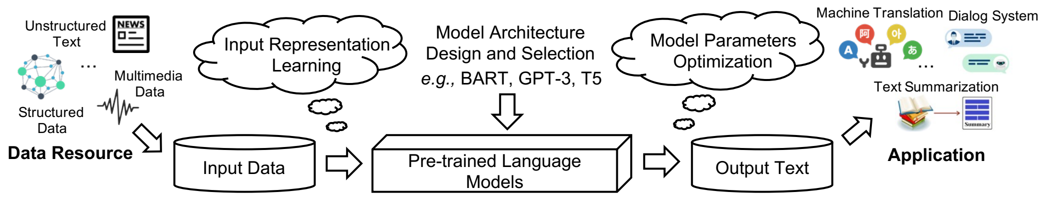

Fig. 1. An illustrative process of applying PLMs to text generation. We divide the process into three main steps: input representation learning, model architecture design and selection, model parameters optimization.

utterances in a conversation [224], in machine translation to translate a text from one language into another [31]; and in text summarization to generate an abridged summary of the source text [40].

The primary goal of text generation is to automatically learn an input-to-output mapping from the data to construct an end-to-end solution with minimal human intervention. This mapping function allows the generation system to generalize in a broader field and to generate free text under the given conditions. Earlier approaches usually adopt statistical language models for modeling the conditional probabilities of words given an n-gram context [11, 13]. Such a statistical approach is known to suffer from the data sparsity issue, and a number of smoothing methods have been developed to alleviate this problem so as to better estimate unobserved term occurrences [179, 210]. Still, word tokens are used as the basic representation units in these approaches, which leads to the issue that similar tokens cannot be easily mapped with each other.

With the emergence of deep learning techniques [96], neural network models have dominated the mainstream methods in text generation and make exceptional success in generating natural language texts. Deep neural generation models usually adopt the sequence-to-sequence framework [177] based on the encoder-decoder scheme: the encoder first maps the input sequence into fix-sized low-dimensional vectors (called *input embeddings*), and then the decoder generates a target text based on the input embeddings. The representation by embeddings makes a key difference from earlier statistical approaches, which makes it easier to cope with the possible relations between inputs and outputs. Various neural models have been proposed with different designs for the encoder-decoder architecture, such as graph neural networks (GNN) for encoding graph inputs [102] and recurrent neural networks (RNN) for decoding texts [108]. Besides, the attention mechanism [2] and copy mechanism [164] are widely used to improve the performance of text generation models. An important merit of deep neural networks for text generation is that they enable end-to-end learning of semantic mappings from the input data to output texts without labor-intensive feature engineering. Moreover, deep neural models employ low-dimensional semantic representations [82] to capture linguistic features of language, which is useful to alleviate data sparsity.

Despite the success of deep neural models for text generation, a major performance bottleneck lies in the availability of large-scale labelled datasets. Most of text generation methods require substantial amounts of manually labelled parallel data, which restricts their applicability in many domains that suffer from a dearth of annotated examples. To date, most of existing labelled datasets for text generation tasks are usually small. In such cases, deep neural networks are likely to overfit on these small datasets and do not generalize well in practice. Moreover, the early neural models for text generation were still relatively shallow with only 1~3 neural layers. Therefore, these models have difficulties in modeling intricate relationships between the context and word meanings and deriving contextual word representations for better generation [151].

In recent years, the paradigm of pre-trained language models (PLMs) is thriving in NLP [151]. The basic idea is to first pre-train the models on large-scale unsupervised corpora and then fine-tune these models in downstream supervised tasks. Such a pretraining-finetuning framework achieved state-of-the-art performance. With the emergence of Transformer [180] and higher computational power, the architecture of PLMs has evolved from shallow to deeper architectures, such as BERT [35] and OpenAI GPT [153]. Substantial work has shown that PLMs can encode massive amounts of linguistic knowledge from the pre-training corpora into their large-scale parameters and learn universal and contextual representations of the language with specially designed objectives such as masked token prediction. Therefore, PLMs are generally beneficial for downstream tasks and can avoid training a new model from scratch. Following the success of PLMs in other NLP tasks, researchers have proposed to apply PLMs to text generation tasks with several steps (see Figure 1) [14, 99, 154]. Pre-trained on large-scale corpora, PLMs can understand natural language accurately and further express in human language fluently, both of which are critical abilities to fulfill text generation tasks. Grounding text generation on PLMs is seen as a promising direction in both academia and industry, which has much advanced the state of the art in this field. Thus, in this survey, we focus on text generation based on large PLMs.

There are a number of survey papers on text generation and on PLMs. For example, Qiu *et al.* [151] summarized two generations of PLMs for the whole NLP domain and introduced various extensions and adaption approaches of PLMs. Kalyan *et al.* [88] gave a brief overview of the advances of self-supervised learning in Transformer-based PLMs. Han *et al.* [70] took a deep look into the history of pre-training, especially its special relation with transfer learning and self-supervised learning. Besides, El-Kassas *et al.* [40] focused on the current application of PLMs to the field of text summarization. Zaib *et al.* [206] discussed the application of PLMs to dialog systems with a special emphasis on question answering systems. These surveys focused on specific applications, *e.g.,* summarization and dialogue systems, but did not go deep into the core technique, *i.e.,* text generation. As text generation is a key component in various applications, it is useful to provide a comprehensive survey on the topic of text generation based on PLMs. Differenbt from the existing surveys, this survey is intended to provide a more general description on this common task, rather than limiting it to a specific type of application. It is worth noting that this survey is an extended version of the short survey [106]. The extensions include: (1) This paper covers a wider range of existing studies, evaluation protocols, open-source libraries, and common applications of PLMs-based text generation. This goes far beyond the scope of the previous short survey; (2) This paper provides a new schematic view involving three key aspects (*i.e.,* input data, model architecture, parameter optimization) about applying PLMs to text generation, which constitute the main content of this paper; (3) To provide a better picture of the existing solutions for various challenges, this paper includes more detailed descriptions and discussions about the their technical contributions.

The remainder of this survey is organized as follows. We first present the task formulation and an overview of PLMs in Section 2. Given the encoded input data, the goal of text generation is to optimize the generation function (*i.e.,* PLMs) for generating satisfactory output text. Thus, three key points are involved when applying PLMs to text generation: 1) how to encode the input data into representations preserving input semantics which can be fused into PLMs (Section 3); 2) how to design an effective PLM to serve as the generation function (Section 4); and 3) how to optimize PLMs given the reference text and to ensure that the generated texts satisfy special text properties (Section 5). Then, we discuss several typical non-trivial challenges and solutions within each key point in Section 6. We present a summary of various useful resources to work with PLMs in Section 7 and common applications in Section 8. Finally, we summarize the contribution of this survey and describe future directions in Section 9.

2 PRELIMINARY

In this section, we first give a general task definition of text generation, then describe the background of PLMs, and finally introduce the three key aspects on PLM-based text generation methods.

2.1 Text Generation

Generally, a text can be modeled as a sequence of tokens $y = \langle y_1, ..., y_j, ..., y_n \rangle$, where each token y_j is drawn from a vocabulary $\mathcal{V}$. The task of text generation aims to generate plausible and readable text in a human language. In most cases, text generation is conditioned on some input data (*e.g.,* text, image, tabular, and knowledge base), which is denoted as x. In particular, the generated text is expected to satisfy some desired language properties such as fluency, naturalness, and coherence. We denote the desired properties for output text as a property set $\mathbb{P}$. Based on the above notations, the task of text generation can be formally described as:

$$y = f_{\mathcal{M}}(x, \mathbb{P}), \tag{1}$$

where the text generation model $f_{\mathcal{M}}$ produces the output text y given the input data x, satisfying some special proprieties from the property set $\mathbb{P}$. In this survey, the text generation model $f_{\mathcal{M}}$ is specially crafted based on a PLM $\mathcal{M}$.

Specifically, according to the type of the input data x and the property set $\mathbb{P}$, text generation can be instantiated into different kinds of tasks:

• When the input data x is **not provided** or is **a random vector**, text generation will degenerate into language modeling or unconditional text generation [152, 153]. In this case, the output text is required to satisfy some common language properties, such as fluency and naturalness.

• When the input data x is a set of **discrete attributes** (*e.g.,* topic words and sentiment labels), it becomes topic-to-text generation [33] or attribute-based generation [90]. The input data x plays the role of controlling the content of the generated text. In such a situation, the output text should be relevant to the input topics or adhere to the required attributes.

• When the input data x is **structured data** such as knowledge base or table, it is considered as data-to-text generation [60, 105]. This task aims to generate a descriptive text about the structured data. Therefore, the output text should be objective and accurate.

• When the input data x is **multimedia input** such as image and speech, it becomes image captioning [191] or speech recognition [45]. We may expect that the caption text be lively for attracting children's attention, and the converted speech text be faithful to the original speech.

• The most common form of input data x is **a text sequence**. This form spans a number of applications such as machine translation [31], text summarization [161] and dialog system [215]. For a specific task, the output text is expected to satisfy desired properties. For example, the summaries in text summarization should not contradict the facts described in the input text, and the responses in dialog should be relevant to the input dialog history and context.

2.2 Pre-trained Language Models

Pre-trained language models (PLMs) are deep neural networks that are pre-trained on large-scale unlabelled corpora, which can be further fine-tuned on various downstream tasks. It has been shown that PLMs can encode a significant amount of linguistic knowledge into their vast amounts of parameters [106, 158]. Therefore, it is promising to apply PLMs to enhance the understanding of language and improve the generation quality.

Owing to the great success of Transformer [180], almost all PLMs employ it as the backbone. As two typical PLMs, GPT [152] and BERT [35] are first built upon Transformer decoder and encoder respectively. Following GPT and BERT, PLMs such as XLNet [198], RoBERTa [124], ERNIE [217], T5 [154] and BART [99] are propopsed in the literature. Among them, XLNet, RoBERTa and ERNIE are developed based on the BERT model, while T5 and BART are encoder-decoder based PLMs. Recent studies have shown that the performance of PLMs can be boosted by increasing the scale of model parameters [89], which triggered the development of large-scale PLMs such as GPT-3 (175B) [14], PANGU (200B) [207], GShard (600B) [98] and Switch-Transformers (1.6T) [46], which consist of billions or trillions of parameters. In addition, PLMs are designed for other tasks such as named entity recognition [147], programming [49], and networking [127]. According to the pre-training objectives, PLMs for text generation can be categorized as masked LMs, causal LMs, prefix LMs, and encoder-decoder LMs, which will be detailed in Section 4.

2.3 PLM-based Text Generation Methods

To effectively leverage PLMs for downstream text generation tasks, we need to consider three key aspects from the perspectives of data, model, and optimization, respectively:

- **Input Data**: *How to encode the input x into a representation preserving the input semantics that can be fused into the PLM $\mathcal{M}$?* For text generation, the input data, containing critical semantic information for the target output, often appears in various data types for different tasks (*e.g.,* sequential text, structured table, multimedia), whereas most PLMs are typically pre-trained on the sequential text data. Therefore, it is a major challenge to develop effective, flexible representation learning approaches for PLMs to capture semantic information from various types of input data.
- **Model Architecture**: *How to design an effective PLM $\mathcal{M}$ to serve as the generation function $f_{\mathcal{M}}$ and adapt to various text generation tasks?* In the literature, a number of PLMs have been developed with generalized architectures for general purposes (*e.g.,* denoised auto-encoder [99] or auto-regressive decoder [153]), While these general architectures cannot cope with some special text generation cases. Therefore, it is important to make specific designs on the underlying PLMs for achieving good task performance when adapting to different text generation tasks.
- **Optimization Algorithm**: *How to optimize the text generation function (i.e., PLMs) $f_{\mathcal{M}}$ given the reference text y and ensure that the generated text satisfies special text properties $\mathbb{P}$?* In order to produce satisfactory text, it is critical to learn the text generation function by developing effective optimization algorithms. A major challenge stems from the fact that some desired properties for output text are difficult to be formulated or optimized.

In the following sections, we will present recent research efforts on PLM-based text generation, with an emphasis on the three aforementioned aspects. The overall organization of our description follows the schema shown in Figure 2.

3 ENCODING INPUT REPRESENTATIONS

As discussed in Section 2, the first aspect is the encoding of input data x into meaningful representations preserving input semantics for PLMs. In this section, we will present three main types of input data for text generation, *i.e.,* unstructured input, structured input, and multimedia input.

3.1 Unstructured Input

In text generation, most studies focus on modeling unstructured text input (*e.g.,* sentence, paragraph, and document), which requires to accurately understand the input information and derive meaningful text representations. The aim of text representation learning is to condense the input text into low-dimensional vectors that can preserve the core semantic meanings. In what follows, we will discuss how to derive effective semantic representations for three kinds of unstructured text data, namely paragraphs, documents and multi-lingual texts.

3.1.1 Paragraph Representation Learning.
A paragraph usually consists of multiple sentences describing different topics and each sentence contains a sequence of words. To capture both low-level word meanings and high-level topic semantics in a paragraph, many studies proposed hierarchy-based or graph-based methods to learn the paragraph representation.

