A Survey of Knowledge-Enhanced Text Generation

WENHAO YU, University of Notre Dame
CHENGUANG ZHU, Microsoft Research
ZAITANG LI, The Chinese University of Hong Kong
ZHITING HU, University of California at San Diego
QINGYUN WANG, University of Illinois at Urbana-Champaign
HENG JI, University of Illinois at Urbana-Champaign
MENG JIANG, University of Notre Dame

The goal of text generation is to make machines express in human language. It is one of the most important yet challenging tasks in natural language processing (NLP). Since 2014, various neural encoder-decoder models pioneered by Seq2Seq have been proposed to achieve the goal by learning to map input text to output text. However, the input text alone often provides limited knowledge to generate the desired output, so the performance of text generation is still far from satisfaction in many real-world scenarios. To address this issue, researchers have considered incorporating various forms of knowledge beyond the input text into the generation models. This research direction is known as *knowledge-enhanced text generation*. In this survey, we present a comprehensive review of the research on knowledge enhanced text generation over the past five years. The main content includes two parts: (i) general methods and architectures for integrating knowledge into text generation; (ii) specific techniques and applications according to different forms of knowledge data. This survey can have broad audiences, researchers and practitioners, in academia and industry.

ACM Reference Format:
Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2020. A Survey of Knowledge-Enhanced Text Generation. *ACM Comput. Surv.* 1, 1, Article 1 (January 2020), 44 pages. https://doi.org/xxx.xxx

1 INTRODUCTION

Text generation, which is often formally referred as natural language generation (NLG), is one of the most important yet challenging tasks in natural language processing (NLP) [49]. NLG aims at producing understandable text in human language from linguistic or non-linguistic data in a variety of forms such as textual data, numerical data, image data, structured knowledge bases, and knowledge graphs. Among these, text-to-text generation is one of the most important applications and thus often referred as "text generation". Researchers have developed numerous technologies and a wide range of applications for this task [50, 66, 142, 168]. Text generation takes text (e.g., a sequence, keywords) as input, processes the input text into semantic representations, and generates

Authors' addresses: Wenhao Yu, wyu1@nd.edu, University of Notre Dame, Notre Dame, Indiana, 46556; Chenguang Zhu, chezhu@microsoft.com, Microsoft Research, Redmond, Washington, 98052; Zaitang Li, 1155107739@link.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, 999077; Zhiting Hu, zhitinghu@gmail.com, University of California at San Diego, San Diego, California, 92092; Qingyun Wang, qingyun4@illinois.edu, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801; Heng Ji, hengji@illinois.edu, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801; Meng Jiang, mjiang2@nd.edu, University of Notre Dame, Notre Dame, Indiana, 46556.

0360-0300/2020/1-ART1 $15.00
https://doi.org/xxx.xxx

desired output text. For example, machine translation generates text in a different language based on the source text; summarization generates an abridged version of the source text to include salient information; question answering (QA) generates textual answers to given questions; dialogue system supports chatbots to communicate with humans with generated responses.

With the recent resurgence of deep learning technologies [78], deep neural NLG models have achieved remarkable performance in enabling machines to understand and generate natural language. A basic definition of the text generation task is to generate an expected *output sequence* from a given *input sequence*, called sequence-to-sequence (Seq2Seq). The Seq2Seq task and model were first introduced in 2014 [29, 135]. It maps an input text to an output text under encoder-decoder schemes. The encoder maps the input sequence to a fixed-sized vector, and the decoder maps the vector to the target sequence. Since then, developing NLG systems has rapidly become a hot topic. Various text generation models have been proposed under deep neural encoder-decoder architectures. Popular architectures include recurrent neural network (RNN) encoder-decoder [29, 135], convolutional neural network (CNN) encoder-decoder [51], and Transformer [139]. The attention mechanism [4] and copy/pointing mechanism [54, 123] are two widely used mechanisms to improve the performance of generation models. Encoder-decoder models have been used for various NLG tasks such as machine translation [168], summarization [88].

Nevertheless, the input text alone contains limited knowledge to support neural generation models to produce the desired output, so the performance of generation is still far from satisfaction in many real-world scenarios. For example, in dialogue systems, conditioning on only the input text, a text generation system often produces trivial or non-committal responses of frequent words or phrases in the corpus [102, 158, 180], such as *"Me too."* or *"Oh my god!"* given the input text *"My skin is so dry."* These mundane responses lack meaningful content, in contrast to human responses rich in knowledge. In comparison, humans are constantly acquiring, understanding, and storing knowledge so that the learned knowledge from *broader sources* can be employed to understand the current situation in communicating, reading, and writing. For instance, in conversations, people often first select *concepts from related topics* (e.g., sports, food), then organize them into understandable content to respond; for summarization, people tend to write summaries containing *keywords* used in the input document and perform necessary modifications to ensure grammatical correctness and fluency; in question answering (QA), people use *commonsense* or *professional knowledge* pertained to the question to infer the answer. Therefore, it is often the case that knowledge beyond the input sequence is required to produce informative output text.

1.1 What is Knowledge-enhanced Text Generation?

In general, knowledge is a familiarity, awareness, or understanding that coalesces around a particular subject [61]. In NLG systems, knowledge is an awareness and understanding of the input text and its surrounding context. This knowledge can be learnt by many different methods and from various information sources, including but not limited to keywords, topics, linguistic features, knowledge bases, knowledge graphs, and grounded texts (see Figure 1). In other words, these sources provide information (e.g., commonsense triples, topic words, reviews, background documents) that can be used as knowledge through different neural representation learning methods, and then applied to enhance the process of text generation. The research direction of incorporating knowledge into text generation is known as *knowledge-enhanced text generation.*

Problem 1 (Knowledge-enhanced Text Generation). *Given a text generation problem where the system is given an input sequence X, and aims to generate an output sequence Y Assume we also have access to additional knowledge denoted as K. Knowledge-enhanced text generation aims*

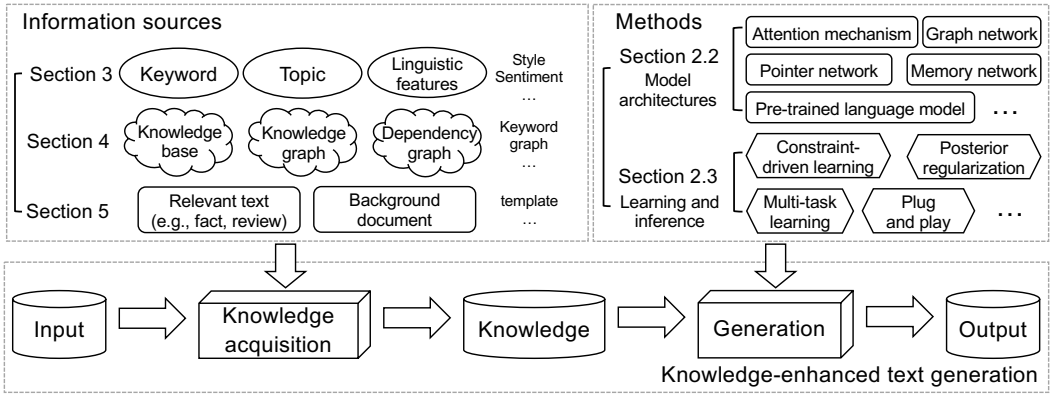

Fig. 1. Categorization of information sources and methods used in knowledge-enhanced text generation systems. Knowledge can be learnt from various information sources, and then integrated into the generation process by a number of methods. Information sources and methods are not limited to the ones listed above.

to incorporate the knowledge K to enhance the generation of Y given X, through leveraging the dependencies among the input, knowledge, and output.

Many existing knowledge-enhanced text generation systems have demonstrated promising performances on generating informative, logical, and coherent texts. In dialogue system, a topic-aware Seq2Seq model helps understand the semantic meaning of an input sequence and generate a more informative response such as *"Then hydrate and moisturize your skin."* to the aforementioned example input *"My skin is so dry."* In summarization, knowledge graph helps produce a structured summary and highlight the proximity of relevant concepts, when complex events related with the same entity may span multiple sentences. A knowledge graph enhanced Seq2Seq model has been shown to generate summaries that are able to correctly answer 10% more document-related questions [65]. In question answering (QA) systems, facts stored in knowledge bases help complete missing information in the question and elaborate details to facilitate answer generation [40, 60]. In story generation, using commonsense knowledge acquired from knowledge graph helps facilitate understanding of the storyline and better narrate following plots step by step, so each step could be reflected as a link on the knowledge graph and the whole story would be a path [56].

1.2 Why a Survey of Knowledge-enhanced Text Generation?

Incorporating knowledge in NLG beyond input text is seen as a promising direction in both academia and industry. Therefore, researchers have proposed various methods to tackle this problem by incorporating knowledge acquired from different information sources. Existing surveys in this area have only partially reviewed some related topics. Garbacea et al. and Gatt et al. provided synthesis of research on the core NLG tasks and main architectures adopted in each task [49, 50], but they did not go deeper to the knowledge-enhanced text generation. Ji et al. conducted a review on knowledge graph techniques, some of which have been applied to enhance NLG performance [71]. Wang et al. summarized how to represent structural knowledge (i.e., knowledge base and knowledge graph) for a number of NLP tasks such as reading comprehension and retrieval [142].

To the best of our knowledge, our survey is the first work that presents a comprehensive review of knowledge-enhanced text generation. It aims to provide NLG researchers a synthesis and pointer to related researches. Our survey also includes a detailed discussion about how NLG can benefit

Table 1. We organize recent work on knowledge-enhanced text generation according to two dimensions: application and source of knowledge. In the table header, **LF** is short for linguistic features, **KB** is short for knowledge base, **KG** is short for knowledge graph, **DG** is short for dependency graph, and **KeyG** is short for keyword graph, and **GT** is short for grounded text. Due to space limitation, we give an external long reading list and code collection in here: *https://github.com/wyu97/KENLG-Reading*.

Application	Input	Section 3			Section 4				Section 5
		Topic	Keyword	LF	KB	KG	DG	KeyG	GT
Dialogue system	Sentence (i.e., utterance)	[158, 164]	[102, 132] [151, 179]	-	[39, 93, 155] [85, 144, 154]	[108, 180] [138, 173]	-	-	[36, 52, 169] [96, 114, 121]
Question answering	Sentence (i.e., query)	-	-	-	[10, 44] [60, 170]	[7, 26, 40]	-	-	[11, 23, 48]
Question generation	Sentence (i.e., answer)	-	-	[181]	-	-	[27, 109]	[59]	-
Creative writing (e.g., story, paper)	Document (e.g., paper title)	-	-	[38]	-	[56, 70] [75, 149]	-	-	-
Summarization	Document (e.g., an article)	[105, 146] [43, 149]	[80, 84] [81]	[2, 104]	[166]	[65, 183]	[72, 87]	[84]	[18, 47, 145]
Machine translation	Sentence	[152, 174]	-	[31, 124]	-	[103, 177]	[1, 6] [131]		[15, 165]
Content manipulation	Document	[83]	-	-	[41, 67, 150]	-	-		-
Content paraphrasing	Document	[46, 90]	-	-	-	-			[22, 73]

from recent progress in deep learning and artificial intelligence, including technologies such as graph neural network, reinforcement learning, neural topic modeling and so on.

To start with, we note that *the primary challenge* in knowledge-enhanced NLG is how to *obtain* useful related knowledge from diverse sources. There has been a rising line of work that discovers knowledge from topic, keyword, knowledge base, knowledge graph and knowledge grounded text. *The second challenge* is how to effectively *understand* and *leverage* the acquired knowledge to facilitate text generation. Multiple methods have been explored to improve the encoder-decoder architecture (e.g., attention mechanism, copy and pointing mechanism).

Based on the first challenge, the main content of our survey is divided into two parts: (1) general methods of integrating knowledge into text generation (Section 2); (2) specific methods and applications according to different sources of knowledge enhancement (Sections 3–5). Based on the second challenge, we categorize recent knowledge-enhanced text generation methods evolved from how knowledge is extracted and incorporated into the process of text generation in each section (named as M1, M2, and etc). Furthermore, we review methods for a variety of NLG applications (see Table 1) in each section to help practitioners choose, learn, and use the methods. In total, we discuss seven mainstream applications presented in more than 80 papers that were published or released during the past five years (2016–2020).

1.3 How is the Survey Organized?

The remainder of this survey is organized as follows. Section 2 presents basic text generation models and general methods of integrating knowledge into text generation. Sections 3–5 review knowledge-enhanced text generation methods and applications according to different sources of knowledge enhancement. Section 6–7 discusses future works and concludes the survey.

Table 2. Symbols used in this survey and their descriptions.

Symbol	Description
$X, \mathcal{X}$	input sequence item, and the set of all input sequences
$Y, \mathcal{Y}$	output sequence item, and the set of all output sequences
$x_i, \mathbf{e}(x_i)$	i-th word in the input sequence and its word embedding
$y_i, \mathbf{e}(y_i)$	i-th word in the output sequence and its word embedding
$y_{<t}$	a sequence of words from y_0 to y_{t-1}, i.e., $(y_0, \cdots, y_{t-1})$
$\mathbf{h}_i$, $\mathbf{s}_i$, $\mathbf{c}_i$	encoding, decoding hidden state and decoding context state at i-th step
$f_{en}(\cdot), f_{de}(\cdot)$	nonlinear encoding and decoding functions (e.g., LSTM, Transformer)
$\psi_{mode}(\cdot)$	a nonlinear score function for a decoding *mode*
$\eta(\cdot)$	a nonlinear, potentially multi-layered, function (e.g., MLP)
$\sigma(\cdot)$	a nonlinear activation function (e.g., tanh, ReLU)
$\mathcal{V}, \mathcal{V}_X$	pre-defined vocabulary and source sequence vocabulary

2 GENERAL METHODS OF INTEGRATING KNOWLEDGE INTO NLG

2.1 The Basic Text Generation Models

Early encoder-decoder frameworks are often based on recurrent neural network (RNN), such as RNN-Seq2Seq [4, 135]. More recently, convolutional neural network (CNN) based encoder-decoder [51] and Transformer encoder-decoder [139] have also been increasingly widely used. Basically, the encoder learns to encode a variable length sequence into a fixed length vector representation. The decoder is to decode a given fixed length vector representation into a variable length sequence [29, 135]. From a probabilistic perspective, the encoder-decoder model framework learns the conditional distribution over a variable length sequence conditioned on yet another variable length sequence:

$$P(Y|X) = P(y_1, \cdots, y_m|x_1, \cdots, x_n) = \prod_{t=1}^{m} p(y_t|X, y_1, \cdots, y_{t-1}). \tag{1}$$

RNN-Seq2Seq [4, 135]. The encoder reads the input sentence X *sequentially*:

$$\mathbf{h}_i = f_{rnn-en}(\mathbf{e}(x_i), \mathbf{h}_{i-1}), \tag{2}$$

where $\mathbf{e}(x_i)$ is the word embedding of word x_i, $\mathbf{h}_i$ is the contextualized hidden representation of x_i, and $f_{rnn-en}(\cdot)$ is an RNN-based encoder (e.g., GRU, LSTM). The last hidden state $\mathbf{h}_n$ is seen as the representation of the whole input sequence, denoted as $\mathbf{c} = \mathbf{h}_n$. During the decoding phase, it adopts another RNN-based function $f_{de}(\cdot)$ to generate the output sequence by predicting the next word y_t based on the hidden state $\mathbf{s}_t$. Both y_t and $\mathbf{s}_t$ are conditioned on $\mathbf{e}(y_{t-1})$ and $\mathbf{c}$ of the input sequence. Formally, the t-th hidden state and the decoding function can be written as

$$\mathbf{s}_t = f_{rnn-de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}), \tag{3}$$

$$p(y_t|y_{t-1}, y_{t-2}, \cdots, y_1, \mathbf{c}) = f_{mlp}(\mathbf{s}_t, \mathbf{e}(y_{t-1}), \mathbf{c}), \tag{4}$$

where $f_{mlp}(\cdot)$ is a nonlinear multi-layered function that outputs the probability of y_t.