Hierarchy-based Representation Learning. For a multi-sentence paragraph such as a multi-turn dialogue, a typical approach is to concatenate sentences as a whole text and predict the output

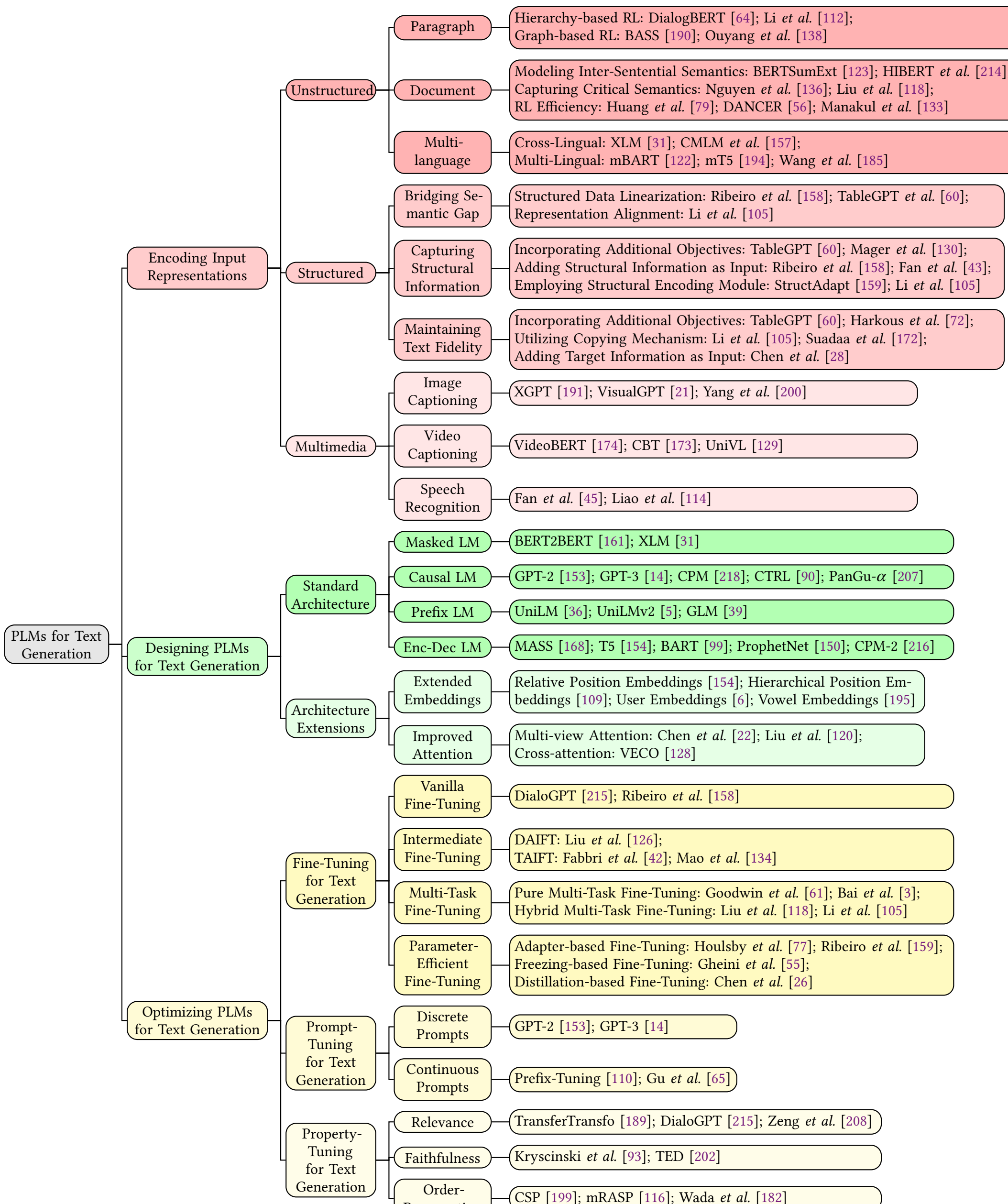

Fig. 2. The main content flow and categorization of this survey.

text [7, 215]. However, flat concatenation cannot effectively capture the semantic dynamics across utterances which is likely to cause inaccurate generation. To deal with this issue, hierarchical encoders have been proposed to model the input paragraph [64, 112]. Gu *et al.* [64] represented the dialogue context using DialogBERT, a hierarchical framework that utilizes sentence- and discourse-level Transformer encoders to encode each dialogue utterance and the sequence of utterance vectors, respectively. However, when encoding each individual utterance, it does not consider the history information, which is essential for understanding dialogue utterances. Thus, Li *et al.* [112] employed a Transformer to encode each utterance into a dense vector, upon which a left-to-right flow module was designed to capture the utterance-level dynamic information flow.

Graph-based Representation Learning. A long paragraph is likely to contain repeated, redundant or contradictory information. How to exploit the key semantics and remove minor information from the intricate paragraph text is critical to promote paragraph-based generation performance. Compared with sequences, by explicitly representing words or phrases as nodes and their relations (*e.g.,* similarity) as edges, graphs can easily aggregate relevant but disjoint context in the text [138, 190]. As a representative example, Wu *et al.* [190] leveraged a phrase-level unified semantic graph, where nodes are phrases extracted by dependency parsing and relations are dependency relations. This graph can be used to aggregate co-referent phrases that are scattered in context for better capturing the long-range relations and global paragraph structures. Besides, in conversational machine reading, Ouyang *et al.* [138] formulated the input text as two complementary graphs, *i.e.,* explicit and implicit discourse graphs, to fully capture the discourse relations and latent vector interactions among all the elementary discourse units.

3.1.2 Document Representation Learning. In many text generation tasks such as document translation and document summarization, the input text might be a long document consisting of multiple paragraphs. When encoding the documents, it is challenging to model cross-sentence (paragraph) semantics and capture the most critical semantics.

Modeling Inter-Sentential Semantics. Most of PLMs are trained as masked language models. They mainly focus on learning token-level representations instead of sentence-level ones. Although segment embeddings are used to represent different sentences separately, they cannot capture the cross-sentence semantics. To encode inter-sentential semantics, several studies [123, 214, 222] proposed to learn document representations in a hierarchical way. For example, Liu *et al.* [123] inserted the "[CLS]" token at the beginning of each sentence to aggregate sentence-level features in lower layers and then combine them with self-attention in higher layers. Besides, Zhang *et al.* [214] proposed HIBERT for learning document representations in a hierarchical fashion by using a sentence encoder to map sentences into sentence vectors and a document encoder to further learn context-sensitive sentence representations given their surrounding sentence vectors as context.

Capturing Critical Semantics. In practice, sentences or paragraphs in long documents will inevitably complement, overlap, or conflict with one another. Therefore, it is necessary to retain the most critical contents and verbalize them in the generated text. To address the issue of key points missing in output text, Nguyen *et al.* [136] introduced a topic model to capture the global topic semantics of the document and a gate mechanism to control the amount of global semantics provided to the text generation module. Similarly, Liu *et al.* [118] proposed two topic-aware contrastive learning objectives, among which the coherence detection objective identifies topics of a dialogue by detecting the coherence change among topics and the sub-summary generation objective forces the model to capture the most salient information and generate a sub-summary for each topic.

Representation Learning Efficiency. Efficiency is a crucial aspect for modeling long documents, especially when generating long text. Since the self-attention mechanism grows quadratically with sequence length, a number of studies aimed to improve the encoding efficiency of self-attention [79, 133]. A representative example is Manakul *et al.* [133], which proposed local self-attention, allowing longer input spans during training; and explicit content selection, reducing memory and compute requirements. Furthermore, several researchers adopted divide-and-conquer encoding methods. By splitting the long document into short sentences, it is easier to summarize each short part of the document separately [56], reducing the computational complexity.

3.1.3 Multi-lingual Representation Learning. Existing PLMs are mainly pre-trained on English text while ignoring other low-resource languages. It is difficult to apply English-based PLMs to solve multi-lingual text generation tasks (*e.g.,* multi-lingual machine translation). Several approaches have been proposed to cope with multilingual texts.

Cross-lingual Representations. The core idea of cross-lingual representation learning is to learn a shared embedding space for two languages, in order to improve PLMs' ability to translate between them. A well-known cross-lingual PLM is XLM [31], which leveraged both monolingual and parallel data to learn cross-lingual representations. However, these learned representations on shared Byte-Pair Encoding (BPE) spaces is implicit and limited. Therefore, Ren *et al.* [157] further computed cross-lingual n-gram embeddings and derived an n-gram translation table based on them for providing explicit representation learning signals.

Multi-lingual Representations. Given more than two languages, multi-lingual PLMs aim to learn representations for any of the languages. Based on English PLMs, BART and T5, Liu *et al.* [122] and Xue *et al.* [194] proposed mBART and mT5, respectively, which are pre-trained once for all languages. Considering the differences across languages (*e.g.,* syntactic rules), several studies utilized contrastive learning to learn multi-lingual representations [139, 185]. In particular, Wang *et al.* [185] proposed two training objectives: contrastive sentence ranking (CSR) and sentence aligned substitution (SAS). CSR creates positive and negative sentence pairs based on their saliency scores, while SAS replaces sentences with those in another language. By contrastively learning these languages in a common text, the model can learn shared representation spaces across languages.

3.2 Structured Input

Structured data (*e.g.,* table, graph, and tree) is a critical kind of input for text generation in many real-world applications, such as medical report [73] and weather report [57] generation. However, it is non-trivial to model structured input for PLMs due to three major challenges: (1) there exists a semantic gap between structured data and PLMs, since PLMs are typically pre-trained on natural language texts; (2) it is non-trivial to encode the structural information in the input data; (3) it requires to maintain fidelity of the generated text with respect to the input.

3.2.1 Bridging the Semantic Gap. In general, PLMs are pre-trained on unstructured text, which differs in form from the structured data. Several methods have been proposed to bridge this gap.

Structured Data Linearization. In order to fit the structured input for PLMs, a simple approach is to linearize the input data into a *sequence* [43, 130, 158]. Specifically, Ribeiro *et al.* [158] linearized knowledge graph (KG) into a sequence of triples by concatenating the relational triples. Besides, some studies adopted template-based heuristic methods to serialize the input data [60]. For example, the attribute-value pair "*name: james beattie*" will be serialized as a sentence "*name is james beattie*".

Representation Alignment. The semantic gap makes it difficult to effectively inject structured data representations into PLMs while directly serializing structured data. Therefore, some people proposed to align the structured data representations with PLM-based word embeddings in semantic spaces. For example, Li *et al.* [105] utilized graph neural networks (GNN) to project KG entities into embeddings, and then performed representation alignment by minimizing the Euclidean distance between the GNN-based and PLM-based entity embeddings.

3.2.2 Capturing the Structural Information. An important feature of structured data is that it represents data in a structural way, such as the $\langle attribute, value\rangle$ pair in table or the $\langle head, relation, tail\rangle$ triple in KB. Such structural information can be used to help generate faithful text by modeling the input in a more accurate way.

Incorporating Additional Training Objectives. To enhance the preservation of structural information, a typical approach is to incorporate auxiliary training objectives related to structural information [60, 105, 130]. One kind of objectives is to reconstruct the semantic structure of the input data. For example, Gong *et al.* [60] utilized the attribute names of input tables as the labels to reconstruct table structure based on the attribute value representations from PLMs, which enforces PLMs to embed table structure into table representations. Another method is to adjust the output text based on the structural information. Mager *et al.* [130] proposed cycle-consistency based losses to assess the quality of output text based on how well it can reconstruct the input structure.

Adding Structural Information as Input. As opposed to prior studies that implicitly capture structural information with training losses, several studies explicitly took structural information as input [43, 158]. Ribeiro *et al.* [158] directly prepended "⟨H⟩", "⟨R⟩", and "⟨T⟩" tokens before the head entity, relation and tail entity of a KG triple to reveal the relations between entities. Besides, Fan *et al.* [43] used the graph embedding of an Abstract Meaning Representation (AMR) graph as input. The graph embedding provides the graph structure information by encoding the depth of each node (from the node to the root node) and the subgraph each node belongs to.

Employing Structural Encoding Module. Since PLMs are originally developed for sequential input, it makes sense to incorporate additional modules to encode the structured input. A representative example is StructAdapt [159], which adds layer-wise graph convolution modules to learn representations built upon the graph connectivity over the PLM encoder. Similarly, Li *et al.* [105] employed GNN to encode KG relations as embeddings, which will be taken as input of PLMs.

3.2.3 Maintaining Text Fidelity. In the literature of linguistics [18], fidelity means the generated text adheres to the content in the structured data. Generating high-fidelity text that correctly describes the information of structured input is the key to data-to-text generation algorithms.

Incorporating Additional Training Objectives. To generate high-fidelity text adhereing to input, Gong *et al.* [60] introduced an Optimal-Transport based content matching loss that measures the distance between the input information and the output text. Harkous *et al.* [72] employed a semantic fidelity classification loss to detect and avoid generation errors such as hallucination.

Utilizing Copy Mechanism. The pointer-generator [164] is a typical method to ensure the faithfulness of generated text about input data by copying important words from input into output. For example, Li *et al.* [105] adopted pointer-generator to copy entities from input knowledge data to output text, and Suadaa *et al.* [172] copied table values into general placeholders to avoid producing hallucinated phrases that do not appear in the input table.

Adding Target Information as Input. To combat with the low-fidelity problem, Chen *et al.* [28] argued that it is important to leverage intermediate meaning representations to achieve faithful generation. Therefore, the authors enhanced the generation module with a logical form representing the semantics of the target text.

3.3 Multimedia Input

In addition to the above textual data, multimedia data (*e.g.*, image, video, and speech) has also been utilized as input of text generation algorithms, *e.g.*, image captioning and speech recognition.

3.3.1 Image Captioning. Image captioning, which aims to generate a textual description for an image, has been extensively studied in the field of computer vision (CV). Many studies have proposed multi-modal PLMs to combine textual and visual modalities. A well-known multi-modal PLM is XGPT [191]. Inspired by text-based GPT, XGPT takes images as inputs and uses the image captioning task as the pre-training task in the pre-training stage. Chen *et al.* [21] also proposed an image captioning PLM, called *VisualGPT*. They designed a self-resurrecting attention mechanism to learn how to encode the visual information and adapt it to the PLM decoder. However, traditional vision-language pre-training fails to capture the relationship between the visual and text modalities.

Yang *et al.* [200] proposed three pre-training tasks to effectively learn better aligned representations among three kinds of input data: text word, visual object, and scene text.

3.3.2 Video Captioning. Video captioning focuses on generating natural language text that can describe the video content. VideoBERT [174] and CBT [173] are two early attempts to investigate video-language pre-training with regard to the video captioning task. However, previous studies usually adopted one single encoder-decoder framework, which is not flexible for diverse downstream tasks. UniVL [129] employed two single-modal encoders to encode text and video separately and a sentence decoder to generate video captions.

3.3.3 Speech Recognition. In practice, speech recognition is hungry for human-transcripted supervised data. Thus, a number of unsupervised and semi-supervised methods were developed to integrate PLMs for weakly-supervised learning. For example, Fan *et al.* [45] proposed an unsupervised approach to pre-training encoder-decoder model with unpaired speech and transcripts. Liao *et al.* [114] proposed a speech recognition post-processing model that attempts to transform the incorrect and noisy recognition output into natural language text for humans and downstream tasks by leveraging the Metadata Extraction (MDE) corpus to construct a small task-specific dataset.

4 DESIGNING PLMS FOR TEXT GENERATION

After encoding the input data into low-dimensional representations, the next step is to develop an effective PLM $\mathcal{M}$ as the text generation function $f_{\mathcal{M}}$. Based on such an architecture of PLM, the text generation objective can be modeled as the conditional probability of the output text y given the input data x, which can be formally factorized by tokens:

$$\Pr_{\mathcal{M}}(y|x) = \prod_{i=1}^{n} \Pr_{\mathcal{M}}(y_i|y_{<i}, x), \tag{2}$$

where y_i denotes the i-th output token, and $y_{<i}$ denotes the previous tokens $y_1, ..., y_{i-1}$.