Transformer [139]. The encoder maps an input sequence to a sequence of continuous representations. The encoder is composed of a stack of identical layers. Each layer has two sub-layers. The first is a multi-head self-attention network, and the second is a position-wise fully connected feed-forward network. For brevity, we use $f_{tf-en}(\cdot)$ to represent the transformer encoder,

$$(\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n) = f_{tf-en}(\mathbf{e}(x_1), \mathbf{e}(x_2), \cdots, \mathbf{e}(x_n)) \tag{5}$$

The decoder is also composed of a stack of identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n)$. Given $\mathbf{H}$, the decoder then generates an output sequence one element at a time. At each step the model is auto-regressive, consuming the previously generated symbols as additional input when generating the next. Efficient implementations of the transformer use the cached history matrix $\mathbf{S}_t$ to generate next token. To compare with RNN-Seq2Seq more intuitively, we summarize the transformer decoder using recurrent notation,

$$\mathbf{S}_t = f_{tf-de}(\mathbf{S}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{H}), \tag{6}$$

where $\mathbf{S}_t = [(\mathbf{K}_t^{(1)}, \mathbf{V}_t^{(1)}), \cdots, (\mathbf{K}_t^{(l)}, \mathbf{V}_t^{(l)})]$, where $(\mathbf{K}_t^{(i)}, \mathbf{V}_t^{(i)})$ corresponds to the key-value pairs from the i-th layer generated at all time-steps from 0 to t. The last layer of $\mathbf{S}_t$ can be further mapped to a logit vector. For brevity, we will use $f_{en}(\cdot)$ and $f_{de}(\cdot)$ to represent encoder and decoder in the following sections, instead of specifically referring to a certain kind of encoder and decoder.

Optimization. A generation process is regarded as a sequential multi-label classification problem. It can be directly optimized by the negative log likelihood (*NLL*) loss. Therefore, the objective of a text generation model via maximum likelihood estimation (MLE) is formulated as:

$$\mathcal{L}_{NLL}(\theta) = -\log p_\theta(Y|X) = -\sum_{t=1}^{m} \log\left(p_\theta(y_t|y_{<t}, X)\right). \tag{7}$$

2.2 Knowledge-enhanced Model Architectures

The perhaps most popular way of enhancing text generation with knowledge is by designing specialized model architectures that reflect the particular knowledge. In the context of neural networks, several general neural architectures are widely used (and customized) to bake the knowledge about the problems being tackled into the model.

2.2.1 Attention Mechanism. It is useful to capture the weight of each time step in both encoder and decoder [4]. During the decoding phase, the context vector $\mathbf{c}_t$ is added, so the hidden state $\mathbf{s}_t$ is:

$$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t). \tag{8}$$

Unlike the vanilla Seq2Seq approach stated in Eq.(3), here the probability is conditioned on the distinct context vector $\mathbf{c}_t$ for target word y_t, and $\mathbf{c}_t$ depends on a sequence of hidden states $\{\mathbf{h}_i\}_{i=1}^n$ that were mapped from input sequence. Then $\mathbf{c}_t$ is computed as a weighted sum of $\{\mathbf{h}_i\}_{i=1}^n$:

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{ti}\mathbf{h}_i, \text{ where } \alpha_{ti} = \frac{\exp(\eta(\mathbf{s}_{t-1}, \mathbf{h}_i))}{\sum_{k=1}^{n} \exp(\eta(\mathbf{s}_{t-1}, \mathbf{h}_k))}, \tag{9}$$

where $\eta(\cdot)$ is parametrized as a multi-layer perception to compute a soft alignment. $\eta(\cdot)$ enables the gradient of loss function to be backpropagated. There are six alternatives for the $\eta(\cdot)$ function (see Table 2 in [49]). The probability α_{ti} reflects the importance of the hidden state of input sequence in presence of the previous hidden state $\mathbf{s}_{t-1}$ for deciding the next hidden state.

Knowledge-related attention. Leveraging attention mechanism to incorporate knowledge representation into the decoding phase has been widely used in recent knowledge-enhanced NLG work. The general idea is to learn a knowledge-aware context vector (denoted as $\widetilde{\mathbf{c}}_t$) by integrating both hidden context vector ($\mathbf{c}_t$) and knowledge context vector (denoted as $\mathbf{c}_t^K$) into decoder update, such as $\widetilde{\mathbf{c}}_t = f_{mlp}(\mathbf{c}_t \oplus \mathbf{c}_t^K)$. The knowledge context vector ($\mathbf{c}_t^K$) calculates attentions over knowledge representations (e.g., topic vectors, node vectors in knowledge graph). Table 3 summarizes of different kinds of knowledge attention proposed by recent papers, including keyword

Table 3. Natural language generation methods that incorporates knowledge-related attention.

Sources	Decoder hidden update	knowledge context attention	Papers
Topic	$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^{tp})$	$\mathbf{c}_t^{tp} = \sum_{j=1}^{\|P\|} \alpha_{tj}\mathbf{p}_j,\ \alpha_{tj} = \frac{\exp(\eta(\mathbf{s}_{t-1}, \mathbf{p}_j, \mathbf{h}_n))}{\sum_{j'=1}^{\|P\|}\exp(\eta(\mathbf{s}_{t-1}, \mathbf{p}_{j'}, \mathbf{h}_n))}$	[90, 152, 158]
	Description: $\mathbf{c}_t^{tp}$ is the topic-aware attention at time step t; $\mathbf{p}_j$ denotes the embedding of j-th topic word generated from topic models (e.g., LDA); $\mathbf{h}_n$ is the last hidden state of encoder used to weaken the effect of irrelevant topic words and highlight the importance of relevant topic words; P is the set of topic words.		
	$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^{tp})$	$\mathbf{c}_t^{tp} = \sum_{j=1}^{\|X\|} \alpha_{tj}\mathbf{q}_j,\ \alpha_{tj} = \frac{\exp(\eta(\mathbf{s}_{t-1}, \mathbf{q}_j)}{\sum_{j'=1}^{\|X\|}\exp(\eta(\mathbf{s}_{t-1}, \mathbf{q}_{j'}))}$	[174]
	Description: $\mathbf{q}_j$ denotes the word distributions over topics for each word in input sequence X; different from topic word attention, it directly calculates attention between decoder hidden state and topic distribution.		
Keyword	$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^{kw})$	$\mathbf{c}_t^{kw} = \sum_{j=1}^{\|K\|} \alpha_{tj}\mathbf{k}_j,\ \alpha_{tj} = \frac{\exp(\eta(\mathbf{s}_{t-1}, \mathbf{k}_j))}{\sum_{j'=1}^{\|K\|}\exp(\eta(\mathbf{s}_{t-1}, \mathbf{k}'_j))}$	[80, 81]
	Description: $\mathbf{c}_t^{kw}$ is the keyword-aware attention at time step t, $\mathbf{k}_j$ denotes the embedding of j-th extracted/assigned keyword, K is the set of all extracted/assigned keywords.		
	$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^{kwg})$	$\mathbf{c}_t^{kwg} = \sum_{u\in\mathcal{U}_{kwg}} \alpha(u)\ \mathbf{u},\ \alpha(u) = \frac{\exp(\eta(\mathbf{s}_{t-1}, \mathbf{u}))}{\sum_{u'\in\mathcal{U}_{kwg}}\exp(\eta(\mathbf{s}_{t-1}, \mathbf{u}'))}$	[84]
	Description: $\mathbf{c}_t^{kwg}$ is the keyword graph aware attention at time step t; $\mathcal{U}_{kwg}$ is the set of nodes in keyword graph; $\mathbf{u}$ is the node embedding (often obtained from graph neural networks).		
Knowledge Base	$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^{kb})$	$\mathbf{c}_t^{kb} = \sum_{j=1}^{\|K\|} \alpha_{tj}\mathbf{k}_j,\ \alpha_{tj} = \frac{\exp(\eta(\mathbf{s}_{t-1}, (\mathbf{k}_j^{(s)}\oplus\mathbf{k}_j^{(p)}\oplus\mathbf{k}_j^{(o)})))}{\sum_{j'=1}^{\|K\|}\exp(\eta(\mathbf{s}_{t-1}, (\mathbf{k}_{j'}^{(s)}\oplus\mathbf{k}_{j'}^{(p)}\oplus\mathbf{k}_{j'}^{(o)})))}$	[44, 60, 144]
	Description: $\mathbf{c}_t^{kb}$ is the knowledge base attention at time step t; $\mathbf{k}_j$ denotes the embedding of j-th knowledge triple obtained from a KB where $\mathbf{k}_j = (\mathbf{k}_j^{(s)} \oplus \mathbf{k}_j^{(p)} \oplus \mathbf{k}_j^{(o)})$; K is the set of all retrieved knowledge triples.		
Knowledge Graph	$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^{kg})$	$\mathbf{c}_t^{kg} = \sum_{u\in\mathcal{U}_{sub}} \alpha(u)\mathbf{u},\ \alpha(u) = \frac{\exp(\eta(\mathbf{s}_{t-1}, \mathbf{u}))}{\sum_{u'\in\mathcal{U}_{sub}}\exp(\eta(\mathbf{s}_{t-1}, \mathbf{u}'))}$	[65, 173, 183]
	Description: $\mathbf{c}_t^{kg}$ is the knowledge graph attention at time step t; $\mathbf{u}$ is the node embedding (often obtained from graph neural networks); $\mathcal{U}_{sub}$ is the sub-graph determined by a specific input sequence.		
	$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^{ke})$	$\mathbf{c}_t^{ke} = \sum_{u\in\mathcal{U}_{sub}} \sum_{(u_i,r,u_j)\in\mathcal{N}(u)} \alpha(u)\beta(u_i,r,u_j)\ \vartheta(\mathbf{u}_i,\mathbf{r},\mathbf{u}_j)$, $\beta(u_i,r,u_j) = \frac{\exp(\eta(\mathbf{s}_{t-1}, \vartheta(\mathbf{u}_i,\mathbf{r},\mathbf{u}_j)))}{\sum_{(u_{i'},r,u_{j'})\in\mathcal{N}(u)}\exp(\eta(\mathbf{s}_{t-1}, \vartheta(\mathbf{u}_{i'},\mathbf{r},\mathbf{u}_{j'})))}$	[56, 180]
	Description: $\mathbf{c}_t^{ke}$ is the knowledge edge attention at time step t; based on knowledge graph attention, it further calculates attention on each knowledge edge; $\vartheta(\cdot)$ learns embedding of knowledge edges, which can be $\vartheta(\mathbf{u}_i,\mathbf{r},\mathbf{u}_j) = f_{mlp}(\text{TransE}(\mathbf{u}_i,\mathbf{r},\mathbf{u}_j))$ [180] and $\vartheta(\mathbf{u}_i,\mathbf{r},\mathbf{u}_j) = (\mathbf{W}_r\mathbf{r})^\top \tanh(\mathbf{W}_i\mathbf{u}_i + \mathbf{W}_j\mathbf{u}_j)$ [56].		
Ground Text	$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^{gt})$	$\mathbf{c}_t^{gt} = \sum_{j=1}^{n} \alpha_{tj}\mathbf{h}_j,\ \alpha_{tj} = \frac{\exp(\eta(\mathbf{s}_{t-1}, \mathbf{h}_j))}{\sum_{j'=1}^{n}\exp(\eta(\mathbf{s}_{t-1}, \mathbf{h}_{j'}))}$	[121, 169]
	Description: $\mathbf{c}_t^{gt}$ is the grounded text attention at time step t; $\mathbf{h}_j$ denotes the contextual embedding (e.g., LSTM hidden state) of j-th word in grounded document; n is the length of grounded document.		
	$\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^{rc})$	$\mathbf{c}_t^{rc} = \sum_{j=1}^{n} \alpha_{tj}\mathbf{h}_j^{rc},\ \alpha_{tj} = \frac{\exp(\eta(\mathbf{s}_{t-1}, \mathbf{h}_j^{rc}))}{\sum_{j'=1}^{n}\exp(\eta(\mathbf{s}_{t-1}, \mathbf{h}_{j'}^{rc}))}$	[11, 96]
	Description: $\mathbf{c}_t^{rc}$ is the grounded text attention learnt from reading background document at time step t; $\mathbf{h}_j^{rc}$ denotes the context-aware background representation, i.e., $\mathbf{h}_i^{rc} = \text{RC}(X, B)[i]$ where $\text{RC}(\cdot,\cdot)$ is a reading comprehension model; n is the length of grounded document.		

attention [80, 81, 84], topic attention [90, 152, 158, 174], knowledge base attention [44, 60, 144], knowledge graph attention [65, 75, 173, 183], grounded text attention [11, 96, 121, 169] and etc.

2.2.2 Copy and Pointing Mechanisms. Copy and pointing mechanisms are used to choose subsequences in the input sequence and put them at proper places in the output sequence.

CopyNet [54]. CopyNet has a differentiable network architecture. It can be easily trained in an end-to-end manner [54]. In CopyNet, the probability of generating a target token is a combination

of the probabilities of two modes, generate-mode and copy-mode. First, CopyNet represents unique tokens in the global vocabulary $\mathcal{V}$ and the vocabulary of source sequence $\mathcal{V}_X$. It builds an extended vocabulary $\mathcal{V}_{\text{ext}} = \mathcal{V} \cup \mathcal{V}_X \cup \{\text{unk}\}$. Then, the distribution over the extended vocabulary is

$$p(y_t) = p_g(y_t) + p_c(y_t), \tag{10}$$

where $p_g(\cdot|\cdot)$ and $p_c(\cdot|\cdot)$ stand for the probability of generate-mode and copy-mode. It is given by

$$p_g(y_t) = \begin{cases} \frac{1}{Z} \exp \psi_g(y_t), & y_t \in \mathcal{V} \cup \{\text{unk}\}, \\ 0, & \text{otherwise}; \end{cases} \tag{11}$$

$$p_c(y_t) = \begin{cases} \frac{1}{Z} \sum_{j:x_j=y_t} \exp \psi_c(x_j), & y_t \in \mathcal{V}_X, \\ 0, & \text{otherwise}; \end{cases} \tag{12}$$

where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are score functions for generate-mode and copy-mode [54], and Z is the normalization term shared by the two modes, i.e., $Z = \sum_{v \in \mathcal{V} \cup \{\text{unk}\}} \exp \psi_g(v) + \sum_{x \in \mathcal{V}_X} \exp \psi_c(x)$, The score of each mode is calculated by

$$\psi_g(y_t = v_i) = \mathbf{v}_i^\top \mathbf{W}_g \mathbf{s}_t, \ v_i \in \mathcal{V} \cup \{\text{unk}\}, \tag{13}$$

$$\psi_c(y_t = x_j) = \mathbf{h}_j^\top \mathbf{W}_c \mathbf{s}_t, \ x_j \in \mathcal{V}_X, \tag{14}$$

where $\mathbf{v}_i$ is the one-hot indicator vector for v_i, where $\psi(\cdot)$ can be alternated by other two forms [92].

Pointer-Generator Network [123]. Similar to CopyNet, Pointer-generator network (PGN) has a differentiable network architecture [123]. Differently, PGN explicitly calculates a switch probability p_m between generate-mode and copy-mode. It recycles the attention distribution to serve as the copy distribution. The vocabulary distribution over $\mathcal{V}_{\text{ext}}$ is calculated by

$$p(y_t) = p_m(\text{g}) \cdot p_g(y_t) + (1 - p_m(\text{g})) \cdot p_c(y_t), \tag{15}$$

where $p_m(\text{g})$ indicates the probability of choosing generate-mode, which is obtained by

$$p_m(\text{g}) = \text{sigmoid}(\mathbf{W}_h \cdot \sum_{j=1}^{n} \alpha_{tj} \mathbf{h}_j + \mathbf{W}_s \cdot \mathbf{s}_t + \mathbf{W}_y \cdot \mathbf{e}(y_{t-1})), \tag{16}$$

where α_{tj} is given in Eq.(9). Vocabulary distribution $P_g(y_t)$ and attention distribution $P_c(y_t)$ are

$$p_g(y_t) = \begin{cases} \frac{1}{Z} \psi_g(y_t), & y_t \in \mathcal{V} \cup \{\text{unk}\}, \\ 0, & \text{otherwise}; \end{cases} \tag{17}$$

$$p_c(y_t) = \begin{cases} \frac{1}{Z} \sum_{j:x_j=y_t} \alpha_{tj}, & y_t \in \mathcal{V}_X, \\ 0, & \text{otherwise}. \end{cases} \tag{18}$$

Note that the mechanisms in CopyNet and PGN can be viewed as a soft switch that chooses between generation and copy modes. They are different from hard-switch methods [57]. Importantly, CopyNet and PGN have been used as the *base model* for a lot of knowledge-enhanced NLG work.

Knowledge-related mode. A knowledge-related mode choose subsequences in the obtained knowledge and put them at proper places in the output sequence. It helps NLG models to generate words that are not included in the global vocabulary ($\mathcal{V}$) and input sequence ($\mathcal{V}_X$). For example, by adding knowledge base-mode, the extended vocabulary ($\mathcal{V}_{ext}$) also includes entities and relations appearing in knowledge base, i.e., $\mathcal{V}_{ext} = \mathcal{V} + \mathcal{V}_X + \mathcal{V}_{KB}$. The probability of generating a target token is a combination of the probabilities of three modes: generate-mode, copy-mode and knowledge base-mode. Therefore, knowledge-related mode is not only capable of the regular generation of words but also the operation of producing appropriate subsequences in different knowledge sources. Table 4 summarizes different kinds of knowledge-related mode proposed by recent papers,

Table 4. Natural language generation methods that incorporates knowledge-related mode.

Sources	Generation mode	Scoring function	Papers
Topic	$p_{tp}(y_t) = \begin{cases} \frac{1}{Z}\sum_{j:p_j=y_t} \exp \psi_{tp}(y_t), & y_t \in \mathcal{V}_P, \\ 0, & \text{otherwise}, \end{cases}$	$\psi_{tp}(y_t = p_j) = f_{mlp}(\mathbf{s}_t, \mathbf{e}(y_{t-1}), \mathbf{c}_t))$	[146, 158]
	Description: the score of copying a word from topics is calculated by a MLP function; In [146, 158], $f_{mlp}(\mathbf{s}_t, \mathbf{e}(y_{t-1}, \mathbf{c}_t)) = \sigma(\mathbf{v}_j^\top(\mathbf{W}_s\mathbf{s}_t + \mathbf{W}_y\mathbf{e}(y_{t-1}) + \mathbf{W}_c\mathbf{c}_t))$ where $\mathbf{v}_j$ is the one-hot indicator vector for topic p_j; Z is a normalization term shared by the generation mode and topic mode; $\mathcal{V}_P$ is the set of all topic words.		
Keyword	$p_{kw}(y_t) = \begin{cases} \frac{1}{Z}\sum_{j:k_j=y_t} \exp \psi_{kw}(y_t), & y_t \in \mathcal{V}_K, \\ 0, & \text{otherwise}. \end{cases}$	$\psi_{kw}(y_t = k_j) = \alpha_{tj}$	[81]
	Description: the score of copying a keyword is the same as keyword attention score as mentioned in Table 3; Z is a normalization term shared by the generation mode and keyword mode; $\mathcal{V}_K$ is the set of all keywords.		
Knowledge Base	$p_{kb}(y_t) = \begin{cases} \frac{1}{Z}\sum_{j:k_j^{(o)}=y_t} \exp \psi_{kb}(y_t), & y_t \in \mathcal{V}_K \\ 0, & \text{otherwise} \end{cases}$	$\psi_{kb}(y_t = k_j^{(o)}) = f_{mlp}(\mathbf{k}_j, \mathbf{e}(y_{t-1}), \mathbf{H}_{KB})$	[60]
	Description: the score of copying a word from knowledge base is calculated by a MLP function; In [60], $f_{mlp}(\mathbf{k}_j, \mathbf{e}(y_{t-1}), \mathbf{H}_{KB}) = \sigma(\mathbf{W}_k\mathbf{k}_j + \mathbf{W}_y\mathbf{e}(y_{t-1}) + \mathbf{W}_h\mathbf{H}_{KB})$ where $\mathbf{H}_{KB}$ is an accumulated vector which record the attentive history for each knowledge triple in candidate knowledge triples. Besides, $\mathcal{V}_K$ is the set of all objects in knowledge triples, i.e., $\mathcal{V}_K = \{k_j^{(o)}\}_{j=1}^{|K|}$. This is because the subjects $\{k_j^{(s)}\}_{j=1}^{|K|}$ are contained in input sequences and used to retrieve knowledge triples, so they do not have to be included in $\mathcal{V}_K$.		
Knowledge Graph	$p_{kg}(y_t) = \begin{cases} \frac{1}{Z}\sum_{j:u_j=y_t} \exp \psi_{kg}(y_t), & y_t \in \mathcal{U}_{sub} \\ 0, & \text{otherwise} \end{cases}$	$\psi_{kg}(y_t = u_j) = \alpha(u_i)\beta(u_i, r, u_j)$	[180]
	Description: the score of copying a node entity from knowledge graph is calculated by a hierarchical attention as mentioned in Table 3; $\mathcal{U}_{sub}$ is the subset of nodes (entities) in KG determined by a specific input sequence.		
	$p_{kg}(y_t) = \begin{cases} \frac{1}{Z}\sum_{j:u_j=y_t} \exp \psi_{kg}(y_t), & y_t \in \mathcal{U}_{sub} \\ 0, & \text{otherwise} \end{cases}$	$\psi_{kg}(y_t = u_j) = \sigma(\mathbf{s}_t \cdot \mathbf{u}_j)$	[173]
	Description: the score of copying a node entity is calculated by a bi-linear model; $\sigma(\cdot)$ is an activation function.		
Grounded Text	$p_{gt}(y_t) = \begin{cases} \frac{1}{Z}\sum_{j:h_j=y_t} \exp \psi_{gt}(y_t), & y_t \in \mathcal{V}_T, \\ 0, & \text{otherwise}. \end{cases}$	$\psi_{gt}(y_t = h_j) = \alpha_{tj}$	[96, 169]
	Description: the score of copying a word in grounded text is the same as grounded text attention score as mentioned in Table 3; $\mathcal{V}_T$ is the set of all words in grounded text (e.g., retrieved text snippets, background document).		

such as topic mode [146, 158], keyword mode [81], knowledge base mode [60], knowledge graph mode [173, 180], background mode [96, 169] and etc.

2.2.3 Memory Network. Memory networks (MemNNs) are recurrent attention models over a possibly large external memory [134]. They write external memories into several embedding matrices, and use query (generally speaking, the input sequence X) vectors to read memories repeatedly. This approach can encode long dialog history and memorize external information.

Given an input set $\{m_1, \cdots, m_i\}$ to be stored in memory. The memories of MemNN are represented by a set of trainable embedding matrices $\mathbf{C} = \{\mathbf{C}^1, \cdots, \mathbf{C}^{K+1}\}$, where each $\mathbf{C}^k$ maps tokens to vectors, and a query (i.e., input sequence) vector $\mathbf{h}_X^k$ is used as a reading head. The model loops over K hops and it computes the attention weights at hop k for each memory m_i using:

$$\mathbf{p}_i^k = \text{softmax}((\mathbf{h}_X^k)^\top \mathbf{C}_i^k), \tag{19}$$

where $\mathbf{C}_i^k = \mathbf{C}^k(m_i)$ is the memory content in i-th position, i.e., mapping m_i into a memory vector. Here, $\mathbf{p}^k$ is a soft memory selector that decides the memory relevance with respect to the query vector $\mathbf{h}_X^k$. Then, the model reads out the memory $\mathbf{o}^k$ by the weighted sum over $\mathbf{C}^{k+1}$,

$$\mathbf{o}^k = \sum_i \mathbf{p}_i^k \mathbf{C}_i^{k+1}. \tag{20}$$

Then, the query vector is updated for the next hop by using $\mathbf{h}_X^{k+1} = \mathbf{h}_X^k + \mathbf{o}^k$. The result from the encoding step is the memory vector $\mathbf{o}^K$, which will become the input for the decoding step.

Knowledge-related memory. Memory augmented encoder-decoder framework has achieved promising progress for many NLG tasks. For example, MemNN is widely used for encoding dialogue history in task-oriented dialogue systems [20, 120, 153]. Such frameworks enable a decoder to retrieve information from a memory during generation. Recent work explored to model external knowledge with memory network such as knowledge base [89, 93, 167] and topic [44, 179].

2.2.4 Graph Network. Graph network captures the dependence of graphs via message passing between the nodes of graphs. Recent advances of graph neural networks (GNNs) [156] and graph-to-sequence (Graph2Seq) [8] potentiate to bridge up the gap between graph representation learning and text generation. Knowledge graph, keyword graph, dependency graph and other graph structures can be integrated into encoder-decoder frameworks through various GNN algorithms. Here we denote a graph as $\mathcal{G} = (\mathcal{U}, \mathcal{E})$, where $\mathcal{U}$ is the set of entity nodes and $\mathcal{E}$ is the set of (typed) edges. Modern GNNs typically follow a neighborhood aggregation approach, which iteratively updates the representation of a node by aggregating information from its neighboring nodes and edges [163]. After k iterations of aggregation, a node representation can capture the structural information within its k-hop neighborhood. Formally, the k-th layer of a node $u \in \mathcal{U}$ is:

$$\mathbf{u}^{(k)} = \textsc{Combine}_k(\mathbf{u}^{(k-1)}, \textsc{Aggregate}_k(\{(\mathbf{u}_i^{(k-1)}, \mathbf{e}_{ij}^{(k-1)}, \mathbf{u}_j^{(k-1)}) : \forall (u_i, e_{ij}, u_j) \in \mathcal{N}(u)\})), \quad (21)$$

where $\mathcal{N}(u) = \{(u_i, e_{ij}, u_j) \in \mathcal{E} | u_i = u \text{ or } u_j = u\}$ denotes the set of edges containing node u, $\mathbf{u}^{(k)}$ and $\mathbf{e}_{ij}^{(k)}$ are feature vectors of a node u and the edge between u_i and u_j at the k-th iteration/layer. The choice of $\textsc{Aggregate}(\cdot)$ and $\textsc{Combine}(\cdot)$ in GNNs is crucial. A number of architectures for $\textsc{Aggregate}(\cdot)$ have been proposed in different GNN works such as GAT [140]. Meanwhile, the $\textsc{Aggregate}(\cdot)$ function used in labeled graphs (e.g., a knowledge graph) is often taken as those GNNs for modeling relational graphs [53, 122], such as a relational GAT [180]:

$$\mathbf{u}^{(k)} = \sigma\Big(\sum_{(u_i, e_{ij}, u_j) \in \mathcal{N}(u)} \alpha(u_i, e_{ij}, u_j) \cdot (\mathbf{u}_i^{(k-1)} \oplus \mathbf{u}_j^{(k-1)})\Big), \quad (22)$$

$$\alpha(u_i, e_{ij}, u_j) = \frac{\exp(\eta(\mathbf{u}_i, \mathbf{e}_{ij}, \mathbf{u}_j))}{\sum_{(u_{i'}, e_{ij}, u_{j'}) \in \mathcal{N}(u)} \exp(\eta(\mathbf{u}_{i'}, \mathbf{e}_{ij}, \mathbf{u}_{j'}))}, \quad (23)$$

$$\eta(u_i, e_{ij}, u_j) = (\mathbf{W}_e \cdot \mathbf{e}_{ij}^{(k-1)})^\top \tanh(\mathbf{W}_i \cdot \mathbf{u}_i^{(k-1)} + \mathbf{W}_j \cdot \mathbf{u}_j^{(k-1)}), \quad (24)$$

where $\sigma(\cdot)$ denotes a nonlinear activation function (often taken as LeakyReLU), and $\mathbf{W}_i, \mathbf{W}_j, \mathbf{W}_e$ are all trainable weight matrices. In practical, K is usually set as $K = 2$ [65, 173, 183] or $K = 1$ [56, 84, 180] (see performance comparisons with different K in Figure 3 of [5]). Graph attention mechanism could be multi-head attention [65]. The attention weight measures the association of a relation e_{ij} between two entity nodes u_i and u_j. To obtain the representation of graph $\mathcal{G}$ (denoted as $\mathbf{h}_G$), the $\textsc{Readout}$ function pools node features from the final iteration K,

$$\mathbf{h}_G = \textsc{Readout}(\{\mathbf{u}^{(K)}, u \in \mathcal{U}\}), \quad (25)$$

where $\textsc{Readout}(\cdot)$ can be a simple permutation invariant function such as averaging or a more sophisticated graph-level pooling function [171].