To compute the conditional probability, traditional neural models mainly adopt the RNN architecture [177] with several variants [164]. In recent years, solely based on attention mechanisms, Transformer [180] can better capture long-range dependency in texts, which is beneficial for modeling and generating texts. With the excellent parallelization capacities, Transformer has become the backbone for developing very large PLMs. When trained on large-scale unlabeled corpora [124], PLMs built on the Transformer architecture can encode rich semantic or linguistic knowledge. Furthermore, it has been shown that PLMs can be effectively fine-tuned to different text generation tasks [99, 168]. All these make PLMs the first choice to implement the text generation function $f_{\mathcal{M}}$.

4.1 Standard Architecture

Existing PLMs for text generation adopt either a single Transformer or a Transformer-based encoder-decoder as the backbone. PLMs, such as GPT-3 [14] and UniLM [36], use a single Transformer encoder/decoder to simultaneously implement the process of input encoding and output decoding. This includes three major variants: masked LMs, causal LMs, and prefix LMs, with different attention mask strategies. In contrast, PLMs built upon Transformer encoder-decoder perform input encoding and output decoding separately. In the following, we describe these four variants in detail.

4.1.1 Masked Language Models. Masked LMs use a full-attention Transformer encoder. Equipped with the full attention, models are usually pre-trained with masked language modeling (MLM) task, *i.e.,* predicting the masked tokens using the bidirectional information. The most representative model is BERT [35], which is used extensively in natural language understanding (NLU).

However, due to the discrepancy between the pre-training task of masked LMs and the downstream generation function, masked LMs are rarely utilized for text generation tasks [198]. It is more common to use masked LMs as the encoder part for text generation, allowing to leverage the excellent bidirectional encoding capacities. For example, Rothe *et al.* [161] proposed to initialize both the encoder and decoder of the generation model with BERT [35], which yields comparable performance with other PLMs specially designed for text generation.

4.1.2 Causal Language Models. Similar to Transformer decoder, causal LMs adopt the diagonal mask matrix. Causal LMs are designed for language modeling, which is to determine the probability of a given sequence of words occurring in a sentence. Causal LMs are straightforward for text generation, predicting the next word conditioned on all previous words.

In the literature, GPT [152] was the first causal LM for the text generation task. Then, GPT-2 [153] explored the transfer capacity of language models for zero-shot generation task, highlighting the significance of sufficient data. Furthermore, GPT-3 [14] showed that massive model parameters can significantly improve the downstream generation tasks, with a few examples or prompts. CTRL [90] is proposed as a conditional causal LM to generate text based on control codes that govern style, content, and task-specific behavior. Causal LMs are simple and straightforward for text generation, but they have several structural and algorithmic limitations: Causal LMs encode the tokens just from left to right, thus ignore the bidirectional information on the input side. Moreover, causal LMs are not specially designed for the sequence-to-sequence generation tasks, thus in practice they do not achieve high performance in tasks such as summarization and translation [153].

4.1.3 Prefix Language Models. Upon a single Transformer, prefix LMs adopt bidirectional encoding scheme in the input side and natural left-to-right generation pattern in the output side. By utilizing the mixture attention mask, the tokens in the input text x can attend to each other, while the tokens in the target text y can only attend to all input tokens and previous generated tokens.

UniLM [36] was the first prefix LM. Compared to causal LMs, UniLM used prefix attention mask to solve conditional generation tasks, similar to the encoder-decoder architecture. UniLMv2 [5] and GLM [39] improved vanilla prefix masking strategy by introducing permuted language modeling in XLNet [198]. Although prefix LMs have several advantages, Raffel *et al.* [154] compared single-Transformer prefix LMs to Transformer-based encoder-decoder LMs and concluded that adding explicit encoder-decoder attention is more effective to capture conditional dependencies.

4.1.4 Encoder-Decoder Language Models. Encoder-decoder LMs follow the standard Transformer architecture for text generation, consisting of stacks of both encoder and decoder layers. During pre-training, MASS [168] and ProphetNet [150] took the sequence with one masked segment as the input of encoder and then the decoder generates the masked tokens in an auto-regressive way. T5 [154] randomly replaced several spans in the source text with different special tokens, and then the decoder predicted every replaced span in turn. BART [99] was pre-trained with denoising auto-encoder (DAE), *i.e.,* the model learns to recover the original text from corrupted text, which is corrupted with different noising methods, such as sentence permutation and token deletion.

4.2 Architecture Extensions

To derive performant PLMs for text generation, many studies proposed to improve the Transformer backbone of PLMs. In this part, we will introduce two major improved techniques, *i.e.,* extended input embeddings and improved attention mechanism.

4.2.1 Extended Input Embeddings. Besides (sub-)word embeddings, almost all PLMs use position embeddings to indicate the indices of input words. Compared to CNN and RNN, the self-attention operation is usually order-independent. Hence, it is essential to provide explicit position information to capture the sequential nature of text. Original Transformer [180] utilized the pre-determined absolute position embeddings with sinusoidal functions, while most PLMs (*e.g.,* BERT and GPT) adopted learned absolute position embeddings. Instead of absolute ones, relative position embeddings produce position embeddings according to the offset between two tokens. For example, T5 [154], UniLMv2 [5] and ProphetNet [150] employed an bucket relative positional method. In addition, hierarchical position embeddings are utilized to indicate inter- and intra- sentence position information, which is often used in some fixed-format text such as poem [109] and lyric [195].

Moreover, it is necessary to incorporate auxiliary embeddings to enrich the input information [88]. Similar to segment embeddings used in BERT, dialogue state embeddings [6, 189] are used to assign each utterance, and user embeddings [6, 69] are utilized to differentiate characters involved in a conversation. In the multilingual scenario, language embedding [30, 168] is commonly introduced to inform the model about the language of each sentence. In addition, rhyme embeddings [109] and vowel embeddings [195] are proposed to indicate acoustics information in poem and lyric.

4.2.2 Improved Attention Mechanism. Although there exist various modules in Transformer (*e.g.,* position-wise FFN, self-attention, etc.), related works mainly focused on improving the self- and cross-attention mechanism for text generation [88]. In order to adapt to long-form text input and alleviate quadratic complexity of full-attention computation, sparse attention is proposed to replace the original self-attention for long-form input. Rather than attending to all other tokens, every token only attends to specific tokens with strategies such as window attention [133, 143, 205], global attention [143, 205], random attention [205] and Sinkhorn attention [223].

In practice, many text generation tasks need to process input data from multiple sources. It is common to leverage one or more encoders to encode multiple inputs. Therefore, several works proposed to utilize different strategies to aggregate multi-source inputs in the cross-attention module. Golovanov *et al.* [58] conducted mean pooling for dialogue history, current state and persona information. Chen *et al.* [22] and Liu *et al.* [120] proposed multi-view attention and knowledge-aware attention to process embeddings from multiple views or knowledge sources. In addition, VECO [128] pluged a cross-attention technique into the Transformer encoder to explicitly build the inter-dependence between multiple languages. BASS [190] and Ribeiro *et al.* [159] substituted the self-attention module with GNN to better extract structural information. Zeng *et al.* [209] appended the gating mechanism after self-attention to inject condition-aware information.

5 OPTIMIZING PLMS FOR TEXT GENERATION

To obtain good performance, it is critical to develop effective optimization algorithms for PLM-based text generation models. We consider three main types of optimization methods, namely fine-tuning, prompt-tuning, and property-tuning. We will detail each optimization method below.

5.1 Fine-Tuning for Text Generation

During pre-training, PLMs are able to capture general linguistic knowledge from large-scale corpora. However, it requires task-specific knowledge to perform downstream text generation tasks. For this purpose, fine-tuning is a popular approach to incorporating task-specific information into PLMs by adjusting their weights using downstream text generation datasets [153].

According to how the parameters of PLMs are updated [88], exiting fine-tuning methods for text generation can be categorized as 1) vanilla fine-tuning, 2) intermediate fine-tuning, 3) parameter-efficient fine-tuning, and 4) multi-task fine-tuning. Compared with vanilla fine-tuning, intermediate and multi-task fine-tuning can alleviate the overfitting issue on small text generation datasets to some extent. As the vanilla fine-tuning requires adjusting the entire model, parameter-efficient methods such as adapters [77] can fine-tune PLMs in a lightweight manner.

5.1.1 Vanilla Fine-Tuning. Vanilla fine-tuning directly updates PLMs using downstream text generation datasets with task-specific losses (*e.g.,* cross-entropy loss [153]). Zhang *et al.* [215] trained the DialoGPT model on the basis of the GPT-2 architecture by modeling a multi-turn dialogue session as a long text and optimizing the generation model with language modeling objective. Ribeiro *et al.* [158] investigated two recent PLMs, BART and T5, for graph-to-text generation and fine-tuned them using the typical auto-regressive cross-entropy loss. A major issue of vanilla fine-tuning is that it is often not sufficiently optimized on small datasets, which is prone to overfitting.

5.1.2 Intermediate Fine-Tuning. The basic idea of intermediate fine-tuning is to incorporate an intermediate dataset consisting of sufficient labeled instances. The intermediate dataset can focus on the same target text generation task but from a different domain, or a similar NLP task from the same target domain. It is helpful to infuse domain- or task-specific knowledge from the intermediate dataset to alleviate the overfitting issue and enhance the performance on small target text generation datasets [146]. According to the relatedness between the intermediate dataset and the target text generation dataset [88], intermediate fine-tuning can be divided into two categories, *i.e.,* domain adaptive intermediate fine-tuning (DAIFT) and task adaptive intermediate fine-tuning (TAIFT).

Domain Adaptive Intermediate Fine-Tuning. According to Kalyan *et al.* [88], DAIFT utilizes an intermediate dataset, which focuses on a similar NLP task (not text generation tasks) from the same target domain, consisting of sufficient labeled instances. By leveraging such an intermediate dataset, PLMs can be enriched with domain-specific knowledge, which is helpful to improve the performance of the target text generation task within the same domain. DAIFT is commonly used in machine translation to eliminate the issue of unseen languages in translation pairs. For example, to improve the translation quality of the low-resource target language (*e.g.,* Kazakh), Liu *et al.* [126] constructed a large-scale intermediate monolingual corpus of the target language and fine-tuned mBART by reconstructing the corrupted target-language text. The intermediate dataset comes from the same language domain as the target dataset (*e.g.,* Kazakh), which can impart language-related linguistic knowledge to PLMs for a better translation performance.

Task Adaptive Intermediate Fine-tuning. In contrast with DAIFT, TAIFT incorporates an intermediate dataset on the same target text generation task but from a different domain. It aims to infuse task-specific knowledge from the massive intermediate labeled dataset for improving the same target text generation task. It has been shown that the additional training with general-purpose text corpora (*e.g.,* Wikipedia, WebText) on the same text generation task can improve the performance on a specific domain (*e.g.,* Movie) [42, 134]. For example, Fabbri *et al.* [42] performed summarization on intermediate pseudo-summaries created from Wikipedia to improve the zero-shot and few-shot performance of abstractive summarization, and Mao *et al.* [134] conducted generation on intermediate BookCorpus dataset (built from WebText) to improve commonsense story generation on the target WritingPrompts dataset.

5.1.3 Multi-Task Fine-Tuning. Multi-task fine-tuning can exploit cross-task knowledge to improve the primary text generation task by incorporating auxiliary tasks. Furthermore, by obtaining knowledge from related NLP tasks, multi-task fine-tuning can enhance the robustness of PLMs and reduce the need for large amounts of labeled instances in the text generation task. According to the similarity between the primary text generation task and auxiliary tasks, multi-task fine-tuning (MTFT) can be divided into two categories, *i.e.,* pure MTFT and hybrid MTFT.

Pure Multi-Task Fine-Tuning. Pure MTFT incorporates auxiliary tasks that are the same as the primary text generation task but from different domains. Previous studies mainly utilized additional datasets to eliminate the data scarcity issue of the primary text generation task [3, 61]. Specifically, Goodwin *et al.* [61] leveraged twenty-one additional summarization datasets to improve zero-shot summarization on previously unseen datasets. Besides, Bai *et al.* [3] incorporated an auxiliary monolingual summarization task to improve the primary cross-lingual summarization task in a low-resource language.

Hybrid Multi-Task Fine-Tuning. Hybrid MTFT incorporates auxiliary tasks that are different from the primary text generation task. These diverse auxiliary tasks can enhance the primary generation task in different aspects. For example, Liu *et al.* [118] and Jin *et al.* [86] fine-tuned PLMs with auxiliary tasks (*e.g.*, coherence detection, style-carrying text reconstruction) to control the content of the generated text according to the topic change and text style (humor, romance, and clickbait). Besides, to improve the faithfulness of the generated text, Li *et al.* [105] and Gong *et al.* [60] introduced auxiliary input reconstruction tasks to reconstruct KG triples and table values for aligning the input information with the generated content.

5.1.4 Parameter-Efficient Fine-Tuning. As the above fine-tuning methods require updating all PLM parameters, it is time-consuming to perform the entire fine-tuning in resource-limited scenarios. Many studies developed parameter-efficient fine-tuning (PEFT) for text generation tasks.

Adapter-based Parameter-Efficient Fine-Tuning. Adapter is a special neural layer proposed by Houlsby *et al.* [77] to fine-tune PLMs in a parameter-efficient way. The adapter module projects the input vector into a small vector and then projects back into the original dimension using two feed-forward layers and a non-linear layer. Specifically, the adapters first project the original d-dimensional features into a smaller dimension, m, apply a non-linearity, then project back to d dimensions. The total number of parameters added per layer, including biases, is $2md + d + m$. By setting $m \ll d$, we can limit the number of additional parameters per task. Thus, it is highly efficient to fix the parameters of original PLMs but only fine-tune the adapters [27, 170]. To address the inefficiency and overfitting issues in low-resource abstractive summarization, Chen *et al.* [27] inserted adapters into both encoder and decoder of PLMs and only fine-tuned the adapters. A number of studies have shown that adapters can help PLMs efficiently capture some input characteristics for generating more accurate output text with a low extra cost in terms of parameters [95, 159]. For example, Ribeiro *et al.* [159] utilized adapters to model the input graph structure effectively when fine-tuning PLMs on graph input.

Freezing-based Parameter-Efficient Fine-Tuning. This approach refers to freezing most parameters and only updating a small proportion of PLM parameters. Recent studies have shown that not all parameters of PLMs are necessary to be fine-tuned for text generation tasks, and some of them can be fixed during fine-tuning without large impact on the model performance. Several studies also revealed that cross-attention (or encoder-decoder attention) layers are more important than self-attention layers when fine-tuning PLMs for machine translation [55, 203]. Therefore, Gheini *et al.* [55] only fine-tuned cross-attention layers while kept the encoder and decoder fixed. This approach achieved comparable translation performance to fine-tuning all parameters.