Applications. Graph network has been commonly used in integrating different knowledge of graph structure, such as knowledge graph and dependency graph. Graph attention network [140] has demonstrated its effectiveness in many NLG tasks since it can be easily combined with sequence attention and jointly optimized [173, 180]. We will introduce different graph structure knowledge in subsequent sections such as knowledge graph (Section 4.2), internal knowledge graph (Section 4.3.1), dependency graph (Section 4.3.2-4.3.3) and keyword/sentence graph (Section 4.3.4).

2.2.5 Pre-trained Models. Given the limited size of the most NLG datasets and relative the large number of parameters, it's hard for the deep neural network to generalize well in the specific tasks. However, since large-scale unlabeled datasets are easy to obtain as well as the success of the pre-trained model in the computer vision (CV) domain, it's natural to consider utilizing a pre-trained model for the NLG tasks which can provide better model initialization and regularization [117]. The first-generation pre-trained models aim to learn good non-contextual embedding (i.e., word embedding), such as Word2Vec [99] and GloVe [111]. The second-generation pre-trained models focus on learning contextual word embeddings, such as GPT [118] and BERT [34].

Non-contextual embedding. They map discrete language symbols into a distributed embedding space. Commonly used non-contextual embeddings such as Word2Vec [99] and GloVe [111] can capture semantic meanings of words and provide initialization for generation tasks. However, there are two major limitations. First, the embeddings are static. The embedding for a word is always the same irrespective of its context, which fails to model polysemous words and capture context information. Second, out-of vocabulary words cannot be mapped with an embedding.

Contextual embedding. To address the issue of polysemous and the context-dependent nature of words, contextualized pre-trained models on the large corpus benefit downstream NLG tasks by providing high quality contextualized language representations. Given a series of tokens $X = \{x_1, x_2, ..., x_n\}$ where $x_i \in \mathcal{V}_X$ is a word or sub-word, the contextual representation of $\mathbf{x}_i$ depends on the whole text, i.e., $(\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n) = f_{context}(x_1, x_2, ..., x_n)$ where $f_{context}(\cdot)$ is a neural network such as a deep long-short term memory (LSTM) or a Transformer [139]. For examples, ELMo [112] uses deep bidirectional LSTM pre-trained on large corpora; OpenAI's GPT [118] uses Transformer decoder instead of multi-layer LSTM. One constraint of GPT is that it only uses unidirectional information. Therefore, BERT [34] utilized the masked language model (MLM) to overcome the drawbacks in training unidirectional language model. BERT was further extended to many other encoder-decoder architectures with MLM such as MASS [130] and BART [79].

Applications. Pre-trained models have been widely used in text generation systems, such as GloVe [26, 56], BERT [27], GPT [33, 115]. Recent papers explored to transfer commonsense knowledge into pre-trained language models by utilizing triple information in knowledge bases [9, 55].

2.3 Knowledge-enhanced Learning and Inference

Besides the choice of model architectures, another common way of injecting knowledge information to the model is through the learning and inference processes. For example, one can encode knowledge into the objective function that guides the model training to acquire desired model behaviors. Such approaches enjoy the flexibility of integrating diverse types of knowledge by expressing the knowledge into certain forms of losses or constraints. In general, knowledge-enhanced learning and inference is agnostic to the model architecture, and can be combined with the various architectures mentioned above. On the other hand, some specialized learning and inference processes could require specific model architectures (e.g., specific models conditioning on knowledge).

2.3.1 Learning with knowledge-related tasks. One could devise learning tasks informed by the knowledge so that the model is trained to acquire the knowledge information.

Knowledge as the target. The first category of knowledge-related tasks creates learning targets based on the knowledge, and the model is trained to recover the targets. The knowledge-related tasks can be different from the standard text generation task of interest, and be combined as auxiliary to the text generation task, resulting in a *multi-task learning* setting. For example, knowledge loss in [36, 74] is defined as the cross entropy between the predicted and true knowledge sentences, and is combined with the standard conversation generation loss to enhance grounded conversation. Similar tasks include keyword extraction loss [81], template re-ranking loss [18, 145], mode loss [155, 180], BOW loss [85, 164], etc. Alternatively, one can directly derive the text generation targets from the knowledge, and use those (typically noisy) targets as supervisions in the standard text generation task. The approach is called *weakly-supervised learning*. Weakly-supervised learning enforces the relevancy between the knowledge and the target sequence. For example, in the problem of aspect-based summarization, the work [136] automatically creates target summaries based on external knowledge bases, which are used to train the summarization model in a supervised manner.

Knowledge as the condition. The second common way of devising knowledge-related tasks is to augment the original text generation task by additionally conditioning the generation on the knowledge. That is, the goal is to learn a function $p_\theta(Y|X, K)$, where X is the input sequence, Y is the target text and K is the knowledge. Generally, the knowledge K is first given externally (e.g., style, emotion) or retrieved from external resources (e.g., facts from knowledge base, a document from Wikipedia) or extracted from the given input text (e.g., keywords, topic words). Second, a conditional text generation model is used to incorporate knowledge and generate target output sequence. In practice, knowledge is often remedied by soft enforcing algorithms such as attention mechanism [4] and copy/pointing mechanism [54, 123]. Regarding knowledge as condition is widely used in knowledge-enhanced text generation. For examples, work has been done in making personalized dialogue response by taking account of persona [176] and emotion [179], controlling various aspects of the response such as politeness [107], grounding the responses in external source of knowledge [36, 52, 180], generating topic-coherent sequence [137, 164] and so on.

2.3.2 Learning with knowledge constraints. Instead of creating standalone tasks (training objectives) that encapsulate knowledge, the second paradigm of knowledge-enhanced learning is to treat knowledge as the *constraints* to regularize the original text generation training objective.

The posterior regularization (PR) framework was proposed to restrict the space of the model posterior on unlabeled data as a way to guide the model towards desired behavior [45, 62, 184]. PR has been used as one of the principled frameworks to impose knowledge constraints on probabilistic models (including deep networks) in general, and has demonstrated effectiveness in regulating the learning of models in language generation [64, 175]. PR augments any regular training objective $\mathcal{L}(\theta)$ (e.g., negative log-likelihood, as in Eq.(7)) with a constraint term to encode relevant knowledge. Formally, denote the constraint function as $g(X, Y) \in \mathbb{R}$ such that a higher $g(X, Y)$ value indicates a better generated sequence Y in terms of the particular knowledge. To incorporate the constraint, PR introduces an auxiliary distribution $q(Y|X)$, and imposes the constraint on q by encouraging a large expected g value: $\mathbb{E}q[g(X, Y)]$. Meanwhile, the model p_θ is encouraged to stay close to q through a KL divergence term. The learning problem is thus a constrained optimization problem:

$$\max_{\theta,q} \mathcal{L}(\theta) - \mathrm{KL}(q(Y|X)||p_\theta(Y|X)) + \xi \tag{26}$$

$$s.t.\ \mathbb{E}q[g(X, Y)] > \xi, \tag{27}$$

where ξ is the slack variable. The PR framework is also related to other constraint-driven learning methods [e.g., 19, 86, 94]. We refer readers to [45] for more discussions.

2.3.3 Inference with knowledge constraints. Pre-trained language models can leverage large amounts of unannotated data and a simple log-likelihood training objective. However, once such models are trained, controlling language generation with particular knowledge becomes difficult without modifying the model architecture to allow for external input knowledge or fine-tuning with specific data [33]. Plug and play language model (PPLM) opens up a new way to control language generation with particular knowledge during inference. At every generation step during inference, the PPLM shifts the history matrix in the direction of the sum of two gradients: one toward higher log-likelihood of the attribute a under the conditional attribute model $p(a|Y)$ and one toward higher log-likelihood of the unmodified pre-trained generation model $p(Y|X)$ (e.g., GPT). Specifically, the attribute model $p(a|Y)$ make gradient based updates to $\Delta\mathbf{S}_t$ as follows:

$$\Delta\mathbf{S}_t \leftarrow \Delta\mathbf{S}_t + \frac{\nabla_{\Delta\mathbf{S}_t} \log p(a|\mathbf{S}_t + \Delta\mathbf{S}_t)}{||\nabla_{\Delta\mathbf{S}_t} \log p(a|\mathbf{S}_t + \Delta\mathbf{S}_t)||^{\gamma}} \tag{28}$$

where γ is the scaling coefficient for the normalization term; $\Delta\mathbf{S}_t$ is update of history matrix $\mathbf{S}_t$ (see Eq.(6)) and initialized as zero. The update step is repeated multiple times. Subsequently, a forward pass through the generation model is performed to obtain the updated $\widetilde{\mathbf{S}}_{t+1}$ as $\widetilde{\mathbf{S}}_{t+1} = f_{de}((\mathbf{S}_t + \Delta\mathbf{S}_t), \mathbf{e}(y_t), \mathbf{H})$. The perturbed $\widetilde{\mathbf{S}}_{t+1}$ is then used to generate a new logit vector.

PPLMs have demonstrated great efficiency and flexibility in combination of differentiable attribute models to steer text generation. Recent work has followed its idea in other tasks [115].

3 NLG ENHANCED BY TOPIC, KEYWORD AND LINGUISTIC FEATURES

3.1 NLG Enhanced by Topic

Topic, which can be considered as a representative or compressed form of text, has been often used to maintain the semantic coherence and guide the NLG process. For example, text summarization requires the short-generated output to capture the topics of the long input document. Dialogue systems avoid generating trivial and digressive utterance. Topic modeling is a powerful tool for finding the high-level content of a document collection in the form of latent topics [12]. A classical topic model, Latent Dirichlet allocation (LDA), has been widely used for inferring a low dimensional representation that captures latent semantics of words and documents [12]. In LDA, each topic is defined as a distribution over words and each document as a mixture distribution over topics. LDA generates words in the documents from topic distribution of document and word distribution of topic. Recent advances of neural techniques open a new way of learning low dimensional representations of words from the tasks of word prediction and context prediction, making neural topic models become a popular choice of finding latent topics from text [17, 58].

Applications. Popular NLG tasks that need topics include dialogue systems [16, 158], text summarization [43, 105, 146, 149] and neural machine translation [152, 174]. In dialogue systems, a vanilla Seq2Seq often generates trivial or non-committal sentences of frequent words or phrases in the corpus [158]. For example, a chatbot may say *"I do not know"*, *"I see"* too often. Though these off-topic responses are safe to reply to many queries, they are boring with very little information. Such responses may quickly lead the conversation to an end, severely hurting user experience. In text summarization, the attention mechanism tries to align words between input and output sequences. However, the input sequence (i.e., original documents) is often much longer than the output sequence (i.e., summary). So, the attention mechanism can hardly ensure the coherence between the two sequences without the guidance from topics. In machine translation, though the input and output languages are different, the contents are the same, and globally, under the same

Table 5. Natural language generation methods that incorporate topic knowledge in text generation.

Task	Method	Description	
Dialogue system	Topic-Seq2Seq [158]	RNN Seq2Seq + LDA topics	M1
	PEE [164]	RNN Seq2Seq + MLP topics	M3
Machine translation	Topic-NMT [174]	RNN Seq2Seq + LDA topics	M1
	BLT-NMT [152]	RNN Seq2Seq + CNNs latent topics	M2
Summarization	Topic-ConvS2S [105]	ConvS2S (CNN Seq2Seq) + LDA topics	M1
	RL-Topic-ConvS2S [146]	Topic-ConvS2S + Reinforcement Learning	M1
	TGVAE [149]	Variational RNN Seq2Seq + MLP topics	M3
	VHTM [43]	Variational RNN Seq2Seq + MLP topics	M3
Content manipulation	TATGM [83]	Variational RNN Seq2Seq + MLP topics	M3
Paraphrase	TGLM [46]	RNN Seq2Seq + CNNs latent topics	M2
	PTA-Seq2Seq [90]	RNN Seq2Seq + LDA topics + Regularization	M1

topic. Therefore, topic can serve as an auxiliary guidance to preserve the semantics information of input text in one language into the output text in the other language.

Table 5 summarizes some representative NLG models that are enhanced by knowledge from topics. They can be grouped into three categories of methodologies:

- **M1: Leverage topic words from generative topic models.** It first discovers topics using generative topic models (e.g., LDA), and then incorporate the topics representations into neural generation models through topic attention mechanism. The topic embeddings provide high level guidance on generating coherent output sequences.
- **M2: Jointly optimize generation model and CNN topic model.** It is to design an end-to-end neural framework that simultaneously learns latent topic representations and generates output sequences. Convolutional neural networks (CNN) are often used to generate the latent topics through iterative convolution and pooling operations.
- **M3: Enhance NLG by neural topic models with variational inference.** Neural topic models have relatively more parameters than conventional topic models, so they suffer from the issue of overfitting. Variational inference may alleviate this issue by assuming a prior distribution (e.g. Gaussian) in the latent space of topics. Besides, the neural property of topic models enables back propagation for joint optimization, contributing to more coherent topics.