Distillation-based Parameter-Efficient Fine-Tuning. Another parameter-efficient fine-tuning approach is to distill large teacher PLMs into small student models. By distilling the knowledge in PLMs for text generation into small generative models (*e.g.*, LSTM), the student models can be efficiently fine-tuned for better generation performance [26, 167]. As a representative example, Chen *et al.* [26] leveraged BERT as the teacher model that generates sequences of word probability logits and treated the Seq2Seq model as the student network, which can effectively learn from the teacher's outputs.

5.2 Prompt-Tuning for Text Generation

Most generative PLMs are pre-trained using language modeling objectives and then fine-tuned on text generation tasks with task-specific objectives. Such a discrepancy between pre-training and fine-tuning affects the performance of PLMs on text generation tasks. As a new learning paradigm, prompt learning [119] reformulates the downstream tasks (text generation tasks) into the language modeling task in pre-training.

5.2.1 Background. According to Liu *et al.* [119], a prompt function $f_{prompt}(\cdot)$ converts the input text x into a prompt $x' = f_{prompt}(x)$ through a two-step process:

1. Apply a textual *template* containing two slots: an input slot $[X]$ for input x and an answer slot $[Z]$ for an intermediate generated answer text z that will later be mapped into y.
2. Fill the input slot $[X]$ with the input text x.

Here the prompt can be *cloze* or *prefix* style. The cloze-style prompt is usually adopted in language understanding tasks, where the empty slot $[Z]$ is either in the middle of the prompt or at the end. For example, in sentiment analysis where $x =$"*I love this movie*", the template may take a clozed form such as "$[X]$ It was a really $[Z]$ movie." to predict the answer in $[Z]$. While in the prefix-style prompt, the input text comes entirely before the empty slot $[Z]$ such as "English: $[X]$ German: $[Z]$" in machine translation. Prefix prompts are widely used in text generation, as they mesh well with the left-to-right nature of language modeling. In the above prompt examples, the template is composed of *discrete* natural language tokens, but the tokens can also be virtual words (*e.g.,* represented by numeric IDs), which would be mapped into *continuous* embeddings later.

5.2.2 Discrete Prompts. Early prompting studies create prompts by manually designing templates based on human introspection. As a pioneering study, GPT-2 [153] performed text generation tasks using various manually-created prompts. For example, the prompt "translate to french, [*input*], [*output*]" is used in machine translation. The prompt defines the semantic mapping from input data to output text in a specific text generation task. By utilizing diverse prompts, a single PLM is able to perform a number of different text generation tasks. These approaches heavily relied on manual efforts to create prompts; but PLMs are highly sensitive to prompts: improperly-created prompts lead to low performance [83]. To avoid the need to manually specify prompts, Shin *et al.* [166] proposed AutoPrompt to automatically search for template tokens. Several other methods have also been proposed to discover discrete prompts automatically such as paraphrasing existing prompts [83], generating prompts using PLMs [51], and mining prompts from a corpus [83].

5.2.3 Continuous Prompts. Continuous prompts (*a.k.a.,* soft prompts), consisting of embedding vectors, are widely explored for text generation tasks. Two major advantages are expected: 1) relaxing the constraint that the prompt template should be natural language words; 2) removing the restriction that the template is parameterized by PLMs' parameters. Instead, continuous prompts have their own parameters that can be optimized based on training data of the text generation tasks. The most well-known method using continuous prompts for text generation is prefix-tuning [110], which freezes the generative PLMs (*e.g.,* GPT-2, BART) and optimizes a sequence of task-specific vectors (called *prefix*). In contrast to full-parameter fine-tuning, which requires storing a tuned copy of the model for each text generation task, prefix-tuning only optimizes the prefix for each text generation task. Similar to prefix-tuning, several studies used continuous prompts to solve other text generation tasks such as dialogue generation [65].

5.3 Property-Tuning for Text Generation

For different generation tasks, we need to consider specific language properties when tuning PLMs. In this section, we discuss three major properties that are widely desired for text generation.

5.3.1 Relevance. According to the linguistic literature [107], in text generation, *relevance* means that the topical semantics conveyed in output text is highly related to the input text. As a representative example, in dialogue systems, the generated responses should be relevant to the historical utterances and other conditions, such as speaker persona and discourse topic.

Compared with traditional neural generative models, PLMs utilize more powerful multi-layer cross-attention mechanism to model the semantic associations between input and output, which can enhance the relevance of generated text to the input data (*e.g.*, the dialogue systems [189, 215]). A good example is DialoGPT [215] based on an auto-regressive language model GPT-2. Specially, DialoGPT was first trained on large-scale dialogue pairs/sessions, which could enable DialoGPT to capture the joint distribution of $\Pr(history, response)$ in conversational flow for generating relevant responses to the history utterance. Furthermore, Zeng *et al.* [208] utilized the masked language modeling objective to solve generate responses based on various types of dialogue context. Specifically, they proposed a TF-IDF based masking which selects more condition-related tokens to be masked, so that PLMs can generate condition-related expressions rather than the general language patterns. Besides, they adopted a non-parametric attention-based gating mechanism to switch between generating a general word or a condition-related word at each position.

5.3.2 Faithfulness. Faithfulness is also an important language property to consider for text generation, which means the generated content should adhere to the semantics of input text. For example, text summarization aims to generate faithful text conveying the salient information of the input text. Faithfulness sometimes refers to the fact that the generated text is in accord with world facts.

To generate faithful texts, PLMs should be able to accurately understand the core semantics of input and acquire sufficient world knowledge for solving the downstream task. It has been shown that PLMs have excellent natural language understanding capacities in capturing core semantics from plain text [35], and they indeed encode a large amount of world knowledge [83], which is potentially beneficial to generate faithful summary by injecting background knowledge into text. For example, Kryscinski *et al.* [93] utilized a contextual network in the PLM decoder to retrieve the most salient parts from the source document to improve the level of faithfulness of generated summaries. Besides, several studies proposed to generate faithful texts by introducing additional losses besides the text generation loss [161, 202]. Specifically, Yang *et al.* [202] fine-tuned PLMs through a theme modeling loss which aims to make the generated summary semantically close to the original article for achieving faithful generation.

5.3.3 Order-Preservation. In the NLP field, *order-preservation* is a special property that refers that the order of semantic units (word, phrase, etc.) in both input and output text is consistent. Such a property is key to several important text generation tasks, such as text paraphrasing and machine translation. In machine translation, when translating from source language to target language, it often requires preserving some order of phrases in the source and target text for ensuring the accuracy of the translation results.

In machine translation, word alignment is an extensively studied approach to achieve the order-preservation property. A representative study is Code-Switching Pre-training (CSP) [199]. CSP first automatically extracted the word-pair alignment information from the source and target monolingual corpora. Then, to enhance the order-preservation property during translation, CSP continually pre-trained PLMs by predicting the sentence fragment on the source side given the aligned fragment in the target language. Moreover, to relax the restriction of discrete word alignment, another line of research aims to conduct continuous representation alignment to improve the order-preservation property. Wada *et al.* [182] focused on aligning word representations of each language by mapping word embeddings of each language into a common latent space. Lin *et al.* [116] proposed mRASP to enforce words and phrases that have similar meanings across multiple languages, to be aligned in the representation space.

Table 1. Summary of major challenges in the three aspects and existing PLM-based solutions

Aspect	Challenge	Solution
Data Aspect	Lacking Enough Training Data	prior knowledge transfer [120, 144, 227], data augmentation [23, 131, 142, 193], multi-task learning [3, 61]
	Bias in Pretraining Corpora	Mitigate the gender bias in word embeddings [9], identify and mask bias-sensitive tokens [34].
Model Aspect	Model Compression	Quantization by truncating PLMs weights [171, 204], pruning less critical weights [44, 62, 68, 76], knowledge distillation [26, 85, 103].
	Model Enhancement	Large-scale PLMs [14, 46, 98, 207], knowledge-enriched PLMs [71, 111, 145, 217], efficient PLMs [74, 84].
Optim. Aspect	Satisfying Text Properties	Enhance coherence [107, 173], preserve factuality [29, 37, 105, 135], improve controllable [33, 91, 141].
	Mitigating Tuning Instabilities	Intermediate fine-tuning [126, 146], mixout strategy [97], supervised contrastive learning [67].

6 CHALLENGES AND SOLUTIONS

The three previous sections described three key aspects together with the basic methods used in PLM-based text generation. In this section, we further discuss the major challenges in each of the aspects and possible solutions. A summary of these challenges and solutions is presented in Table 1.

6.1 Data Aspect

We first discuss the challenges and solutions related to the data aspect.

6.1.1 Lacking Sufficient Training Data. In a number of text generation tasks, it is difficult to obtain sufficient annotated data. Transfer learning provides an effective solution by transferring the knowledge of data-rich source tasks into data-scarce target text generation tasks. Besides, data augmentation and multi-task learning can be also used to address this problem.

Transfer Learning. To deal with the data scarcity issue, several studies proposed first fine-tuning PLMs on large amounts of external labeled corpora and then transferring into data-scarce target text generation tasks [120, 144, 227]. In particular, Peng *et al.* [144] and Zou *et al.* [227] first fine-tuned PLMs on substantial labeled dialog/summary data and then fine-tuned for the target dialog/summarization task in a new domain with limited labeled data. Similarly, Liu *et al.* [120] first trained models on large-scale ungrounded dialogs and unstructured knowledge base separately to improve the low-resource knowledge-grounded dialog generation task.

Data Augmentation. In recent literature, data augmentation has emerged as a critical method for increasing the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. One line of research is to use retrieval models to obtain real data from external corpora as the augmented data [142, 193]. For the query-focused summarization task, Pasunuru *et al.* [142] used a search engine, *i.e.,* Bing, to retrieve the answer paragraph as the synthetic summary and used the top ranked documents as input text. Another line of work is to use perturbation-based methods by corrupting the original text [23, 131]. For example, Chen *et al.* [23] presented a set of data augmentation methods for conversation summarization, such as random swapping/deletion to randomly swap or delete utterances in conversations.

Multi-Task Learning. Leveraging other data-rich tasks and datasets can also overcome the data scarcity issue. Most studies usually incorporated similar auxiliary generation tasks for enhancing the primary text generation task [61]. However, these methods usually adopt independent decoders for each task, thus breaking the semantic connections between high- and low-resource text generation tasks. To bridge this gap, Bai *et al.* [3] employed a unified decoder which learns the alignments and patterns across multiple languages in machine translation.

6.1.2 Data Bias from Pre-training Corpora. In sociology, bias is an unjustified prejudice in favour of or against a person, group, or thing [54]. PLMs are generally trained using real-world data in such a way that they model the statistical properties of the training data. As a result, they inherit the biases and stereotypes that are common in the data [54]. These biases and stereotypes can pose significant challenges in downstream text generation tasks [14].

It has been shown that the generated texts from PLMs are likely to be biased towards some attributes [14], *i.e.,* favoring a particular race, gender or aged people, which is not desired for the text generation tasks. These undesirable biases are unexpectedly hidden in model components such as word embeddings [10] and attention heads [181]. A simple approach to mitigating the gender bias in word embeddings is to "swap" gendered terms in training data when generating word embeddings [221]. Furthermore, simply masking names and pronouns may also reduce biases and improve the performance of certain language tasks [34]. However, to date, there is still no general, unified approach to reducing the data bias from PLMs for text generation. Some of these techniques for bias detection and mitigation have been critiqued as merely capturing over-simplified dimensions of bias with proper debiasing requiring more holistic evaluation [59].

6.2 Model Aspect

In this section, we present the challenges from the architecture design, and discuss corresponding solutions for text generation.

6.2.1 Model Compression. Although PLMs have achieved great success on text generation, the backbone Transformers are still bulky and resource-hungry, resulting in high memory consumption, computational overhead, and energy cost. To address these issues, more and more approaches are proposed to compress PLMs [50], such as quantization, pruning, and knowledge distillation.

Quantization. Quantization means reducing the number of unique values used to represent PLMs weights, which in turn allows to represent them using fewer bits [50]. As most PLMs are built upon Transformer, quantization can be generally applied to those weights residing in fully-connected layers (*i.e.,* embedding layers, linear layers, and feed-forward network layers). However, when the model parameters are compressed, the generation capacity might be reduced. To alleviate the issue of generating unsatisfactory text with truncated PLMs, a promising solution is to first identify important weights and then avoid truncating them during the quantization step [204].

Pruning. Pruning refers to identifying and removing redundant and/or less important weights [50]. Pruning methods for text generation largely fall into two categories [50]. The first type of unstructured pruning prunes individual weights by locating the set of least important weights in PLMs. The importance of weights can be measured by specific metrics such as absolute values [62] and gradients [68]. The second type of structured pruning prunes structured blocks of weights or even complete components of PLMs by reducing and simplifying certain modules such as attention heads [76] and Transformer layers [44].

Knowledge Distillation. Knowledge distillation refers to training a smaller model (called the *student*) using the output of PLMs (called the *teacher*). First, the student model can directly learn from the output word distribution of the final softmax layer in PLMs, which allows the student to mimic the generated text of the teacher by replicating the word distribution across the whole vocabulary [26]. Second, the student can also learn from the output tensors of PLMs encoders [103]. Intuitively, the representations of PLMs encoder may contain meaningful semantics and contextual relationships between input tokens, which is helpful for generating accurate text.

Third, by replicating attention distributions between input data and output text, the student can also learn the contextual dependency between input and output [85].

6.2.2 Model Enhancement. Although PLMs have achieved great success nowadays, they are still far from our expectations. Recently, there has been a surge of interest in the research community to strengthen existing PLMs to improve the performance of text generation.

Large-scale PLMs. Kaplan *et al.* [89] have shown that the performance of PLMs can be boosted by scaling up the amount of PLMs' parameters. This observation sparked the development of large-scale PLMs in text generation [14, 207]. The most representative large-scale PLMs for text generation is GPT-3 [14], which contains 175 billion parameters, 10x more than any previous non-sparse PLMs. With a large number of parameters, GPT-3 can achieve strong performance in various text generation tasks without any gradient updates or fine-tuning.

Knowledge-Enriched PLMs. Recent research has found that integrating knowledge from external knowledge sources can enhance the text generation performance of PLMs [175, 225]. Specifically, ERNIE 3.0 [175] was pretrained on a 4TB corpus consisting of plain texts and a large-scale knowledge graph for both language understanding and generation tasks. Without incorporating explicit knowledge, CALM [225] can encode commonsense knowledge into parameters by teaching PLMs to write and reason with common concepts through pre-training strategies, yielding better performance on text generation tasks.

Efficient PLMs. Pre-training PLMs on large-scale text data is prohibitively expensive. Recently, it has been demonstrated that by meticulously structuring the model architecture, it is possible to obtain equivalent or higher text generation performance with less pre-training data [225] or lower pre-training costs [84]. For example, CALM [225] developed a mutually reinforced pre-training framework with generative and contrastive objectives, thus achieving comparable results to other larger PLMs such as T5 while only being pre-trained on a small corpus for a few steps.

6.3 Optimization Aspect

In this part, we discuss challenges and solutions about the optimization of PLMs for text generation.

6.3.1 Satisfying Special Text Properties. In Section 5.3, we introduced three basic text properties. In this section, we will present three more difficult properties for text generation tasks, *i.e.,* coherence, factuality, and controllability.

Coherence. In linguistics [101], language coherence is what makes a multi-sentence text meaningful, both logically and syntactically. An essential technique to improving coherence is to elaborately plan the generated content, which is known as text planning [78, 107]. For example, Li *et al.* [107] designed a text generation model based on a two-level text plan: (1) the document plan is modeled as a sequence of sentence plans in order, and (2) the sentence plan is modeled as an entity-based subgraph from KG. The local coherence is naturally enforced by KG subgraphs, and the global coherence can be improved by generating a coherent sequence of subgraphs. Wang *et al.* [186] proposed a two-stage planning, *i.e.,* the first stage is to organize the story outline which illustrates the story plots and events, and the second stage is to expand the outline into a complete story.

Factuality. The input data (*e.g.,* infobox) for text generation tasks (*e.g.,* table-to-text generation) usually contains some factual information. In such cases, the generated content should adhere to the original input facts. However, lacking direct access to the input facts or explicit supervision makes PLMs unable to retain text factuality in generation process. For data-to-text generation, the pointer generator [164] is usually adopted to copy the input facts into output for preserving factuality [29, 105]. Furthermore, to make summarization models produce more factual summaries, some studies proposed evaluation metrics or correction methods to measure and revise the generated text for preserving factuality [37, 135].

Controllability. In text generation, many applications need a good control over the output text. For example, to generate reading materials for kids, we would like to guide the output stories to be safe, educational and easily understandable by children. The Plug and Play Language Model, also known as PPLM [33], is an example of a controllable PLM that combines a PLM with one or more simple attribute classifiers that direct text generation without further PLM training. Several studies achieved controllablility from a distributional view [91, 141]. Pascual *et al.* [141] described a plug-and-play decoding approach in a single sentence: given a topic or keyword, the model adds a shift to the probability distribution over the vocabulary towards semantically similar words.

6.3.2 Mitigating Tuning Instabilities. Due to the catastrophic forgetting nature of PLMs and small size of text generation datasets, tuning PLMs for text generation is usually unstable *i.e.,* fine-tuning the model with different random seeds results in a wide variance of performance. The possible solutions include intermediate fine-tuning, mixout and using supervised contrastive loss.

Intermediate Fine-Tuning. Recent studies have shown that first training PLMs on data-rich intermediate labeled datasets (*e.g.,* a similar NLP task from the same target domain) before fine-tuning them on data-scarce target text generation tasks can achieve better performance in target tasks [126, 146]. For example, Liu *et al.* [126] constructed an intermediate monolingual corpus of the target language (*e.g.,* Kazakh) and fine-tuned mBART to reconstruct the corrupted monolingual text for improving the translation quality of the low-resource target language.

Mixout Strategy. When fine-tuning PLMs, dropout [169] has been used as a regularization method to prevent performance degeneration if there are only a small number of training instances. Lee *et al.* [97] introduced a variant of dropout, mixout, which stochastically mixes parameters of two PLMs. The mixout strategy can regularize learning by minimizing the deviation from one of the two PLMs and the strength of regularization adapts along the optimization trajectory.

Contrastive Learning. The most used cross-entropy loss in text generation, *i.e.,* the KL-divergence between one-hot vectors of labels and the distribution of model's outputs, lacks robustness to noise labels [219] or adversarial examples [41]. Thus, fine-tuning PLMs with cross-entropy loss tends to be unstable, especially when labeled data is limited. An effective solution is to capture the similarity between examples in one class and contrast them with examples in other classes [67]. To this end, Gunel *et al.* [67] combined the cross-entropy loss with a supervised contrastive learning loss that pushes the words from the same class close and the words from different classes further apart.

7 EVALUATION AND RESOURCES

In this section, we will discuss several commonly used evaluation metrics and resources with respect to PLMs for text generation.

7.1 Evaluation

With the growing variety of text generation applications and datasets, there are several advantages of automatic evaluation: it is potentially much cheaper and quicker than human evaluation, and it is repeatable [8]. Therefore, we mainly concentrate on automatic evaluation metrics for text generation in this part. Following Celikyilmaz *et al.* [19], we present four categories of metrics, *i.e.,* n-gram overlap metrics, diversity metrics, semantic similarity metrics, and logit-based metrics. We list the metrics used in each text generation task in Table 2.

7.1.1 N-Gram Overlap Metrics. These metrics measure the degree of word "matching" between machine-generated and ground-truth texts at the word level.

BLEU. The Bilingual Evaluation Understudy (BLEU) [140] is one of the first metrics used to compare the similarity of two sentences. This metric was originally proposed for machine translation by comparing a candidate translation of text with one or more reference translations and now applied in various generation tasks. BLEU-n measures the precision of the co-occurrences of n-grams between the generated and real text and conducts length penalty on shorter generated text. Specially, SacreBLEU [149] is recommended for use in machine translation to avoid inconsistency issue. Several smoothing methods [20] are also proposed to evaluate short sentences.

ROUGE. Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [115] is a set of metrics for measuring automatic summarization of long texts consisting of multiple sentences. ROUGE-n counts the F1 score of the overlapping n-grams between generated and ground-truth texts.

METEOR. The Metric for Evaluation of Translation with Explicit ORdering (METEOR) [4] is proposed to address some issues found in BLEU. Compared to BLEU, METEOR is computed based on the harmonic mean of the unigram precision and recall, and measures word-to-word matches between generated and real text based on WordNet.

ChrF++. Character n-gram F-score (ChrF++) [148] is an automatic evaluation metric for machine translation. Different from the word level co-occurrence of BLEU, ChrF++ is mainly focused on the character-level matching so as to consider morpheme overlapping.

7.1.2 Diversity Metrics. Lexical diversity is desirable in many text generation tasks, such as dialogue systems and story generation. For these tasks, it is necessary to conduct diversity evaluation on generated texts.

Distinct. Distinct-n measures the degree of diversity by calculating the number of distinct n-grams in generated text [100]. This metric is scaled by total number of generated tokens to avoid favoring long sentences.

7.1.3 Semantic Similarity Metrics. The above metrics are focused on the literal word comparison. Many studies also proposed to compare the implicit semantics between generated text and ground-truth text. A typical approach is to map both generated text and ground-truth text into sentence vectors and then compare their embedding similarity.

BERTScore. Given the excellent performance of BERT across many tasks, BERTScore [213] leverages the pre-trained contextual embeddings from BERT and compares words in candidate and reference texts by cosine similarity. BERTScore has proven to correspond well with human judgments on sentence-level and system-level evaluations [19].

*7.1.4 **Logit-Based Metrics**.* In text generation, the probability of a generated text $y = \langle y_1, \dots, y_n \rangle$ can be formulated as $\Pr(y) = \prod_{j=1}^{n} \Pr(y_j | y_{1:j-1}; x)$, where x denotes the input data, and $y_{1:j-1}$ denotes the previous tokens $\langle y_1, \dots, y_{j-1} \rangle$. Logit-based metrics evaluate the generated text from a probabilistic view.

PPL. In information theory, perplexity (PPL) is a measurement of how well a probability distribution or probability model predicts a sample compared with the ground-truth [12]. A low perplexity indicates the probability distribution is good at predicting the sample. Therefore, the perplexity of the discrete probability distribution $\Pr(\cdot)$ is defined as:

$$\mathrm{PPL}(\Pr(y)) := e^{\mathrm{H}(\Pr(y))} = e^{-\sum_y \Pr(y) \ln \Pr(y)} = \prod_y \Pr(y)^{-\Pr(y)}, \tag{3}$$

where $\mathrm{H}(\Pr(y))$ is the entropy of the distribution $\Pr(\cdot)$.

7.2 Resources

In this section, we will introduce some available open-source libraries and benchmarks.

7.2.1 Open-Source Libraries. There are a number of public text generation libraries that can be used to implement PLM-based text generation models. Transformers [188] is an all-featured library for Transformer-based PLMs, and Fairseq [137] is a library to train custom models for translation,

Table 2. A summary of common datasets and metrics used in each generation task. †BLEU with smoothing method 7 (with NLTK version 3.4) is usually employed in open-domain dialogue system [6]. ‡Inform (rate) and Success (rate) are two accuracy metrics specially designed for task-oriented dialogue system [15].

Tasks	Sub-Tasks	Datasets	Metrics
Machine Translation	Unsupervised MT	WMT'14 English-French [31], WMT'16 German-English [31]	SacreBLEU
	Supervised MT		
Summarization	Vanilla Summarization	CNN/DailyMail [168], XSum [168], GigaWord [168]	ROUGE, BERTScore
	Dialogue Summarization	SAMSum [22]	ROUGE
Dialogue System	Open-Domain Dialogue System	PersonaChat [6], DailyDialogue [6], DSTC7-AVSD [6]	Perplexity BLEU†, Distinct
	Task-Oriented Dialogue System	MultiWOZ [15]	BLEU, Inform‡, Success‡
Question Generation		SQuAD [36]	BLEU, ROUGE, METEOR
Story Generation		ROCStories [66], WritingPrompts [155]	Perplexity BLEU, Distinct
Data-to-text Generation		AGENDA [158], LDC2017T10 [130], WikiBio [29], WebNLG [158], E2E [25]	BLEU, ROUGE, METEOR, chrF++

summarization, language modeling and other text generation tasks. Besides, some of libraries like FastSeq [196], DeepSpeed [156], and LightSeq [187] are useful to increase the inference speed of models. TextBox [104] supports 21 text generation models, including several prevalent PLMs, and diverse generation strategies (*e.g.,* top-k, beam search) and evaluation metrics (*e.g.,* BLEU, Distinct). One can easily choose different PLMs, optimization methods, and evaluation metrics by setting corresponding hyper-parameters with just a few lines of code.

7.2.2 Evaluation Benchmarks. In order to evaluate the comprehensive capacities of PLMs, several important evaluation benchmarks are created and released, which involve multiple evaluation tasks from different aspects. In addition to GLUE [184] and SuperGLUE [183] which are general language understanding evaluation benchmarks, an increasing number of general benchmarks targeted for text generation have recently been proposed. Liu *et al.* [117] introduced the General Language Generation Evaluation (GLGE) benchmark, a new multi-task benchmark for evaluating the generalization capabilities of text generation. GLGE contains 8 English language generation tasks, covering summarization, question generation, generative question answering, and dialogue. For each task, GLGE designs three sub-tasks in terms of task difficulty (*i.e.,* GLGE-Easy, GLGE-Medium, and GLGE-Hard).

8 APPLICATION

As discussed in Section 2, text generation can be instantiated into different kinds of applications. To summarize existing text generation applications, we present an overview of different tasks (as well as corresponding common datasets and metrics) in Table 2. In what follows, we will highlight three classic applications, *i.e.,* machine translation, text summarization and dialogue system, and briefly discuss how to design a task-specific PLM to adapt to specific text generation tasks.

8.1 Machine Translation

Machine translation (MT) is the process of automatically translating one language into another. With the advent of deep learning, Neural Machine Translation (NMT) has emerged as the dominant method in both academic research and commercial use [32]. Machine translation can be classified into two types: unsupervised machine translation and supervised machine translation, depending on whether parallel corpora are available for fine-tuning PLMs.

8.1.1 Unsupervised Machine Translation. Unsupervised Machine Translation (UMT) refers to the use of solely monolingual corpora without any parallel data for both pre-training and fine-tuning PLMs. UMT enables machine translation to no longer rely on large-scale annotated corpora, and also brings remarkable advances in low-resource language translation. When using PLMs for UMT, there are typically two steps involved [94]: 1) PLMs are pre-trained on monolingual corpora in a variety of languages, learning word embeddings and modeling probabilities for each sentence in each language; 2) Iterative back-translation is then leveraged to combine the source-to-target and target-to-source model with the denoising auto-encoding and back-translation objectives.

Pre-training on Monolingual Corpora. Recent PLM-based research has mainly focused on the first step of UMT. Specifically, XLM [31] and mBERT [35] were pre-trained on multiple monolingual data using MLM task, and then the PLM was used to initialize both the encoder and the decoder for machine translation. mBART [122] followed the pre-training scheme of BART [99] on multiple languages, while these PLMs just performed the original pre-training task with mixed monolingual corpora, without considering the relationship between languages. CMLM [157] further proposed cross-lingual MLM to randomly mask tokens in monolingual sentences and predicted corresponding translation candidates. Therefore, CMLM was able to align the embeddings of different languages. CSP [199] shared the similar idea, replacing some words in the source sentences with their translation words and then predicting the replaced words.

Leveraging Iterative Back-translation. In the back-translation stage, Garcia *et al.* [53] proposed using multi-task learning. They investigated *multilingual UNMT*, which involved the use of a third language when translating one language into another. The extra language can provide auxiliary monolingual data or parallel data containing only one language in the source or target language. They aggregated back-translation loss and introduced a cross-translation term to incorporate the auxiliary corpus. Li *et al.* [113] also applied the cross-translation term and additionally included a knowledge distillation objective for the third (intermediate) language.

8.1.2 Supervised Machine Translation. Supervised machine translation (SMT) refers to fine-tuning PLMs based on parallel corpora. Here, we will discuss how to utilize existing self-supervised PLMs and how to design PLMs for parallel corpora.

Directly Fine-tuning Unsupervised PLMs. Almost all PLMs mentioned above using unsupervised (self-supervised) pre-training, such as XLM [31] and mBART [122], can be directly fine-tuned with bilingual pairs. Moreover, considering the excellent encoding capability of BERT, BERT-fused model [226] leveraged BERT to extract contextual embedding for the source sentence, and fused the representations with each layer of the encoder and decoder. CTNMT [197] leveraged asymptotic distillation and dynamic switching gate to integrate the BERT embedding. Graformer [176] grafted mBERT as the encoder and mGPT as the decoder, and then trained a cross-attention module to combine them. Tang *et al.* [178] proposed to fine-tune mBART on multiple language pairs, which is called *multilingual fine-tuning*.

Designing PLMs for Parallel Corpora. Most of PLMs are pre-trained on monolingual corpora using self-supervised pre-training tasks such as MLM and DAE. Nevertheless, these pre-training objectives are different from the downstream translation task. Hence, mRASP [116] pre-trained the model on bilingual pairs with supervised Seq2Seq loss by randomly replacing the words in the source sentence with the words which have the same meaning in other languages. As a result, words with similar meaning across different languages are encouraged to share similar representations. mRASP2 [139] applied contrastive learning to minimize the representation gap of similar sentences and maximize that of unrelated sentences. Despite significant success, pre-training on parallel data requires massive labour and financial resources to create vast amounts of bilingual pairs.