3.1.1 Leverage Topic Words from Generative Topic Models. Topics help understand the semantic meaning of sentences. It determines the semantic spectrum to a certain extent. For example, in human-human conversations, people might first select topically related concepts (e.g., sport, food) in their minds, then organize content and select words to respond. Thus, an effective solution to NLG tasks is to first discover topics using generative topic models (e.g., LDA), and then incorporate the topics representations into neural generation models. In existing work, there are two mainstream methods to represent topics obtained from generative topic models. The first way is to use the generated topic distributions for each word (i.e., word distributions over topics) in the input sequence [105, 174]. The second way is to assign a specific topic to the input sequence, then picks the top-k words with the highest probabilities under the topic, and use word embeddings (e.g., GloVe) to represent topic words [90, 146, 158]. Explicitly making use of topic words can bring stronger guidance than topic distributions, but the guidance may deviate from the target output sequence when some generated topic words are irrelevant.

Zhang et al. proposed a topic-informed Seq2Seq model by concatenating the topic distributions with encoder and decoder hidden states [174]. Xing et al. designed a topic-aware Seq2Seq model (called "Topic-Seq2Seq") in order to use topic words as prior knowledge to help dialogue generation [158]. Specifically, topic words are obtained from a pre-trained LDA model and represented as $\{\mathbf{p}_i\}_{i=1}^{k}$. At the decoding phase, each word is generated according to both the input sequence and topic words through a joint attention mechanism. It summarizes context vectors from input sequence attention ($\mathbf{c}_t$) and topic attention ($\mathbf{c}_t^{tp}$) simultaneously. Then, the decoder updates it hidden state $\mathbf{s}_t$ by taking above two context vectors, i.e., $\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{c}_t, \mathbf{c}_t^{tp}, \mathbf{e}(y_{t-1}))$, where $\mathbf{c}_t$ is the decoder context state as introduced in Eq.(9) and $\mathbf{c}_t^{tp}$ calculates attentions over each topic word $\mathbf{p}_i$ (details in Table 3). The probability of generating a target word is a combination of the probabilities of two modes: generate-mode and topic-mode (details in Table 4).

Follow-up work. Topic-Seq2Seq used LDA to extract topics that help dialogue systems improve their consistency and topicality, which has been further extended by many studies [90, 105, 146]. Liu et al. added two penalty terms to directly supervise the importance assigned to the topic and the selection of topic words [90]. Combining the two types of regularization has been shown more efficient than mere supervision from the generation loss. Moreover, Wang et al. proposed a topic-aware ConvS2S (short for convolutional sequence-to-sequence) [146]. ConvS2S can be stacked to represent long input sequence and find the long distant dependency among words. It can avoid gradient vanishing issue caused by an RNN-Seq2Seq model. Besides, they used reinforcement training to directly optimize the proposed model with respect to the non-differentiable metric ROUGE, which can avoid the exposure bias during inference. Narayan et al. proposed to concatenate word embeddings and topic distribution vectors (at both word-level and document-level) together as encoder inputs, thus, the textual knowledge embedded in the input document and the topical knowledge would be attended together to enhance the decoding phase of summarization [105].

3.1.2 Jointly Optimize Generation Model and CNN Topic Model. LDA models are usually unsupervised with an assumption that the word distributions of topics are Dirichlet distributions. However, LDA models may not be able to find proper topics that the target task (e.g., NLG) requires. Also, the LDA models were separated from the training process of neural generation model and were not able to adapt to the diversity of dependencies between input and output sequences. Convolutional neural networks were used to learn latent topic representations through iterative convolution and pooling operations. There are growing interests of using the CNNs to map latent topics implicitly into topic vectors that can be used to enhance NLG tasks [46, 152]. Empirical analyses showed that convolution-based topic extractors could outperform LDA-based topic models for multiple applications (e.g., conversation/dialogue system, text summarization). However, theoretical analysis was missing to ensure the quality of the topics captured by the convolutions. And their interpretability is not as satisfactory as the LDA-based topic models.

3.1.3 Enhance NLG by Neural Topic Models with Variational Inference. Neural topic models combine the advantages of neural networks and probabilistic topic models [17, 58]. They can be trained efficiently by backpropagation, scaled to large data sets, and easily adapted to any available contextual information. In topic models, documents have a multinomial distribution over topics and topics have a multinomial distribution over words [12]. Cao et al. connected this view of topic models with neural networks by embedding the relationship with differentiable functions [17]. In order to facilitate efficient inference, Dirichlet distributions can be employed as the prior to generate the parameters of the multinomial distribution θ_d for each document [97]. The generative process of LDA is represented as: (1) $\theta_d \sim \text{Dirichlet}(\alpha)$; (2) $t_i \sim \text{Multinomial}(\theta_d)$; (3) $w_i \sim \text{Multinomial}(\beta_{t_i})$, where d denotes the bag-of-words representation of a document, t_i represents the topic assignment

for word w_i, and β_{t_i} represents the topic distribution over words given topic assignment t_i. The marginal likelihood for document d is:

$$p(d|\alpha,\beta) = \int_\theta p(\theta|\alpha)\big(\prod_{i=1}^{n}\sum_{t_i} p(w_i|\beta_{t_i})p(t_i|\theta)\big)d\theta = \int_\theta p(\theta|\alpha)p(d|\beta,\theta)d\theta. \tag{29}$$

However, a directed generative model comes up against the problem of establishing low variance gradient estimators. Miao et al. parameterized the multinomial distributions with neural networks and jointly learned the model parameters via variational inference [97]. They created neural structures for constructing topic distributions conditioned on a draw from a multivariate Gaussian distribution, represented as $\theta_d \sim \mathrm{G}(\mu_0, \sigma_0^2)$, where $\mathrm{G}(\mu_0, \sigma_0^2)$ is composed of a neural network conditioned on an isotropic Gaussian $\mathrm{N}(\mu_0, \sigma_0^2)$. Now, the marginal likelihood for document d is:

$$p(d|\beta,\mu_0,\sigma_0^2) = \int_\theta p(\theta|\mu_0,\sigma_0^2)\big(\prod_{i=1}^{n}\sum_{t_i} p(w_i|\beta_{t_i})p(t_i|\theta)\big)d\theta = \int_\theta p(\theta|\mu_0,\sigma_0^2)p(d|\beta,\theta)d\theta. \tag{30}$$

Compared with Eq.(29), the latent variable θ is parameterized by a neural network conditioned on a draw from a Gaussian distribution. To carry out neural variational inference, Miao et al. constructed an inference network $q(\theta|\mu(d),\sigma(d))$ to approximate the posterior $p(\theta|d)$, where $\mu(d)$ and $\sigma^2(d)$ are functions of d. The network was implemented by a multi-layer perceptron (MLP) [97, 98]. Taking a Gaussian prior distribution makes re-parameterization feasible to build an unbiased and low-variance gradient estimator for the variational distribution [35]. Without conjugacy prior, the updates of the parameters are derived directly and easily from the variational lower bound. Formally, a variational lower bound (aka., evidence lower bound (ELBO)) for the document log-likelihood is constructed as:

$$\mathcal{J}_{topic} = \mathbb{E}_{q(\theta|d)}[\log p(d|\beta,\theta)] - \mathrm{KL}(q(\theta|d)||p(\theta|\mu_0,\sigma_0^2)), \tag{31}$$

where $q(\theta|d)$ is the variational distribution approximating the true posterior $p(\theta|d)$. Its lower bound is estimate by sampling θ from $q(\theta|d) = \mathrm{G}(\theta|\mu(d),\sigma^2(d))$.

In order to combine neural topic model and neural generation model, the idea is to use the Variational Auto-Encoder (VAE) [35]. It adopts autoregressive networks (e.g., LSTM) both as the encoder and decoder. VAE can learn latent codes z of texts by reconstructing texts with its decoder. It assumes that the generation process is controlled by codes in a continuous latent space. This kind of VAE implementation considers sequential information of texts that can model the linguistic structure of texts. Wang et al. proposed topic guided variational autoencoder (TGVAE), to draw latent code z from a topic-dependent Gaussian Mixture Prior in order to incorporate the topical knowledge into latent variables [149]. The topic-dependent Gaussian Mixture Model (GMM) is defined as:

$$p(z|\beta,t) = \sum_{i=1}^{T} t_i \mathrm{N}(\mu(\beta_i),\sigma^2(\beta_i)), \tag{32}$$

where T is the number of topics, $\mu(d)$ and $\sigma^2(d)$ are functions implemented by MLP. TGVAE uses bag-of-words as input and embeds an input document into a topic vector. The topic vector is then used to reconstruct the bag-of-words input, and the learned topic distribution over words is used to model a topic-dependent prior to generate an output sequence Y from conditioned on an input sequence X. Specifically, the joint marginal likelihood can be written as:

$$p(Y,d|X) = \int_\theta \int_z p(\theta)p(d|\beta,\theta)p(z|\beta,\theta)p(Y|X,z)d\theta dz. \tag{33}$$

To maximize the log-likelihood $\log p(Y, d|X)$, a variational objective function is constructed as:

$$\mathcal{J}_{seq2seq} = \mathbb{E}_{q(z|X)}[\log p(Y|X, z)] - \mathbb{E}_{q(\theta|d)}[\mathrm{KL}(q(z|X)||p(z|\beta, \theta))], \tag{34}$$

where $q(z|X)$ is variational distributions for z. The combined object function is given by:

$$\mathcal{J} = \mathcal{J}_{topic} + \mathcal{J}_{seq2seq}. \tag{35}$$

Follow-up work. Fu et al. extended TGVAE with a variational hierarchical topic-aware mechanism (VHTM) that incorporated topical knowledge into words embedding and paragraph attention for long document summarization [43]. The topic-related parts with different levels of granularities in long documents are positioned and extracted to generate more germane summaries. Li et al. addressed that TGVAE and VHTM did not separate the semantic and structure latent codes explicitly [83]. They proposed two separate autoencoders that comprises a topic modeling component capturing topical knowledge and a sequence modeling capturing structural features (e.g., style). Moreover, the encoder of the topic modeling is served as a discriminator to force the decoder of the sequence modeling to generate texts having the semantics as close to the original texts as possible. By separating the semantic and structural codes, the model can control structural and semantic information independently. One interesting application is content manipulation (CM), which aims to generate sequence with same structure (e.g., style) in different semantic spaces [41, 83, 150].

3.1.4 Pros and Cons Discussion in NLG Enhanced by Topic. Topic models (e.g., LDA) has a strict probabilistic explanation since the semantic representations of both words and documents are combined into a unified framework. Besides, topic models can be easily used and integrated into generation frameworks. For examples, topic words can be represented as word embeddings; topic embedding can be integrated into the decoding phase through topic attention. However, LDA models are separated from the training process of neural generation model, so they cannot adapt to the diversity of dependencies between input and output sequences. Neural topic models combine the advantages of neural networks and probabilistic topic models, they can be trained efficiently by backpropagation, scaled to large data sets. Generally, neural topic models can provide better topic coherence than LDAs [17, 83, 149]. However, neural variational approaches share a same drawback that topic distribution is assumed to be an isotropic Gaussian, which makes them incapable of modeling topic correlations. Existing neural topic models assume that the documents should be i.i.d to adopt VAE. In fact, the documents are composed of words, which tend to be correlated instead of completely independent. So, the correlations between documents are critical for topic modeling.

3.2 NLG Enhanced by Keywords

Keyword (aka., key phrase, key term) is often referred as a sequence of one or more words, providing a compact representation of the content of a document. The mainstream methods of keyword acquisition for documents can be divided into two categories [128]: keyword assignment and keyword extraction. Keyword assignment means that keywords are chosen from a controlled vocabulary of terms or predefined taxonomy. Keyword extraction selects the most representative words explicitly presented in the document, which is independent from any vocabulary. Keyword extraction techniques (e.g., TF-IDF , TextRank , PMI) have been widely used over decades. Many NLG tasks can benefit from incorporating such a condensed form of essential content in a document to maintain the semantic coherence and guide the generation process.

Applications. Dialogue system [126, 132, 162, 179] and summarization [80, 81, 84] are two mainstream tasks that have adopted keyword knowledge. In a dialogue system, keywords help enlighten and drive the generated responses to be informative since vanilla Seq2Seq model tends to generate universally relevant replies which carry little semantics (e.g., *"I don't know"*, *"Okay"*). Besides,

Table 6. NLG methods that enhance text generation by incorporating keyword with keyword assignment (M1) or keyword extraction (M2). For brevity, we abbreviate keyword extraction to "KExt", keyword embedding to "KEmb", keyword sequence encoder to "KSeqE", and keyword graph encoder to "KGraphE".

Task	Method	Description	
Dialogue system	Seq2BF [102]	Seq2Seq + KExt (PMI) + KEmb	M2
	E-SCBA [82]	Keyword assignment with 7 emotions	M1
	EmoChat [179]	E-SCBA + two memory modules	M1
	EmoDS [132]	Keyword assignment after decoding	M1
	CDL [126]	EmoChat + dual tasks learning	M1
Summarization	KIGN [80]	Seq2Seq + KExt (TextRank) + KSeqE	M2
	ComGen [84]	Seq2Seq + KExt (PMI, TFIDF) + KGraphE	M2
	G-S2A [59]	Seq2Seq + KExt + KGraphE	M2
	KGAS [81]	Seq2BF + KExt (BiLSTM) + multi-task	M2

personalized dialogue, as an emerging topic, plays an important role for improving user experience in human-computer interaction. Recent work introduced personalized information into the generation of dialogue to help deliver better dialogue response such as emotion (e.g., *"sad"*) [82, 132, 179], persona [176, 178], politeness [107] and etc. In summarization tasks, abstractive summarization is popular for displaying a document summary in a coherent form that is easily readable and grammatically correct. However, it suffers when the generation process is hard to control and often misses salient information [80]. Making use of keywords as explicit guidance can provide significant clues of the main points about the document [80, 81]. It is closer to the way that humans write summaries: make sentences to contain the keywords, and then perform necessary modifications to ensure the fluency and grammatically correctness.

Researchers have developed a great line of keyword-enhanced NLG methods as summarized in Table 6. These methods can be categorized into two methodologies:

- **M1: Incorporate keyword assignment into text generation.** In keyword assignment, keywords are pre-stored and selected from the prepared vocabularies. To obtain the keywords, a general way is to learn a keyword classifier to predict appropriate keywords based on the input sequence, then use them to guide generation.
- **M2: Incorporate keyword extraction into text generation.** Keyword extraction selects the most important words presented in the document. There are numerous keyword extraction techniques ranging from statistical approaches (e.g., TF-IDF, TextRank) to supervised learning approaches (e.g., BiLSTM). Extracted keywords are then used to enhance the decoding phase.