8.2 Text Summarization

Text summarization is the process of condensing text into a brief summary that retains key information from the source text [40]. The mainstream approaches to text summarization based on PLMs are either extractive or abstractive. Extractive summarization selects a subset of sentences from the source text and concatenates them to form the summary [123, 214]. In contrast, abstractive summarization generates the summary automatically from the abstract representation of input texts [164, 211]. As abstractive summarization is more related to text generation, we only discuss abstractive summarization in this section.

8.2.1 Document Summarization. Document is a widely-used literary form, such as news, opinions, reviews, and scientific papers. PLMs, such as UniLM [5, 36], MASS [168], T5 [154], BART [99] and PEGASUS [211], can be directly fine-tuned for document summarization. During pre-training, these models learn to predict the masked important sentences in the input document based on the remaining ones, which shares the similar idea of summarization.

Without directly generating summaries, several studies first extracted keywords, key sentences or relations as guidance and then combined these with PLMs for generation. CIT [162] employed RoBERTa [124] to extract the important words and sentences from the input document. In addition, topic models are used to capture the global topic semantics of the document, which can be integrated into the summarization model [136]. GSum [38] proposed a general framework taking different kinds of guidance signals into the generation model, including keywords, triples, highlighted sentences and retrieved summaries. Apart from external guidance, several tricks can be applied to document summarization. Cao *et al.* [16] improved the attention mechanism to emphasize salient content in the document. Refactor [121] first generated multiple summaries under different setups and then scored them and finally selected an optimal candidate summary.

8.2.2 Dialogue Summarization. Dialogues, such as chat and medical conversation, consist of multi-turn utterances by two or more individuals. Hence, it is critical to capture the semi-structured dialogue content and users' interactions in dialogue [47]. For dialogue summarization, it is straightforward to direclty reuse document summarization models. Zhang *et al.* [212] first truncated the dialogue text into several chunks, then summarized each chuck into partial summaries, and finally rewrote these partial summaries into a complete summary.

Meanwhile, several studies also explored some specific characteristics of dialogue for improving dialogue summarization. Chen *et al.* [22] first extracted different topic views from conversations, and then utilized a multi-view decoder to combine these views for generating summaries. Furthermore, Chen *et al.* [24] constructed discourse relation graphs and action graphs of conversations, in order to concentrate on the most salient utterances and understand concrete details of users' action. Considering the low information density, topic drifts and frequent coreferences of dialogue [47], some researchers conducted auxiliary tasks to extract intrinsic information of dialogue. Feng *et al.* [48] utilized DialoGPT [215], a PLM specially designed for dialogue, to automatically extract keywords, detect redundant utterances and divide a dialogue into topically coherent segments.

8.3 Dialogue System

Dialogue system (*a.k.a.,* conversational agent) aims to make machines communicate with human fluently. Technically, machines are required to generate a response conditioned on history contexts. According to downstream applications, dialogue systems are commonly categorized into open-domain and task-oriented dialogue systems. The former intends to converse with humans engaged on open topics such as daily life, sports and entertainment [80], while the latter is focused on assisting users to complete specific tasks, such as hotel reservation and product purchase [220].

8.3.1 Open-domain dialogue System. Open-domain dialogue system is also known as chat-bots focusing on daily chat. For example, Microsoft XiaoIce is a well-known open-domain dialogue system to satisfy human needs for communication, affection, and social belonging [224].

Continuous Pretraining with dialogue Corpora. PLMs, such as GPT-2, are pre-trained on general text corpora, thus various studies continually pre-trained general-purpose PLMs to fit dialogue systems. Due to the difficulty in obtaining large-scale dialogue corpora, informal text resources (such as forum posts and comments in Reddit, Twitter and Weibo) are usually employed for continual pre-training. As two typical models, DialoGPT [215] and Meena [1] used English or Chinese dialogue corpora to continually pre-train casual LMs like GPT-2. Besides, Blender [160] and PLATO [6] utilized the Seq2Seq loss to generate the next utterance based on previous utterances. Moreover, PLATO [6] incorporated the next utterance classification (NUC) loss, similar to the next sentence prediction task in BERT, to judge whether the response is relevant to history dialogues to enhance the coherence of utterances. In order to penalize bland responses and decrease repetitions, DialoGPT [215] employed mutual information maximization to predict the input given generated response and Blender [160] adopted unlikelihood training objective to penalize repetitive n-grams.

Directly Fine-tuning Existing PLMs. In addition to pre-training on dialogue corpora, researchers also explored fine-tuning existing PLMs on dialogue tasks. TransferTransfo [189] adapted GPT to the dialogue task through multi-task learning. Based on TransferTransfo, Golovanov *et al.* [58] modified the architecture to better model multiple inputs including dialogue history, persona information, and current state. Besides, to capture the hierarchical structure of dialogue, hierarchical encoders have been proposed to model the dialogue input [64, 112]. Gu *et al.* [64] proposed a hierarchical framework, dialogueBERT, that uses sentence- and discourse-level Transformer encoders to encode each dialogue utterance and the sequence of utterance vectors, respectively. Furthermore, controllability is also important to consider in dialogue systems. Zeng *et al.* [209] utilized condition-aware Transformer block to steer the response in a specific topic label. StyleDGPT [201] attempted to enforce the target style of the generated response with KL loss at both word and sentence levels.

8.3.2 Task-Oriented Dialogue System. Task-oriented (*a.k.a.,* goal-oriented) dialogue system is a widely-used text generation application in real life, such as helping users order tickets. Generally, task-oriented dialogue system was divided into four modules, *i.e.,* natural language understanding, dialogue state tracking, dialogue policy learning and natural language generation [220].

Most previous work only focused on the last generation module in task-oriented dialogue system by using generative PLMs (*e.g.,* GPT). For example, SC-GPT [144] used the ground-truth results of previous three modules (*e.g.,* dialogue state) and serialized them as input of the last generation module to generate response. Kale *et al.* [87] further designed a manual schema to better convert previous results into a natural language. Shalyminov *et al.* [165] proposed to generate and retrieve several responses based on the dialogue context and utilized the NUC task to select the best one. PRAL [63] utilized two separate GPT-2 to model the user and system, and adopted a third GPT-2 to perform knowledge distillation and incorporate commonsense knowledge into the final dialogue generation. Besides, more and more studies proposed to jointly learn these four modules based on a shared PLM. Budzianowski *et al.* [15] and Hosseini-Asl *et al.* [75] generated the dialogue state, system action and final response successively, based on the original dialogue history.

8.4 Others

In this part, we will briefly introduce other text generation tasks, such as question generation, story generation and data-to-text generation.

8.4.1 Question Generation. Question generation can be seen as a dual task of question answering (QA), *i.e.,* generate coherent questions based on given passages and answers. Existing PLMs, such as UniLM [5, 36] and ProphetNet [150], can be employed for this task by taking as input the concatenation of the passage and answer. Moreover, researchers explored this task in different QA settings. For example, Huang *et al.* [81] proposed a two-stage model to solve multi-hop question generation, and Cao *et al.* [17] attempted to generate open-ended questions which are answered by multiple sentences. Moreover, Majumder *et al.* [132] proposed a clarification question generation task to ask questions about the missing information in the passage in order to reduce the ambiguity.

8.4.2 Story Generation. Story (or narrative, news) generation requires to generate a long-form open-ended text leveraging the given title or premise. It is challenging to produce a coherent and informative text based on limited input [52]. To enrich the content of generated text, some studies aimed to incorporated external knowledge into PLMs. Guan *et al.* [66] and Mao *et al.* [134] utilized commonsense knowledge base to fine-tune PLMs to generate reasonable stories. Megatron-Cntrl [192] used extracted keywords to retrieve knowledge sentences and then selected top-ranked sentences for story generation. Besides, to generate coherent long-form text, PlotMachines *et al.* [155] extracted keywords from input as outline to organize the output structure; Guan *et al.* [66] leveraged the contrastive learning loss to judge whether two sentences are consecutive in original text.

8.4.3 Data-to-text Generation. The above tasks take unstructured text as input, while the data-to-text generation task generates descriptive text about structured input data, such as table, knowledge graph (KG) and abstract meaning representation (AMR). First, a naive and straightforward approach is to directly linearize the structured table [29, 60] and KG [72, 158] into textual form as the input of PLMs. Considering the graph structure of KG and AMR, Li *et al.* [105] and Ribeiro *et al.* [159] employed graph neural network to learn a better representation for each node. Moreover, to cope with the structural information, a typical approach is to incorporate auxiliary training objectives such as predicting the value of table [60] and the relation of knowledge graph [105].

8.4.4 Other Generation Tasks. Besides the aforementioned tasks, there are also other text generation applications. ColdGANs [163] explored the unconditional language generation. KG-BART [125] investigates the commonsense generation, *i.e.,* generating a natural language consisting of provided commonsense concept (word), which can be considered as the hard-constrained conditional generation [52]. Moreover, text style transfer aims to convert a text into another style while preserving the basic semantics of input [52], such as sentiment transfer and writing style transfer [92]. In addition, some researchers devoted to literary creation, such as poem [109] and lyric [195].

9 CONCLUSION AND FUTURE DIRECTIONS

In this survey, we presented an overview of current representative research efforts on PLMs-based text generation, and expect it can facilitate future research. We began with introducing three key aspects when applying PLMs to text generation, based on which the main content of our survey is divided into three sections from the view of input representation learning, model architecture design, and parameter optimization. Besides, we discussed several non-trivial challenges related to the above three aspects. Finally, we reviewed various evaluation metrics, open-source libraries, and common applications to help practitioners evaluate, choose and employ PLMs for text generation.

Despite the great progress made in recent years, we are faced with several open problems and several future directions are promising to deal with them.

Controllable Generation. Controllable text generation with PLMs is an interesting direction but still at a very early stage. Controlling some attributes of the generated text has many practical use cases, such as generating positive responses to patients suffering from depression in dialogue systems. However, PLMs are usually pre-trained in universal corpora, which is difficult to control the multi-grained attributes of the generated text (*e.g.*, sentiment, topic, and coherence). Keskar et al. [90] has explored text generation with control codes that govern style, content and task-specific behavior. However, these control codes are preset and coarse-grained. Future work can explore multi-grained control and develop PLMs that are sufficiently steerable.

Optimization Exploration. Fine-tuning is the predominant optimization way to distill the linguistic knowledge stored in PLMs to downstream generation tasks. Now, prompt-based learning has become a performant and lightweight optimization method [119]. Future work can explore a broader range of optimization approaches that can combine the advantages of current methods.

Language-agnostic PLMs. Nowadays, almost all the PLMs for text generation are mainly for English. These PLMs will encounter challenges when dealing with non-English generation tasks. Therefore, language-agnostic PLMs are worthy to be investigated. This requires us to capture universal linguistic and semantic features across different languages. An interesting direction is explore how to reuse existing English-based PLMs for text generation in non-English languages.

Ethical Concern. Currently, PLMs are pre-trained on large-scale corpora crawled from web without fine-grained filtering, potentially causing ethical issues such as generating private content about users. Therefore, researchers should try their best to prevent misusing PLMs. Besides, the text generated by PLMs might be prejudiced, which is in line with the bias in training data along the dimensions of gender, race, and religion [14]. As a result, we should intervene PLMs for preventing such biases. The research on the general approach is extensive but still preliminary for PLMs.

In conclusion, text generation based on PLMs has greatly contributed to the advance of the state of the art in this field. However, the current state of the art in different text generation tasks is still far from what one could expect. Extensive research efforts are needed to better adapt PLMs to text generation tasks.

REFERENCES

[1] Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a Human-like Open-Domain Chatbot. *arXiv preprint arXiv:2001.09977* (2020).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[3] Yu Bai, Yang Gao, and Heyan Huang. 2021. Cross-Lingual Abstractive Summarization with Limited Parallel Resources. In *ACL*.

[4] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *IEEvaluation@ACL*.

[5] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon. 2020. UniLMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training. In *ICML*.

[6] Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. PLATO: Pre-trained Dialogue Generation Model with Discrete Latent Variable. In *ACL*.

[7] Siqi Bao, Huang He, Fan Wang, Hua Wu, Haifeng Wang, Wenquan Wu, Zhen Guo, Zhibin Liu, and Xinchao Xu. 2021. PLATO-2: Towards Building an Open-Domain Chatbot via Curriculum Learning. In *ACL/IJCNLP Findings*.

[8] Anja Belz and Ehud Reiter. 2006. Comparing Automatic and Human Evaluation of NLG Systems. In *EACL*.

[9] Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H. Chi. 2017. Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations. *arXiv preprint arXiv:1707.00075* (2017).

[10] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. 2016. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In *NIPS*.

[11] Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A Statistical Approach to Machine Translation. *Comput. Linguistics* 16, 2 (1990).

[12] Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. 1992. An Estimate of an Upper Bound for the Entropy of English. *Comput. Linguistics* (1992).

[13] Ralf Brown and Robert Frederking. 1995. Applying statistical English language modeling to symbolic machine translation. In *TMI*. 221–239.

[14] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

[15] Pawel Budzianowski and Ivan Vulic. 2019. Hello, It's GPT-2 - How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems. In *NGT@EMNLP-IJCNLP*.

[16] Shuyang Cao and Lu Wang. 2021. Attention Head Masking for Inference Time Content Selection in Abstractive Summarization. In *NAACL-HLT*.

[17] Shuyang Cao and Lu Wang. 2021. Controllable Open-ended Question Generation with A New Question Type Ontology. In *ACL/IJCNLP*.

[18] Christopher Carroll. 2016. Measuring academic research impact: creating a citation profile using the conceptual framework for implementation fidelity as a case study. *Scientometrics* 109, 2 (2016), 1329–1340.

[19] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of Text Generation: A Survey. *arXiv preprint arXiv:2006.14799* (2020).

[20] Boxing Chen and Colin Cherry. 2014. A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU. In *WMT@ACL*.

[21] Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. 2021. VisualGPT: Data-efficient Adaptation of Pretrained Language Models for Image Captioning. *arXiv preprint arXiv:2102.10407* (2021).

[22] Jiaao Chen and Diyi Yang. 2020. Multi-View Sequence-to-Sequence Models with Conversational Structure for Abstractive Dialogue Summarization. In *EMNLP*.

[23] Jiaao Chen and Diyi Yang. 2021. Simple Conversational Data Augmentation for Semi-supervised Abstractive Dialogue Summarization. In *EMNLP*.

[24] Jiaao Chen and Diyi Yang. 2021. Structure-Aware Abstractive Conversation Summarization via Discourse and Action Graphs. In *NAACL-HLT*.