3.2.1 Incorporate Keyword Assignment into Text Generation. In this methodology, when assigning a keyword to an input document, the set of possible keywords is bounded by a pre-defined vocabulary [128]. One advantage is that the quality of keywords is guaranteed, because irrelevant keywords would not be included in the vocabulary. The keyword assignment is implemented by a classifier that maps the input document to a word in the pre-defined vocabulary [30, 82, 132, 179]. Another advantage is that even if two semantically similar documents do not have common words, they can still be assigned with the same keyword. Unfortunately, some NLG scenarios do not hold an appropriate pre-defined vocabulary, so keyword assignment cannot be widely used to enhance NLG tasks. One applicable scenario is to use a pre-determined domain specific vocabulary to maintain relevance between the input sequence and the output sequence [30]. Another scenario is to generate dialogue with specific attributes such as persona [129, 164], emotion [82, 132, 179].

Adding assigned keyword into the decoder. A straightforward method of keyword assignment is to assign the words from pre-defined vocabulary and use them as the keywords [129, 164, 182]. Sometimes, however, the input sequence does not have an explicit keyword, but we can find one from the pre-defined vocabulary. For example, a dialogue utterance *"If you had stopped him that day, things would have been different."* expresses sadness but it does not have the word "sad." To address this issue, Li et al. propose a method to predict an emotion category by fitting the sum of hidden states from encoder into a one-layer MLP classifier [82]. Then, the response will be generated with the guidance of the emotion category. More concretely, the predicted emotion category $k \in \mathcal{K}$ is first represented by a real-valued, low dimensional vector $\mathbf{k}$. Then, the emotion category embedding $\mathbf{k}$ are combined with the original context vector $\mathbf{c}_t$ together as a new context vector to generate target words, i.e., $\mathbf{c}_t \leftarrow (\mathbf{c}_t \oplus \mathbf{k})$. In order to dynamically track how much the emotion is expressed in the generated sequence, Zhou et al. propose a memory module to capture the emotion dynamics during decoding [179]. Each category is initialized with an emotion state vector before the decoding phase starts. At each step, the emotion state decays by a certain amount. Once the decoding process is completed, the emotion state decays to zero, indicating that the emotion is completely expressed. Xu et al. further improve the performance by learning response generation and query generation with emotions as a dual task, and use the duality to model the mutual relation between them [162].

Assigning keyword for generated sequence. As mentioned in [132], explicitly incorporating emotional keywords suffers from expressing a certain emotion overwhelmingly. Instead, Song et al. propose to increase the intensity of the emotional experiences not by using emotional words explicitly, but by implicitly combining neutral words in distinct ways on emotion [132]. Specifically, they use an emotion classifier to build a sentence-level emotion discriminator, which helps to recognize the responses that express a certain emotion but not explicitly contain too many literal emotional words. The discriminator is connected to the end of the decoder (i.e., the predicted output sequence $\hat{Y}$). Overall, the generation loss of assigning keyword for generated sequence is:

$$\mathcal{L}_{KA-NLL}(\theta) = -\log(p(k|\hat{Y})) - \sum_{t=1}^{m} \log\left(p(y_t|y_{<t}, X)\right), \tag{36}$$

3.2.2 Incorporate Keyword Extraction into Text Generation. Keyword extraction selects salient words from input documents [128]. Many recent works have explored to make use of statistical keyword extraction techniques (e.g., PMI [102], TextRank [80, 84]), and neural-based keyword extraction techniques (e.g., BiLSTM [81]). The process of incorporating extracted keywords into generation is much like the process discussed in Section 3.2.1. It takes keywords as an additional input into decoder. Recent works improve encoding phase by adding another sequence encoder (e.g., RNN) to represent keywords [80, 81], denoted as $\{\mathbf{k}_i\}_{i=1}^{p}$. Then, the contextualized keywords embeddings $\{\mathbf{k}_i\}_{i=1}^{p}$, word embedding $\mathbf{e}(y_{t-1})$, and the context vector $\mathbf{c}_t$ are fed into the decoder together to update the hidden state $\mathbf{s}_t$, i.e., $\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{e}(y_{t-1}), \mathbf{c}_t, \mathbf{c}_t^k)$, where $\mathbf{c}_t^k$ is the keyword context vector. Li et al. use a bi-directional LSTM to encode extracted keywords, and take the last forward hidden state $\overrightarrow{\mathbf{k}_p}$ and backward hidden state $\overleftarrow{\mathbf{k}_1}$ as the key context vector $\mathbf{c}_t^k$, i.e., $\mathbf{c}_t^k = (\overrightarrow{\mathbf{k}_p} \oplus \overleftarrow{\mathbf{k}_1})$ [80]. Li et al. use keyword attention to calculate $\mathbf{c}_t^k$ (details in Table 3) [81]. In addition, Li et al. propose to use multi-task learning for training a keyword extractor network and generating summaries [81]. Because both summarization and keyword extraction aim to select important information from input document, these two tasks can benefit from sharing parameters to improve the capacity of capturing the gist of the input text. In practice, they take overlapping words between the input document and the ground-truth summary as keywords, and adopt a

BiLSTM-Softmax as keyword extractor. Overall, the generation loss is written as:

$$\mathcal{L}_{KE-NLL}(\theta) = -\sum_{k \in K} \log(p(k|X)) - \sum_{t=1}^{m} \log\left(p(y_t|y_{<t}, X, K)\right). \tag{37}$$

3.2.3 Pros and Cons Discussion of Different Methods. The primary advantage of keyword assignment is that the quality of keywords is guaranteed, because irrelevant keywords are not included in the pre-defined vocabulary. Another advantage is that even if two semantically similar documents do not have common words, they can still be assigned with the same keyword. However, there are mainly two drawbacks. On one hand, it is expensive to create and maintain dictionaries in new domains. So, the dictionaries might not be available. On the other hand, potential keywords occurring in the document would be unfortunately ignored if they were not in the vocabulary. Therefore, keyword assignment is suitable for the task that requires specific categories of keywords to guide the generated sentences with these key information. For example, dialogue systems generate responses with specific emotions or attitudes. Keyword extraction selects the most representative words explicitly presented in the document, which is independent from any vocabulary. So, keyword extraction is easy to use. The drawbacks of using keyword extraction lie in two aspects. First, it cannot guarantee consistency because similar documents may still be represented by different keywords if they do not share the same set of words. Second, when an input document does not have a proper representative word, and unfortunately, the keyword extractor selects an irrelevant word from the document as a keyword, this wrong guidance will mislead the generation. Therefore, keyword extraction is suitable for the task that the output sequence needs to keep important information in the input sequence such as document summarization and paraphrase.

3.3 NLG Enhanced by Linguistic Features

In NLG community, feature enriched encoder means that the encoder not only reads the input sequence, but also incorporates auxiliary hand-crafted features [181]. Linguistic features are the most common hand-crafted features, such as lemmas, part-of-speech (POS) tags, dependency parsing, and semantic parsing [124]. In this section, we introduce lemmatisation features, POS tags, NER tags and leave dependency parsing, semantic parsing in Section 4.3.2-4.3.3.

3.3.1 Lemmatisation features. In morphology and lexicography, a lemma is the canonical form and dictionary form of a set of words that have the same meaning.* For example, "run", "runs", "ran", and "running" are forms of the same lexeme, with "run" as the lemma. Modeling sequences of tokens in morphologically rich languages (e.g., German, Czech) is a difficult task of great importance in many NLG tasks like machine translation [31, 124]. Therefore, lemmatisation has been used to help reduce data sparseness and allow inflectional variants of the same word to explicitly share a representation in the model. Semmrich et al. indicated the hidden representation between word forms should be shared in some dimensions if the word forms share the same base form (i.e., a lexeme) [124]. Conforti et al. directly concatenated the word embedding and lemma embedding to obtain a single vector representation for each input word [31].

3.3.2 POS tags and NER tags. Part-of-speech tagging (POS) assigns token tags to indicate the token's grammatical categories and part of speech such as *noun (N)*, *verb (V)*, *adjective (A)*. Named-entity recognition (NER) classifies named entities mentioned in unstructured text into pre-defined categories such as *person (P)*, *location (L)*, *organization (O)*. CoreNLP is the most common used tool [95]. In spite of homonymy and word formation processes, the same surface word form may

*https://en.wikipedia.org/wiki/Lemma_(morphology)

Table 7. Natural language generation methods that incorporate knowledge bases in text generation model.

Task	Method	Description	
Question answering	GenQA [170]	The first work on KB-based QA	M1
	CoreQA [60]	GenQA + three modes of using KB	M1
	HetMem [44]	CoreQA + document (KB's source)	M1
	KEAG [10]	CoreQA + reading contexts	M1
Dialogue system	KBCopy [39]	Enhance dialogue with KB by copy	M1
	Mem2Seq [93]	KBCopy + facts/utterances memory	M1
	GLMP [153]	Mem2Seq + global-to-local memory	M1
	MKST [144]	KBCopy + two-stage decoder	M1
	PostKS [85]	KBCopy + facts selection	M3
	TopicKA [155]	KBCopy + topical facts selection	M3
Content manipulation	U-TCM [150]	KB makes Text CM (TCM) unsupervised	M2
	DS-TCM [41]	U-TCM at document scale manipulation	M2
	FactEditor [67]	U-TCM + three actions for text editing	M2
Summarization	HATS [166]	Seq2Seq + facts retrieval + RL + GAN	M1

be shared between several word types. Incorporating NER tags and POS tags can detect named entities and understand input sequence better, hence, further improve NLG [2, 38, 104, 181].

4 NLG ENHANCED BY KNOWLEDGE BASE/GRAPH, DEPENDENCY GRAPH

4.1 NLG Enhanced by Knowledge Base

One of the biggest challenges in NLG is to discover the dependencies of elements within a sequence and/or across input and output sequences. The dependencies are actually various types of *knowledge* such as commonsense, factual events, and semantic relationship. Knowledge base (KB) is a popular technology that collects, stores, and manages large-scale information for knowledge-based systems like search engines. It has a great number of triples composed of subjects, predicates, and objects. People also call them "facts" or "factual triplets". Recently, researchers have been designing methods to use KB as external knowledge for learning the dependencies easier, faster, and better. Commonly used KBs in general domains include *DBpedia*, *Freebase*, and *Wikidata*.

Applications. One popular task of leveraging KBs in text generation is generation-based question answering (QA) [10, 44, 60]. It is often difficult to generate proper answers only based on a given question. This is because, depending on what the question is looking for, a good answer may have different forms. It may completes the question precisely with the missing information. It may elaborate details of some part of the question. It may need reasoning and inference based on some facts and/or commonsense. As a result, only incorporating input question into neural generation models often fails the task due to the lack of commonsense/factual knowledge [10]. Related structured information of commonsense and facts can be retrieved from KBs.

On the other hand, a proper answer must be in understandable natural language instead of a simple subject or object entity in the base. Here is an example:

- Question: *Do you know where was Jet Li from?*
- Related facts in KB: *<Jet Li, birthplace, Beijing> ; <Jet Li, nationality, Singapore>*
- Answer by KB-based reasoning: *Beijing*
- Expected seq answer: *Jet Li was born in Beijing. He is now a Singaporean citizen.*

Other interesting applications include completing dialogue system [85, 93, 144, 154, 155] and "content manipulation" (CM) [24, 41, 67]. The needs of KB in generating conversations or dialogues

are relevant with QA but differ from two aspects. First, a conversation or dialogue could be open discussions when started by an open topic like "*Do you have any recommendations?*" Second, responding an utterance in a certain step needs to recall previous contexts to determine involved entities. KB will play an important role to recognize dependencies in the long-range contexts. In addition, the task of content manipulation is to describe KB data in a more desired way rather than just putting subject, predicate, and object together as a rigid sentence [24, 41]. It can be seen as the opposite task of style [63, 127]. The gold output of content manipulation is the text consisting of the content of KB data in the same style as the reference sentence. For example, a sports journalist may use particular expressions and language styles to describe specific kinds of games [68].

To address this problem, researchers have developed a great line of methods as summarized in Table 7. To handle different kinds of relationships between KB and input/output sequences, these methods can be categorized into three methodologies:

- **M1: Design supervised tasks around KB for joint optimization.** The knowledge for accomplishing some tasks on KB may be useful for generating a sequence from the other sequence. In other words, when a model learns to perform well in finding information in KB, this extra training can help it generate the output sequence. People design supervised learning tasks around the KB to jointly optimize model parameters.
- **M2: Design unsupervised methods with KB as a conditional factor of sequence.** The input and output sequences may be expected to have underlying information in common (like facts or events), while they look very different as in two different languages or different styles. Then KB, which maintains a large set of facts and events, can be considered as a condition of generating the input and output sequences. For example, adversarial methods are widely used for machine translation and language style transfer.
- **M3: Enhance incorporation by selecting KB or facts.** The relevance of KB with sequences plays an essential role in discovering knowledge for sequence generation. So, selecting relevant KB and/or selecting relevant facts from a KB can enhance KB's incorporation.

We discuss in detail about each methodology in the following subsections.

4.1.1 Design Supervised Tasks around KB for Joint Optimization. Knowledge bases (KBs) that acquire, store, and represent factual knowledge can be used to enhance neural based question answering (QA) and dialogue systems. However, designing effective incorporation to achieve a desired enhancement is challenging because a vanilla Seq2Seq often fails to represent discrete isolated concepts though they perform well to learn smooth shared patterns (e.g., language diversity). To fully utilize the knowledge bases, the idea is to jointly train neural models on multiple tasks. For example, the target task is answer sequence generation, and additional tasks include question understanding and fact retrieval in the KB. Knowledge can be shared across a unified encoder-decoder framework design. Typically, question understanding and fact retrieval are relevant and useful tasks, because a question could be parsed to match (e.g., string matching, entity linking, named entity recognition) its subject and predicate with the components of a fact triple in KB, and the answer is the object of the triple. For example, if the question is *"Where was Barack Obama born in the U.S.?"*, the phrase *"Barack Obama"* can be matched to a fact triple, (*Barack Obama*, *born*, *Hawaii*), in the KB. Parsing the question and retrieving relevant facts can exclude unrelated information and prevent unrelated knowledge from hindering answer generation.