[25] Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020. KGPT: Knowledge-Grounded Pre-Training for Data-to-Text Generation. In *EMNLP*.

[26] Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2020. Distilling Knowledge Learned in BERT for Text Generation. In *ACL*.

[27] Yi-Syuan Chen and Hong-Han Shuai. 2021. Meta-Transfer Learning for Low-Resource Abstractive Summarization. In *AAAI*.

[28] Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020. Logic2Text: High-Fidelity Natural Language Generation from Logical Forms. In *EMNLP Findings*.

[29] Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020. Few-Shot NLG with Pre-Trained Language Model. In *ACL*.

[30] Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. 2020. Cross-Lingual Natural Language Generation via Pre-Training. In *AAAI*.

[31] Alexis Conneau and Guillaume Lample. 2019. Cross-lingual Language Model Pretraining. In *NeurIPS*.

[32] Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A Survey of Multilingual Neural Machine Translation. *CSUR* (2020).

[33] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. In *ICLR*.

[34] Erenay Dayanik and Sebastian Padó. 2020. Masking Actor Information Leads to Fairer Political Claims Detection. In *ACL*.

[35] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

[36] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *NeurIPS*.

[37] Yue Dong, Shuohang Wang, Zhe Gan, Yu Cheng, Jackie Chi Kit Cheung, and Jingjing Liu. 2020. Multi-Fact Correction in Abstractive Text Summarization. In *EMNLP*.

[38] Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. GSum: A General Framework for Guided Neural Abstractive Summarization. In *NAACL-HLT*.

[39] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. All NLP Tasks Are Generation Tasks: A General Pretraining Framework. *arXiv preprint arXiv:2103.10360* (2021).

[40] Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert Syst. Appl.* (2021).
[41] Gamaleldin F. Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. 2018. Large Margin Deep Networks for Classification. In *NeurIPS*.
[42] Alexander R. Fabbri, Simeng Han, Haoyuan Li, Haoran Li, Marjan Ghazvininejad, Shafiq R. Joty, Dragomir R. Radev, and Yashar Mehdad. 2021. Improving Zero and Few-Shot Abstractive Summarization with Intermediate Fine-tuning and Data Augmentation. In *NAACL-HLT*.
[43] Angela Fan and Claire Gardent. 2020. Multilingual AMR-to-Text Generation. In *EMNLP*.
[44] Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing Transformer Depth on Demand with Structured Dropout. In *ICLR*.
[45] Zhiyun Fan, Shiyu Zhou, and Bo Xu. 2019. Unsupervised pre-training for sequence to sequence speech recognition. *arXiv preprint arXiv:1910.12418* (2019).
[46] William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv preprint arXiv:2101.03961* (2021).
[47] Xiachong Feng, Xiaocheng Feng, and Bing Qin. 2021. A Survey on Dialogue Summarization: Recent Advances and New Frontiers. *arXiv preprint arXiv:2107.03175* (2021).
[48] Xiachong Feng, Xiaocheng Feng, Libo Qin, Bing Qin, and Ting Liu. 2021. Language Model as an Annotator: Exploring DialoGPT for Dialogue Summarization. In *ACL/IJCNLP*.
[49] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *EMNLP Findings*.
[50] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2020. Compressing Large-Scale Transformer-Based Models: A Case Study on BERT. *arXiv preprint arXiv:2002.11985* (2020).
[51] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *ACL*.
[52] Cristina Garbacea and Qiaozhu Mei. 2020. Neural Language Generation: Formulation, Methods, and Evaluation. *arXiv preprint arXiv:2007.15780* (2020).
[53] Xavier Garcia, Pierre Foret, Thibault Sellam, and Ankur P. Parikh. 2020. A Multilingual View of Unsupervised Machine Translation. In *EMNLP Findings*.
[54] Ismael Garrido-Muñoz, Arturo Montejo-Ráez, Fernando Martínez-Santiago, and L Alfonso Ureña-López. 2021. A survey on bias in deep NLP. *Applied Sciences* 11, 7 (2021), 3184.
[55] Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. On the Strengths of Cross-Attention in Pretrained Transformers for Machine Translation. *arXiv preprint arXiv:2104.08771* (2021).
[56] Alexios Gidiotis and Grigorios Tsoumakas. 2020. A Divide-and-Conquer Approach to the Summarization of Long Documents. *TASLP* (2020).
[57] Eli Goldberg, Norbert Driedger, and Richard I. Kittredge. 1994. Using Natural-Language Processing to Produce Weather Forecasts. *IEEE Expert* (1994).
[58] Sergey Golovanov, Rauf Kurbanov, Sergey I. Nikolenko, Kyryl Truskovskyi, Alexander Tselousov, and Thomas Wolf. 2019. Large-Scale Transfer Learning for Natural Language Generation. In *ACL*.
[59] Hila Gonen and Yoav Goldberg. 2019. Lipstick on a Pig: Debiasing Methods Cover up Systematic Gender Biases in Word Embeddings But do not Remove Them. In *WNLP@ACL*.
[60] Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. TableGPT: Few-shot Table-to-Text Generation with Table Structure Reconstruction and Content Matching. In *COLING*.
[61] Travis R. Goodwin, Max E. Savery, and Dina Demner-Fushman. 2020. Towards Zero Shot Conditional Summarization with Adaptive Multi-task Fine-Tuning. In *EMNLP Findings*.
[62] Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning. In *RepL4NLP@ACL*.
[63] Jing Gu, Qingyang Wu, Chongruo Wu, Weiyan Shi, and Zhou Yu. 2021. PRAL: A Tailored Pre-Training Model for Task-Oriented Dialog Generation. In *ACL/IJCNLP Short*.
[64] Xiaodong Gu, Kang Min Yoo, and Jung-Woo Ha. 2021. DialogBERT: Discourse-Aware Response Generation via Learning to Recover and Rank Utterances. In *AAAI*.
[65] Xiaodong Gu, Kang Min Yoo, and Sang-Woo Lee. 2021. Response Generation with Context-Aware Prompt Learning. *CoRR* abs/2111.02643 (2021).
[66] Jian Guan, Fei Huang, Minlie Huang, Zhihao Zhao, and Xiaoyan Zhu. 2020. A Knowledge-Enhanced Pretraining Model for Commonsense Story Generation. *TACL* (2020).

[67] Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2021. Supervised Contrastive Learning for Pre-trained Language Model Fine-tuning. In *ICLR*.

[68] Fu-Ming Guo, Sijia Liu, Finlay S. Mungall, Xue Lin, and Yanzhi Wang. 2019. Reweighted Proximal Pruning for Large-Scale Language Representation. *arXiv preprint arXiv:1909.12486* (2019).

[69] DongHoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-End Neural Pipeline for Goal-Oriented Dialogue Systems using GPT-2. In *ACL*.

[70] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-Trained Models: Past, Present and Future. *arXiv preprint arXiv:2106.07139* (2021).

[71] Boran Hao, Henghui Zhu, and Ioannis Ch. Paschalidis. 2020. Enhancing Clinical BERT Embedding using a Biomedical Knowledge Base. In *COLING*.

[72] Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have Your Text and Use It Too! End-to-End Neural Data-to-Text Generation with Semantic Fidelity. In *COLING*.

[73] Sadid A. Hasan and Oladimeji Farri. 2019. Clinical Natural Language Processing with Deep Learning. In *Data Science for Healthcare*.

[74] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-Enhanced Bert with Disentangled Attention. In *ICLR*.

[75] Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A Simple Language Model for Task-Oriented Dialogue. In *NeurIPS*.

[76] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. DynaBERT: Dynamic BERT with Adaptive Width and Depth. In *NeurIPS*.

[77] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *ICML*.

[78] Xinyu Hua, Ashwin Sreevatsa, and Lu Wang. 2021. DYPLOC: Dynamic Planning of Content Using Mixed Language Models for Text Generation. In *ACL/IJCNLP*.

[79] Luyang Huang, Shuyang Cao, Nikolaus Nova Parulian, Heng Ji, and Lu Wang. 2021. Efficient Attentions for Long Document Summarization. In *NAACL-HLT*.

[80] Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. 2020. Challenges in Building Intelligent Open-domain Dialog Systems. *TOIS* (2020).

[81] Xinting Huang, Jianzhong Qi, Yu Sun, and Rui Zhang. 2021. Latent Reasoning for Low-Resource Question Generation. In *ACL/IJCNLP Findings*.

[82] Touseef Iqbal and Shaima Qureshi. 2020. The survey: Text generation models in deep learning. *Journal of King Saud University-Computer and Information Sciences* (2020).

[83] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know. *TACL* (2020).

[84] Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. ConvBERT: Improving BERT with Span-based Dynamic Convolution. In *NeurIPS*.

[85] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *EMNLP Findings*.

[86] Di Jin, Zhijing Jin, Joey Tianyi Zhou, Lisa Orii, and Peter Szolovits. 2020. Hooks in the Headline: Learning to Generate Headlines with Controlled Styles. In *ACL*.

[87] Mihir Kale and Abhinav Rastogi. 2020. Template Guided Text Generation for Task-Oriented Dialogue. In *EMNLP*.

[88] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. 2021. AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing. *arXiv preprint arXiv:2108.05542* (2021).

[89] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361* (2020).

[90] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858* (2019).

[91] Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. 2021. A Distributional Approach to Controlled Text Generation. In *ICLR*.

[92] Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating Unsupervised Style Transfer as Paraphrase Generation. In *EMNLP*.

[93] Wojciech Kryscinski, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving Abstraction in Text Summarization. In *EMNLP*.

[94] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-Based & Neural Unsupervised Machine Translation. In *EMNLP*.
[95] Hang Le, Juan Miguel Pino, Changhan Wang, Jiatao Gu, Didier Schwab, and Laurent Besacier. 2021. Lightweight Adapter Tuning for Multilingual Speech Translation. In *ACL/IJCNLP Short*.
[96] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. 2015. Deep learning. *Nat.* (2015).
[97] Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models. In *ICLR*.
[98] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *ICLR*.
[99] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*.
[100] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL-HLT*.
[101] Jiwei Li and Eduard H. Hovy. 2014. A Model of Coherence Based on Distributed Sentence Representation. In *EMNLP*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 2039–2048.
[102] Junyi Li, Siqing Li, Wayne Xin Zhao, Gaole He, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2020. Knowledge-Enhanced Personalized Review Generation with Capsule Graph Neural Network. In *CIKM*.
[103] Jianquan Li, Xiaokang Liu, Honghong Zhao, Ruifeng Xu, Min Yang, and Yaohong Jin. 2020. BERT-EMD: Many-to-Many Layer Mapping for BERT Compression with Earth Mover's Distance. In *EMNLP*.
[104] Junyi Li, Tianyi Tang, Gaole He, Jinhao Jiang, Xiaoxuan Hu, Puzhao Xie, Zhipeng Chen, Zhuohao Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2021. TextBox: A Unified, Modularized, and Extensible Framework for Text Generation. In *ACL Demonstration*.
[105] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021. Few-shot Knowledge Graph-to-Text Generation with Pretrained Language Models. In *ACL/IJCNLP Findings*.
[106] Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Pretrained Language Models for Text Generation: A Survey. *CoRR* abs/2105.10311 (2021). arXiv:2105.10311 https://arxiv.org/abs/2105.10311
[107] Junyi Li, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021. Knowledge-based Review Generation by Coherence Enhanced Text Planning. In *SIGIR*.
[108] Junyi Li, Wayne Xin Zhao, Ji-Rong Wen, and Yang Song. 2019. Generating Long and Informative Reviews with Aspect-Aware Coarse-to-Fine Decoding. In *ACL*.
[109] Piji Li, Haisong Zhang, Xiaojiang Liu, and Shuming Shi. 2020. Rigid Formats Controlled Text Generation. In *ACL*.
[110] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL*.
[111] Zhongyang Li, Xiao Ding, Kuo Liao, Ting Liu, and Bing Qin. 2021. CausalBERT: Injecting Causal Knowledge Into Pre-trained Models with Minimal Supervision. *arXiv preprint arXiv:2107.09852* (2021).
[112] Zekang Li, Jinchao Zhang, Zhengcong Fei, Yang Feng, and Jie Zhou. 2021. Conversations Are Not Flat: Modeling the Dynamic Information Flow across Dialogue Utterances. In *ACL/IJCNLP*.
[113] Zuchao Li, Hai Zhao, Rui Wang, Masao Utiyama, and Eiichiro Sumita. 2020. Reference Language based Unsupervised Neural Machine Translation. In *EMNLP Findings*.
[114] Junwei Liao, Yu Shi, Ming Gong, Linjun Shou, Sefik Emre Eskimez, Liyang Lu, Hong Qu, and Michael Zeng. 2021. Generating Human Readable Transcript for Automatic Speech Recognition with Pre-Trained Language Model. In *ICASSP*.
[115] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*.
[116] Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020. Pre-training Multilingual Neural Machine Translation by Leveraging Alignment Information. In *EMNLP*.
[117] Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu, Linjun Shou, Ming Gong, Pengcheng Wang, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, Ruofei Zhang, Winnie Wu, Ming Zhou, and Nan Duan. 2021. GLGE: A New General Language Generation Evaluation Benchmark. In *ACL/IJCNLP Findings*.
[118] Junpeng Liu, Yanyan Zou, Hainan Zhang, Hongshen Chen, Zhuoye Ding, Caixia Yuan, and Xiaojie Wang. 2021. Topic-Aware Contrastive Learning for Abstractive Dialogue Summarization. In *EMNLP Findings*.
[119] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *arXiv preprint arXiv:2107.13586* (2021).
[120] Shilei Liu, Xiaofeng Zhao, Bochao Li, Feiliang Ren, Longhui Zhang, and Shujuan Yin. 2021. A Three-Stage Learning Framework for Low-Resource Knowledge-Grounded Dialogue Generation. In *EMNLP*.
[121] Yixin Liu, Zi-Yi Dou, and Pengfei Liu. 2021. RefSum: Refactoring Neural Summarization. In *NAACL-HLT*.

[122] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual Denoising Pre-training for Neural Machine Translation. *TACL* (2020).
[123] Yang Liu and Mirella Lapata. 2019. Text Summarization with Pretrained Encoders. In *EMNLP/IJCNLP*.
[124] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).
[125] Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. 2021. KG-BART: Knowledge Graph-Augmented BART for Generative Commonsense Reasoning. In *AAAI*.
[126] Zihan Liu, Genta Indra Winata, and Pascale Fung. 2021. Continual Mixed-Language Pre-Training for Extremely Low-Resource Neural Machine Translation. In *ACL/IJCNLP Findings*.
[127] Antoine Louis. 2020. *NetBERT: A Pre-trained Language Representation Model for Computer Networking*. Ph.D. Dissertation.
[128] Fuli Luo, Wei Wang, Jiahao Liu, Yijia Liu, Bin Bi, Songfang Huang, Fei Huang, and Luo Si. 2021. VECO: Variable and Flexible Cross-lingual Pre-training for Language Understanding and Generation. In *ACL/IJCNLP*.
[129] Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming Zhou. 2020. Univl: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353* (2020).
[130] Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md. Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A Language-Model-First Approach for AMR-to-Text Generation. In *ACL*.
[131] Ahmed Magooda and Diane J. Litman. 2021. Mitigating Data Scarceness through Data Synthesis, Augmentation and Curriculum for Abstractive Summarization. In *EMNLP Findings*.
[132] Bodhisattwa Prasad Majumder, Sudha Rao, Michel Galley, and Julian J. McAuley. 2021. Ask what's missing and what's useful: Improving Clarification Question Generation using Global Knowledge. In *NAACL-HLT*.
[133] Potsawee Manakul and Mark J. F. Gales. 2021. Long-Span Summarization via Local Attention and Content Selection. In *ACL/IJCNLP*.
[134] Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian J. McAuley, and Garrison W. Cottrell. 2019. Improving Neural Story Generation by Targeted Common Sense Grounding. In *EMNLP/IJCNLP*.
[135] Feng Nan, Cícero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Kathleen R. McKeown, Ramesh Nallapati, Dejiao Zhang, Zhiguo Wang, Andrew O. Arnold, and Bing Xiang. 2021. Improving Factual Consistency of Abstractive Summarization via Question Answering. In *ACL*.
[136] Thong Nguyen, Anh Tuan Luu, Truc Lu, and Tho Quan. 2021. Enriching and Controlling Global Semantics for Text Summarization. In *EMNLP*.
[137] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *NAACL-HLT Demonstrations*.
[138] Siru Ouyang, Zhuosheng Zhang, and Hai Zhao. 2021. Dialogue Graph Modeling for Conversational Machine Reading. In *ACL/IJCNLP Findings*.
[139] Xiao Pan, Mingxuan Wang, Liwei Wu, and Lei Li. 2021. Contrastive Learning for Many-to-many Multilingual Neural Machine Translation. In *ACL/IJCNLP*.
[140] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL*.
[141] Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. A Plug-and-Play Method for Controlled Text Generation. In *EMNLP Findings*.
[142] 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 *AAAI*.
[143] Ramakanth Pasunuru, Mengwen Liu, Mohit Bansal, Sujith Ravi, and Markus Dreyer. 2021. Efficiently Summarizing Text and Graph Encodings of Multi-Document Clusters. In *NAACL-HLT*.
[144] Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot Natural Language Generation for Task-Oriented Dialog. In *EMNLP Findings*.
[145] Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge Enhanced Contextual Word Representations. In *EMNLP/IJCNLP*.
[146] Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks. *arXiv preprint arXiv:1811.01088* (2018).
[147] Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How Multilingual is Multilingual BERT?. In *ACL*.
[148] Maja Popovic. 2017. chrF++: words helping character n-grams. In *WMT*.
[149] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *WMT*.
[150] Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training. In *EMNLP Findings*.

[151] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained Models for Natural Language Processing: A Survey. *arXiv preprint arXiv:2003.08271* (2020).
[152] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
[153] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* (2019).
[154] 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. *JMLR* (2020).
[155] Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking. In *EMNLP*.
[156] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *SIGKDD*.
[157] Shuo Ren, Yu Wu, Shujie Liu, Ming Zhou, and Shuai Ma. 2019. Explicit Cross-lingual Pre-training for Unsupervised Machine Translation. In *EMNLP/IJCNLP*.
[158] Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating Pretrained Language Models for Graph-to-Text Generation. *arXiv preprint arXiv:2007.08426* (2020).
[159] Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural Adapters in Pretrained Language Models for AMR-to-Text Generation. In *EMNLP*.
[160] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. Recipes for Building an Open-Domain Chatbot. In *EACL*.
[161] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks. *TACL* (2020).
[162] Itsumi Saito, Kyosuke Nishida, Kosuke Nishida, and Junji Tomita. 2020. Abstractive Summarization with Combination of Pre-trained Sequence-to-Sequence and Saliency Models. *arXiv preprint arXiv:2003.13028* (2020).
[163] Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. ColdGANs: Taming Language GANs with Cautious Sampling Strategies. In *NeurIPS*.
[164] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*.
[165] Igor Shalyminov, Alessandro Sordoni, Adam Atkinson, and Hannes Schulz. 2020. Hybrid Generative-Retrieval Transformers for Dialogue Domain Adaptation. *arXiv preprint arXiv:2003.01680* (2020).
[166] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *EMNLP*.
[167] Sam Shleifer and Alexander M. Rush. 2020. Pre-trained Summarization Distillation. *arXiv preprint arXiv:2010.13002* (2020).
[168] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked Sequence to Sequence Pre-training for Language Generation. In *ICML*.
[169] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958.
[170] Asa Cooper Stickland, Xian Li, and Marjan Ghazvininejad. 2021. Recipes for Adapting Pre-trained Monolingual and Multilingual Models to Machine Translation. In *EACL*.
[171] Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gribonval, Hervé Jégou, and Armand Joulin. 2021. Training with Quantization Noise for Extreme Model Compression. In *ICLR*.
[172] Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. 2021. Towards Table-to-Text Generation with Numerical Reasoning. In *ACL/IJCNLP*.
[173] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. 2019. Contrastive Bidirectional Transformer for Temporal Representation Learning. *arXiv preprint arXiv:1906.05743* (2019).
[174] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019. VideoBERT: A Joint Model for Video and Language Representation Learning. In *ICCV*.
[175] Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, Weixin Liu, Zhihua Wu, Weibao Gong, Jianzhong Liang, Zhizhou Shang, Peng Sun, Wei Liu, Xuan Ouyang, Dianhai Yu, Hao Tian, Hua Wu, and Haifeng Wang. 2021. ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation. *CoRR* abs/2107.02137 (2021).
[176] Zewei Sun, Mingxuan Wang, and Lei Li. 2021. Multilingual Translation via Grafting Pre-trained Language Models. In *EMNLP Findings*.
[177] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*.
[178] Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2021. Multilingual Translation from Denoising Pre-Training. In *ACL/IJCNLP Findings*.

[179] Tao Tao, Xuanhui Wang, Qiaozhu Mei, and ChengXiang Zhai. 2006. Language Model Information Retrieval with Document Expansion. In *NAACL*.

[180] 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 *NIPS*.

[181] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. 2020. Investigating Gender Bias in Language Models Using Causal Mediation Analysis. In *NeurIPS*.

[182] Takashi Wada and Tomoharu Iwata. 2018. Unsupervised Cross-lingual Word Embedding by Multilingual Neural Language Models. *arXiv preprint arXiv:1809.02306* (2018).

[183] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *NeurIPS*.

[184] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *ICLR*.

[185] Danqing Wang, Jiaze Chen, Hao Zhou, Xipeng Qiu, and Lei Li. 2021. Contrastive Aligned Joint Learning for Multilingual Summarization. In *ACL/IJCNLP Findings*.

[186] Wei Wang, Piji Li, and Hai-Tao Zheng. 2021. Consistency and Coherency Enhanced Story Generation. In *ECIR*.

[187] Xiaohui Wang, Ying Xiong, Yang Wei, Mingxuan Wang, and Lei Li. 2021. LightSeq: A High Performance Inference Library for Transformers. In *NAACL-HLT Industry*.

[188] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP Demonstrations*.

[189] Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents. *arXiv preprint arXiv:1901.08149* (2019).

[190] Wenhao Wu, Wei Li, Xinyan Xiao, Jiachen Liu, Ziqiang Cao, Sujian Li, Hua Wu, and Haifeng Wang. 2021. BASS: Boosting Abstractive Summarization with Unified Semantic Graph. In *ACL/IJCNLP*.

[191] Qiaolin Xia, Haoyang Huang, Nan Duan, Dongdong Zhang, Lei Ji, Zhifang Sui, Edward Cui, Taroon Bharti, and Ming Zhou. 2021. XGPT: Cross-modal Generative Pre-Training for Image Captioning. In *NLPCC*.

[192] Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models. In *EMNLP*.

[193] Xinnuo Xu, Guoyin Wang, Young-Bum Kim, and Sungjin Lee. 2021. AugNLG: Few-shot Natural Language Generation using Self-trained Data Augmentation. In *ACL*.

[194] 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 *NAACL-HLT*.

[195] Lanqing Xue, Kaitao Song, Duocai Wu, Xu Tan, Nevin L. Zhang, Tao Qin, Wei-Qiang Zhang, and Tie-Yan Liu. 2021. DeepRapper: Neural Rap Generation with Rhyme and Rhythm Modeling. In *ACL/IJCNLP*.

[196] Yu Yan, Fei Hu, Jiusheng Chen, Nikhil Bhendawade, Ting Ye, Yeyun Gong, Nan Duan, Desheng Cui, Bingyu Chi, and Ruifei Zhang. 2021. FastSeq: Make Sequence Generation Faster. *arXiv preprint arXiv:2106.04718* (2021).

[197] Jiacheng Yang, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Weinan Zhang, Yong Yu, and Lei Li. 2020. Towards Making the Most of BERT in Neural Machine Translation. In *AAAI*.

[198] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*.

[199] Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020. CSP: Code-Switching Pre-training for Neural Machine Translation. In *EMNLP*.

[200] Zhengyuan Yang, Yijuan Lu, Jianfeng Wang, Xi Yin, Dinei Florêncio, Lijuan Wang, Cha Zhang, Lei Zhang, and Jiebo Luo. 2021. TAP: Text-Aware Pre-Training for Text-VQA and Text-Caption. In *CVPR*.

[201] Ze Yang, Wei Wu, Can Xu, Xinnian Liang, Jiaqi Bai, Liran Wang, Wei Wang, and Zhoujun Li. 2020. StyleDGPT: Stylized Response Generation with Pre-trained Language Models. In *EMNLP Findings*.

[202] Ziyi Yang, Chenguang Zhu, Robert Gmyr, Michael Zeng, Xuedong Huang, and Eric Darve. 2020. TED: A Pretrained Unsupervised Summarization Model with Theme Modeling and Denoising. In *EMNLP Findings*.

[203] Weiqiu You, Simeng Sun, and Mohit Iyyer. 2020. Hard-Coded Gaussian Attention for Neural Machine Translation. In *ACL*. 7689–7700.

[204] Ali Hadi Zadeh, Isak Edo, Omar Mohamed Awad, and Andreas Moshovos. 2020. GOBO: Quantizing Attention-Based NLP Models for Low Latency and Energy Efficient Inference. In *MICRO*.

[205] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences.

In *NeurIPS*.
[206] Munazza Zaib, Quan Z. Sheng, and Wei Emma Zhang. 2020. A Short Survey of Pre-trained Language Models for Conversational AI-A New Age in NLP. In *ACSW*.
[207] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, Chen Li, Ziyan Gong, Yifan Yao, Xinjing Huang, Jun Wang, Jianfeng Yu, Qi Guo, Yue Yu, Yan Zhang, Jin Wang, Hengtao Tao, Dasen Yan, Zexuan Yi, Fang Peng, Fangqing Jiang, Han Zhang, Lingfeng Deng, Yehong Zhang, Zhe Lin, Chao Zhang, Shaojie Zhang, Mingyue Guo, Shanzhi Gu, Gaojun Fan, Yaowei Wang, Xuefeng Jin, Qun Liu, and Yonghong Tian. 2021. PanGu-α: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation. *arXiv preprint arXiv:2104.12369* (2021).
[208] Yan Zeng and Jian-Yun Nie. 2020. Generalized Conditioned Dialogue Generation Based on Pre-trained Language Model. *arXiv preprint arXiv:2010.11140* (2020).
[209] Yan Zeng and Jian-Yun Nie. 2021. A Simple and Efficient Multi-Task Learning Approach for Conditioned Dialogue Generation. In *NAACL-HLT*.
[210] ChengXiang Zhai and John D. Lafferty. 2001. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *CIKM*. 403–410.
[211] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In *ICML*.
[212] 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 *EMNLP Findings*.
[213] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *ICLR*.
[214] Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization. In *ACL*.
[215] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-Scale Generative Pre-training for Conversational Response Generation. In *ACL Demonstrations*.
[216] Zhengyan Zhang, Yuxian Gu, Xu Han, Shengqi Chen, Chaojun Xiao, Zhenbo Sun, Yuan Yao, Fanchao Qi, Jian Guan, Pei Ke, Yanzheng Cai, Guoyang Zeng, Zhixing Tan, Zhiyuan Liu, Minlie Huang, Wentao Han, Yang Liu, Xiaoyan Zhu, and Maosong Sun. 2021. CPM-2: Large-scale Cost-effective Pre-trained Language Models. *arXiv preprint arXiv:2106.10715* (2021).
[217] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *ACL*.
[218] Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, YuSheng Su, Haozhe Ji, Jian Guan, Fanchao Qi, Xiaozhi Wang, Yanan Zheng, Guoyang Zeng, Huanqi Cao, Shengqi Chen, Daixuan Li, Zhenbo Sun, Zhiyuan Liu, Minlie Huang, Wentao Han, Jie Tang, Juanzi Li, Xiaoyan Zhu, and Maosong Sun. 2020. CPM: A Large-scale Generative Chinese Pre-trained Language Model. *arXiv preprint arXiv:2012.00413* (2020).
[219] Zhilu Zhang and Mert R. Sabuncu. 2018. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In *NeurIPS*.
[220] Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. 2020. Recent advances and challenges in task-oriented dialog systems. *Sci. China Technol. Sci.* (2020).
[221] Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018. Learning Gender-Neutral Word Embeddings. In *EMNLP*.
[222] Hao Zheng and Mirella Lapata. 2019. Sentence Centrality Revisited for Unsupervised Summarization. In *ACL*.
[223] Ming Zhong, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. DialogLM: Pre-trained Model for Long Dialogue Understanding and Summarization. *arXiv preprint arXiv:2109.02492* (2021).
[224] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. The Design and Implementation of XiaoIce, an Empathetic Social Chatbot. *Comput. Linguistics* (2020).
[225] Wangchunshu Zhou, Dong-Ho Lee, Ravi Kiran Selvam, Seyeon Lee, and Xiang Ren. 2021. Pre-training Text-to-Text Transformers for Concept-centric Common Sense. In *ICLR*.
[226] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating BERT into Neural Machine Translation. In *ICLR*.
[227] Yicheng Zou, Bolin Zhu, Xingwu Hu, Tao Gui, and Qi Zhang. 2021. Low-Resource Dialogue Summarization with Domain-Agnostic Multi-Source Pretraining. In *EMNLP*.