GenQA was the first work to generate answers using factual knowledge bases [170]. During the generation, GenQA is able to retrieve words from the KBs. However, it could not adopt relevant words from the question (i.e., input sequence). It could not handle complex questions that were associated with multiple facts, either. He et al. recognized these two issues and proposed CoreQA. CoreQA used both copying and retrieving mechanisms to generate answer sequences with an

end-to-end fashion [60]. Specifically, it had a retrieval module to understand the question and find related facts from the KB. The retrieval module calculates score S by a matching function between representation of input query ($\mathbf{h}_X$) and each relevant fact ($\{\mathbf{k}_i\}_{i=1}^N$) based on a multi-layer perception (MLP): $S(\mathbf{h}_X, \mathbf{k}) = f_{mlp}(\mathbf{h}_X, \mathbf{k})$. Scoring function also could be bi-linear model [170], rule-based method [10] and etc. A fact representation ($\mathbf{k}$) in KB is the concatenation of word representations of *subject* $\mathbf{k}^{(s)}$, *predicate* $\mathbf{k}^{(p)}$, and *object* $\mathbf{k}^{(o)}$. It is denoted by $\mathbf{k} = (\mathbf{k}^{(s)} \oplus \mathbf{k}^{(p)} \oplus \mathbf{k}^{(o)})$. Then, the question and all retrieved facts are transformed into latent representations by two separate encoders. During the decoding phase, the integrated representations are fed into the decoder. The decoder updates its hidden state ($\mathbf{s}_t$) by performing a joint attention on both input sequence and retrieved facts, i.e., $\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{c}_t, \mathbf{c}_{t-1}^{kb}, \mathbf{e}(y_{t-1}))$, where $\mathbf{c}_t$ is introduced in Eq.(9) and $\mathbf{c}_t^{kb}$ calculates attention over each retrieved fact (see details in Table 3). CoreQA predicts words on a mixed probabilistic model of three modes: (i) generating a word from global vocabulary, (ii) retrieval a factual word from KB, and (iii) copying a word from input sequence (see details in Table 4).

Follow-up work. CoreQA devoted a general and end-to-end framework to generate natural answers by leveraging retrieved information from KBs. It has been extended by many works [10, 44, 154]. For example, Fu et al. extended GenQA and CoreQA by adding a heterogeneous memory in the neural language generation framework, when textual documents were also given for knowledge retrieval besides KBs. Their work used not only the facts in KBs but also semi-structured entities in the retrieved documents [44]. Gao et al. take a Wasserstein distance based adversarial learning method as an additional training signal for generating more consistence answer [48]. Bi et al. developed a method to incorporate a textual document attached to the given question [10]. For conversation and dialogue systems, Madotto et al., addressed CoreQA is not easy to effectively incorporate historical KB into the decoder's hidden states after the systems run multiple steps [93]. They proposed Mem2seq, which used a memory network (i.e., previous utterances are mapped to vectors called memory) to dynamically update information. It combined multi-hop attention mechanisms with pointer networks [141] to effectively incorporate knowledge obtained from the KB. Wu et al. further improved Mem2Seq with a global-to-local memory pointer network [153].

4.1.2 Design Unsupervised Methods with KB as a Conditional Factor of Sequence. Natural language has a variety of styles. News articles, social media tweets, scientific publication are written in different styles. Recently, there has been growing interests in text style transfer. A great line of techniques for controllable NLG have been developed [63, 127]. They modified the property of sentences like transferring from positive sentiment to negative. However, they do not update any essential content to make it consistent with the property change. So, the task of content manipulation was proposed and studied. It is to express the given fact(s) in an expected writing style of a reference (input) sequence [150]. For example, given a structured fact k (player: "Lebron James", points: "20", assists: "10"), and a reference sentence X *"Kobe easily dropped 30 points"*, the goal is to generate a styled sentence Y to describe the fact like *"Lebron easily dropped 20 points and 10 assists"*. The challenge is to understand the content's structure, manipulate according to the reference sentence, and polish the output for grammatical correctness and fluency in an *unsupervised* way. In other words, the problem is to generate a new desired sentence Y based on (X, k), without ground-truth sentence Y for training.

Wang et al. proposed an unsupervised neural generation model to address this problem. The model optimized two objectives with an extra content coverage constraint [150]. One objective is *content fidelity* and the other is *style preservation*. First, content fidelity aims to accurately describe k by reconstructing the original description k', i.e., $p(X'|k, X)$. For instance, X' could be "*Lebron James put together a 20-point, 10-assist double-double*". Since X' was originally written by

human to describe k, it could be seen as maximum data fidelity. Second, style preservation aims to retain writing style by reconstructing X based on k, i.e., $p(X|k', X)$. Here k can be considered as a conditional fact of the input/output sentences. Therefore, the two objectives are competitive with each other. Coupled the two objectives together, the loss function is defined as below:

$$\mathcal{L}_{CM}(\theta) = \mathcal{L}_{content}(\theta) + \mathcal{L}_{style}(\theta) = \log p_\theta(X'|k, X) + \log p_\theta(X|k', X). \tag{38}$$

Briefly speaking, X denotes original description of k without following the reference sentence X', and k' is the fact that X' originally describes. However, Wang et al. model facts and references as independent modules [150]. Instead, Feng et al. created a fused representations of facts and references using an interactive attention mechanism with a hierarchical encoder [41]. It can effectively capture the semantic relatedness with the references to strengthen the capability of content selection from the two types of inputs. In addition, it also uses back-translation [125] to improve the content fidelity and style preservation by constructing pseudo training pairs.

4.1.3 Enhance Incorporation by Selecting KB or Facts in KB. The general method to incorporate KB into NLG is to parse input sequence, retrieve relevant facts, and subsequently, a knowledge-aware output can be generated based on the input sequence and previously retrieved facts. Ideally, the relevance of the facts is satisfactory with the input and output sequence dependencies, which is not always true in real cases. Lian et al. addressed the issue of selecting relevant facts from KBs based on retrieval models (e.g. semantic similarity) might not effectively achieve appropriate knowledge selection [85]. The reason is that different kinds of selected knowledge facts can be used to generate diverse responses for the same input utterance. Given a specific utterance and response pair, the posterior distribution over knowledge base from both the utterance and the response may provide extra guidance on knowledge selection. The challenge lies in the discrepancy between the prior and posterior distributions. Specifically, the model learns to select effective knowledge only based on the prior distribution, so it is hard to obtain the correct posterior distribution during inference.

To tackle this issue, the works of Lian et al. [85] and Wu et al. [154] approximated the posterior distribution using the prior distribution in order to select appropriate knowledge even without posterior information. They introduced an auxiliary loss, called Kullback-Leibler divergence loss (KLDivLoss), to measure the proximity between the prior distribution and the posterior distribution,

$$\mathcal{L}_{KLDiv}(\theta) = \sum_{i=1}^{N} p(k = k_i|X, Y) \log \frac{p(k = k_i|X, Y)}{p(k = k_i|X)}, \tag{39}$$

where N is the number of retrieved facts. When minimizing KLDivLoss, the posterior distribution $p(k|X, Y)$ can be regarded as labels to apply the prior distribution $p(k|X)$ for approximating $p(k|X, Y)$. Finally, the total loss is written as the sum of the KLDivLoss and NLL (generation) loss.

4.2 NLG Enhanced by Knoweldge Graph

Knowledge graph (KG), as a type of structured human knowledge, has greatly attracted research attention from both academia and industry. A KG is a structured representation of facts (a.k.a. knowledge triplets) consisting of entities[†], relations, and semantic descriptions [71]. The terms of knowledge base and knowledge graph are often interchangeably used, but they are not necessarily synonymous. A knowledge graph is organized as a graph, so connections between entities are first-class citizens in a KG. In a KG, People can easily traverse links to discover how entities are interconnected with certain knowledge. Recent advances in artificial intelligence research have demonstrated the effectiveness of KG in various applications such as recommendation system [143].

[†]For brevity, "entities" denotes both entities (e.g., Prince) and concepts (e.g., musician) throughout the paper.

Applications. Dialogue system is the most popular task that makes use of KG for the semantics in linked entities and relations [5, 108, 138, 173, 180]. A dialogue may shift focus from one concept to another, breaking the discourse into several segments, which can be interpreted as a linked path connecting multiple entities on a KG. Other interesting applications include question answering [7], creative writing (e.g., scientific writing [75, 147], story generation [56, 70]), and machine translation [103, 177]. Scientific writing aims to explain natural processes and phenomena step by step, so each step could be reflected as a link on KG and the whole explanation would be a path. In story generation, using implicit knowledge in KGs can facilitate understanding of the storyline and better predict what will happen in the next plot. In abstractive summarization, using commonsense knowledge in KGs can help produce summaries that do not conflict with the facts in the article; representations on KGs can produce a structured summary and highlight the proximity of relevant concepts, when complex events related with the same entity may span multiple sentences.

Instead of using separate, independent knowledge triplets, leveraging KG benefits text generation from the rich semantics in linked entities and relations. As important components of graph, node embedding and path of connected links serve as important roles in various text generation tasks. The corresponding techniques are knowledge graph embedding (KGE) [148] and path based knowledge graph reasoning [25]. Furthermore, harnessed with the emerging graph neural network (GNN) [156] and graph-to-sequence (Graph2Seq) encoder-decoder frameworks [8], it becomes possible to encode multi-hop and high-order relations in KGs. Since there is no widely-accepted formal definition of KG-related concepts [71], we give the definition of (1) KG, (2) sequence-associated subgraph, and (3) k-hop path that will be used in the following sections.

Definition 4.1 (Knowledge graph (KG)). A knowledge graph (KG) is a directed and multi-relational graph composed of entities and relations which are regarded as nodes and different types of edges. Formally, a KG is defined as $\mathcal{G} = (\mathcal{U}, \mathcal{E}, \mathcal{R})$, where $\mathcal{U}$ is the set of entity nodes and $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{R} \times \mathcal{U}$ is the set of typed edges between nodes in $\mathcal{U}$ with a certain relation in the relation schema $\mathcal{R}$.

It should be noted that in an ontological KG (sometimes referred as a curated KG), each knowledge edge is fully disambiguated against a fixed vocabulary of entities and relations; in an open KG, however, nodes are mentions of entities and edges are open relations [14]. In this section, we only introduce ontological KG and leave the open KG in Section 4.3.1. Besides, KG-based technologies can be applied to knowledge bases (KBs) when the entity-relation-entity triplets in KBs are represented as edges in the KGs. Formally, the triplet is written as (u_i, r, u_j), where $u_i, u_j \in \mathcal{U}, r \in \mathcal{R}$ denote the head entity (as source node), tail entity (as target node), and relation (as type of edge).

Definition 4.2 (Sequence-associated K-hop subgraph). A sequence-associated K-hop subgraph is defined as $\mathcal{G}_{sub} = (\mathcal{U}_{sub}, \mathcal{E}_{sub}, \mathcal{R})$, where $\mathcal{U}_{sub}$ is the union of the set of entity nodes mapped through an *entity linking* function $\psi : \mathcal{V} \times \mathcal{X} \rightarrow \mathcal{U}$ *and* their neighbors within K-hops. Similarly, $\mathcal{E}_{sub} \subseteq \mathcal{U}_{sub} \times \mathcal{R} \times \mathcal{U}_{sub}$ is the set of typed edges between nodes in $\mathcal{U}_{sub}$.

Definition 4.3 (K-hop path). A k-hop path from entity nodes $u_0 \in \mathcal{U}$ to entity nodes u_k is defined as a sequence of $k+1$ entity nodes connected by k relations, denoted by $\Phi_k(u_0, u_k) = \left\{ u_0 \xrightarrow{r_1} u_1 \xrightarrow{r_2} \cdots \xrightarrow{r_k} u_k \right\}$ where $u_{t-1} \xrightarrow{r_t} u_t$ represents either $(u_{t-1}, r_t, u_t) \in \mathcal{E}$ or $(u_t, r_t, u_{t-1}) \in \mathcal{E}$, $t \in [1, k]$.

To address this problem, researchers have developed a great line of methods as summarized in Table 8. To learn the relationship between KG semantics and input/output sequences, these methods can be categorized into three methodologies:

- **M1: Incorporate knowledge graph embeddings into language generation.** KGE techniques are to embed entities and relations into continuous vector spaces through a certain

Table 8. Natural language generation methods that incorporate knowledge graph in text generation model. For brevity, we abbreviate knowledge graph embedding to "KGE" and path finding to "PF".

Task	Method	KG Source	M1: KGE	M2: PF	M3: GNN
Dialogue system	CCM [180]	ConceptNet [133]	TransE [13]		GAT [140]
	AKGCM [108]	A self-built KG		RL [160]	
	DyKgChat [138]	A self-built KG		PR [76, 77]	
	ConceptFlow [173]	ConceptNet [133]	TransE [13]		GAT [140]
Machine translation	KG-NMT [103]	DBpedia [3]	TransE [13]		
	KGENMT [177]	DBpedia [3]	TransE [13]		
Question answering	MHPGM [7]	ConceptNet [133]		PR [76, 77]	
Creative writing	StoryEnding [56]	ConceptNet [133]			GAT [140]
	GraphWriter [75]	An abstract KG			GTN [172]
	PaperRobot [147]	A biology KG	TransE [13]		GAT [140]

objective, to preserve inherent structures of a KG. However, KGE is only directly dependent on one-hop relation path and restricted with rigorous objective of a certain scoring function.

- **M2: Perform reasoning over knowledge graph via path finding strategies.** Path finding algorithms provide flexible multi-hop walks on graphs, not restricted to one hop relations. More importantly, path-finding on KG is a process of knowledge reasoning, which can be well integrated with the scenarios of complex QA or dialogue systems.
- **M3: Augment graph representations with graph neural networks.** Recent advances of GNNs and Graph2Seq potentiate to bridge up the gap between graph representation learning and generation. Besides, many NLG task might not need to follow a reasoning process but to understand the global context under a particular generation process. So, GNNs serve as an important role of integrating rich semantic and structural knowledge into text generation.

4.2.1 Incorporate Knowledge Graph Embeddings into language Generation. Knowledge graph embedding (KGE) techniques learn node embedding from KGs [148]. Since KGs provide connectivity information (i.e., different types of relations) between entity nodes, KGE aims to capture semantic relatedness between the entity nodes. The primary idea is to represent entities and relations in a low-dimensional vector space $\mathbb{R}^d$, where $d \ll |\mathcal{U} \cup \mathcal{R}|$, to reduce data dimensionality while preserving the inherent structure of a KG. In KGE, TransE [13] is the most widely used technique. In TransE, given a KG edge (u_i, r, u_j), the relation is seen as a translation vector $\mathbf{r}$ so that the embedded entities $\mathbf{u}_i$ and $\mathbf{u}_j$ can be connected with low error, namely $\mathbf{u}_i + \mathbf{r} \approx \mathbf{u}_j$. For example, we have $\vec{Tokyo} + \vec{IsCapticalOf} \approx \vec{Japan}$ for the knowledge edge (*Tokyo*, *IsCapticalOf*, *Japan*). TransE is adopted by many researchers for NLG tasks because of its simplicity and effectiveness [103, 147]. A common strategy to form a negative set for the margin-based objective in KGE is to build corrupted knowledge edges by replacing either the head or tail of the knowledge edge with a randomly chosen entity node. Then, the low-dimensional entity and relationship embeddings are optimized through stochastic gradient descent with L2 normalization of the entity embeddings:

$$\mathcal{L}_{TransE} = \sum_{u_i, r, u_j \in \mathcal{E}} \sum_{\bar{u}_i, \bar{r}, \bar{u}_j \in \bar{\mathcal{E}}} \max(0, \gamma + ||\mathbf{u}_i + \mathbf{r} - \mathbf{u}_j||_2^2 - ||\bar{\mathbf{u}}_i + \bar{\mathbf{r}} - \bar{\mathbf{u}}_j||_2^2)), \tag{40}$$

where (u_i, r, u_j) is a positive edge and $(\bar{u}_i, \bar{r}, \bar{u}_j)$ is a negative edge, and γ denotes the margin. Then, the natural language generation (NLG) process can benefit from incorporating the entity and relation representations (i.e., embeddings) in both encoding and decoding phase.

Enhancing encoder. On the encoder side, entity nodes are first mapped through an entity linking function $u = \psi(v, X)$. Second, the two channels of word representations ($\mathbf{x}$) and corresponding entity representations ($\mathbf{u}$) are fused together (e.g. $\mathbf{u} \oplus \mathbf{x}$) to enrich the encoder [173, 180].

Enhancing decoder. On the decoder side, the attention mechanism is employed to perform on hidden representations as well as knowledge edges $\mathcal{E}_{sub}$ in the sequence-associated subgraph $\mathcal{G}_{sub}$. Formally, the decoder updates its state $\mathbf{s}_t$ as $\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{c}_t, \mathbf{c}^{ke}_{t-1}, \mathbf{e}(y_{t-1}))$, where $\mathbf{c}_t$ is introduced in Eq.(9) and $\mathbf{c}^{ke}_t$ calculates attention over each knowledge edge (see details in Table 3).

4.2.2 Perform Reasoning over Knowledge Graph via Path Finding Strategies. KGE makes use of one-hop relational path and learns node representations through a certain semantic relatedness (e.g. TransE). An intelligent machine is supposed to have the ability to conduct explicit reasoning over relational paths to make multiple inter-related decisions, rather than merely embedding entities in KGs as latent vectors [157, 160]. Taking the QA task an example, to handle complex queries that do not have an obvious answer, intelligent machines perform reasoning over a KG, infer potential answer-related entities, and generate the corresponding answer. So, the challenge lies in identifying a subset of desired entities and mentioning them in a response [101]. These connected entities usually follow natural conceptual threads which are able to help generate reasonable and logical answers or keep conversations engaging and meaningful. Path based methods explore various patterns of connections among different entity nodes (a.k.a. meta-paths or meta-graphs) and learn walkable paths on KG to provide additional guidance for the generation process. The path finding based methods can be mainly divided into two categories: (1) path ranking based methods and (2) reinforcement learning (RL) based path finding methods.

Path ranking. Path ranking algorithm (PRA) emerges as a promising method for learning and inferring paths on large KGs [76, 77]. PRA uses random walks to perform multiple bounded depth-first search processes to find relational paths. Coupled with elastic-net based learning [185], PRA picks more plausible path to prune non-ideal, albeit factually correct KG paths. For example, Tuan et al. proposed a neural conversation model with dynamic knowledge graphs based on the PRA algorithm [138]. In the decoding phase, it selected an output from two networks, i.e. a general GRU decoder network and a PRA based multi-hop reasoning network, at each time step, in order to generate informative responses. However, the major disadvantage is that PRA operates in a fully discrete space, making it complex to find similar entities and relations in a KG. Bauer et al. rank and filter paths to ensure both the quality and variety of added information via a 3-step scoring strategy, i.e., initial node scoring, cumulative node scoring, and path selection [7]. Ji et al. heuristically prune the noisy edges between entity nodes and propose a path routing algorithm to propagate the edge probabilities along multi-hop paths to the connected entity nodes [69].

Reinforcement learning path finding. Reinforcement learning based methods perform reasoning on finding a path in a continuous space. These methods incorporate various criteria in their reward functions of path finding, making the path finding process flexible. Xiong et al. proposed DeepPath, the first work that employed Markov decision process (MDP) and used RL based approaches to find paths in KGs [160]. However, the state of MDP requires the target entity to be known in advance, so the path finding strategy depends on the answer entity. Thus, it is often not applicable in many real-world QA and dialogue scenarios. Recent RL based knowledge graph reasoning methods have demonstrated appealing performances on non-generative QA scenarios such as answer retrieval [32] and reading comprehension [116]. Leveraging RL based path finding for NLG tasks typically consists of two stages [91, 161]. First, they take sequence X as input, retrieve a starting node u_0 from $\mathcal{G}$, then perform multi-hop graph reasoning, and finally arrive at a target

node u_k that incorporates the appropriate knowledge for output sequence generation. Second, they represent the sequence X and the selected path $\Phi_k(u_0, u_k)$ through two separate encoders. They decode a sequence with multi-source attentions on the input sequence and the selected path (details in Table 3). Path based knowledge graph reasoning converts graph structure of a KG into a linear structure of a path, making it easily represented by sequence encoders (e.g, RNN) [40, 108, 138]. For instance, Niu et al. encoded the selected path $\Phi_k(u_0, u_k)$ and input sequence X though two RNNs, and decodes with a general copy mechanism as introduced in Eq.12 [108]. Overall the generation loss with RL based path finding method is $\mathcal{L}_{KG-RL-NLL}(\theta) = -\sum_{t=1}^{m} \log\left(p(y_t|y_{<t}, \Phi_k(u_0, u_k), X)\right)$.

Follow-up work. Niu et al. addressed that previous methods cannot effectively exploit the long text information within nodes [108]. They improves by using a machine reading comprehension model and a bi-linear model to score each possible node from both global and local perspective. Xu et al. propose an improved adversarial meta-learning algorithm to facilitate dialogue generation with dynamic KGs since entities or relations are temporal and evolve as a single time scale process [161].

4.2.3 Augment Graph Representation with Graph Neural Networks. Path based methods have shown great abilities for reasoning knowledge over a certain KG to bridge knowledge gap between question and answer, or capture conceptual flow in conversation/dialogue systems. However, many NLG tasks might not need to follow a reasoning process but to better understand global context under a particular generation process [56, 65, 75]. For instance, the summarization task requires structured representation to facilitate the connection of relevant entities, and the preservation of global context (e.g. entity interactions) [65]. A graph representation produces a structured summary and highlights the proximity of relevant concepts. Recent advances of Graph Neural Networks (GNN) demonstrate a promising advancement in graph-based representation learning [156]. In order to help NLG, graph-to-sequence (Graph2Seq) models encode the full structural information contained in the graph via a neural encoder-decoder architectures [8]. Thus, GNNs have been gradually playing an important role in helping generation process.

Enhancing encoder. For encoding phase, a general process of leveraging GNNs for incorporating KG is to augment semantics of a word by including the corresponding entity node vector [56, 65, 173, 180]. A pre-defined entity linking function $\psi : \mathcal{V} \times \mathcal{X} \rightarrow \mathcal{U}$ maps a word in the input sequence to a corresponding entity node in the KG. For an input sequence, all linked entities and their neighbors within K-hops are represented as a *sequence-associated K-hop subgraph* $\mathcal{G}_{sub}$. For each entity node in $\mathcal{G}_{sub}$, it uses the KG structure as well as entity and edge features (if have, e.g., semantic description) to learn a representation vector $\mathbf{u}$ for each entity node. The sub-graph representation $\mathbf{h}_{subG}$ is learned thorough a $\textsc{Readout}(\cdot)$ function from all entity node representations (i.e., $\mathbf{h}_{subG} = \textsc{Readout}(\{\mathbf{u}, u \in \mathcal{U}_{sub}\})$ as introduced in Section 2.2.4). Zhou et al. was the first to design such a KG interpreter to facilitate the understanding of a input sequence [180].

Besides, some recent work transforms the input KG into its equivalent Levi graph [8], which treats each entity node and relational edge equally in a KG. In this process, an edge is replaced by two nodes: one representing the forward direction of the relation and the other representing the reversed direction. These nodes are connected to the entity nodes so that the directionality of the former edge is maintained. So, it restructures the original knowledge graph as an unlabeled directed graph. Existing NLG works have used such a KG format to help generation process [65, 75, 183].

Enhancing Decoder. For decoding phase, the initial hidden state is augmented by subgraph representation $\mathbf{h}_{subG}$, i.e., $\mathbf{s}_0 = \mathbf{h}_n \oplus \mathbf{h}_{subG}$ [8]. The KG enhanced decoder attentively reads the retrieved subgraph to obtain a graph-aware context vector, then uses the vector to update the decoding state [56, 65, 173, 180, 183]. It adaptively chooses a generic word or an entity from the

retrieved subgraph for word generation. Since graph-level attention alone may overlook fine-grained knowledge edge information, many existing works adopt a hierarchical graph attention mechanism [56, 180]. It first attentively reads the retrieved subgraph $\mathcal{G}_{sub}$ and then attentively reads all knowledge edges $\mathcal{E}_{sub}$ involved in $\mathcal{G}_{sub}$ for final word generation. The decoder updates its state as $\mathbf{s}_t = f_{de}(\mathbf{s}_{t-1}, \mathbf{c}_t, \mathbf{c}^{kg}_{t-1}, \mathbf{c}^{ke}_{t-1}, \mathbf{e}(y_{t-1}))$, where $\mathbf{c}_t$ is introduced in Eq.(3); $\mathbf{c}^{kg}_t$ and $\mathbf{c}^{ke}_{t-1}$ denote graph-level context vector and edge-level context vector at time step t. The hierarchical graph attention mechanism pays attention to not only subgraph but also fine-grained entities and relations. The graph-level context vector $\mathbf{c}^{kg}_t$ is a weighted sum of the entity node vectors, measuring the association between the decoding state $\mathbf{s}_t$ and entity node vector $\mathbf{u}$ (see details in Table 3).

Follow-up work. Followed by [180], Wang et al. proposed a novel system for scientific writing, called PaperRobot [147]. It can iteratively construct knowledge graphs, predict new links between concepts, and write new paper drafts. Zhang et al. proposed ConceptFlow that represents the potential conversation flow as traverses in the concept space along commonsense relations [173]. The traverses in the concept graph are guided by graph attention mechanisms (GAT) to attend on more appropriate concepts. Specifically, ConceptFlow is able to grow the grounded concepts by hopping from the conversation utterances, along the commonsense relations, to distant but meaningful concepts, guiding the model to generate more informative and on-topic responses.

4.2.4 Pros and Cons Discussion of Different Methods. Knowledge graph embedding (KGE) embeds components of a KG including entities and relations into continuous vector spaces, so as to simplify the manipulation while preserving the inherent structure of the KG. Those entity and relation embeddings can simply be used to enrich input text representations (e.g., concatenating embeddings), bridging connections between entity words linked from input text in latent space. However, KGE fails to model complex relation paths. KGE is often used as a basic method in combination with KG path reasoning and GNNs [173, 180]. The choice between path reasoning and GNNs mainly depends on the purpose and application of using knowledge graphs. Relation path reasoning leverages path information over the graph structure, allowing one to infer indirect facts. (Conversational) question answering system is the most important application. GNN is introduced for learning multi-hop connectivity structure by iteratively aggregating information from neighboring nodes and edges. It directly supplies structured knowledge related to entity words in the input text, and assists in understanding global context under a particular generation process.

4.3 NLG Enhanced by Other Graph Structures

4.3.1 Internal knowledge graph (open KG). Above mentioned KGs (e.g., ConceptNet) are constructed based on data beyond the input text. We refer them as *external KGs*. On the contrary, an *internal KG* is defined as a KG constructed solely based on the input text. In this section, we will mainly discuss recent work that incorporated internal KG tos help text generation [26, 40, 65, 183].

Internal KG plays an important role in understanding the input sequence especially when it is of great length. By constructing an internal KG intermediary, redundant information can be merged or discarded, producing a substantially compressed form to represent the input document [40]. Besides, representations on KGs can produce a structured summary and highlight the proximity of relevant concepts, when complex events related with the same entity may span multiple sentences [65]. One of the mainstream methods of constructing an internal KG is using open information extraction (OpenIE). Unlike traditional information extraction (IE) methods, OpenIE is not limited to a small set of target entities and relations known in advance, but rather extracts all types of entities and relations found in input text [106]. In this way, OpenIE facilitates the domain independent discovery of relations extracted from text and scales to large heterogeneous corpora.

After obtaining an internal KG, the next step is to learn the representation of the internal KG and integrate it into the generation model. For example, Zhu et al. use a graph attention network (GAT) to obtain the representation of each node, and fuse that into a transformer-based encoder-decoder architecture via attention [183]. Their method generates abstractive summaries with higher factual correctness. Huang et al. extend by first encoding each paragraph as a sub-KG using GAT, and then connecting all sub-KGs with a Bi-LSTM [65]. This process models topic transitions and recurrences, which enables the identification of notable content, thus benefiting summarization. Fan et al. propose to construct an internal KG under a multiple input document scenario [40]. The graph construction process (i) compresses multiple documents to a significantly smaller size, allowing models to encode the entirety of the compression, and (ii) reduces redundancy through merge operations, allowing relevant information to be more easily identified. Fan et al. add hierarchical and memory compressed attention mechanisms to a standard Graph2Seq [8], in order to encode the full graph and attend the most relevant information in it.

4.3.2 Syntactic dependency graph. Syntactic dependency graph is a directed acyclic graph representing syntactic relations between words [6]. For example, in the sentence "The monkey eats a banana", "monkey" is the subject of the predicate "eats", and "banana" is the object. Enhancing sequence representations by utilizing dependency information captures source long-distance dependency constraints and parent-child relation for different words [1, 6, 21, 27]. In NLG tasks, dependency information is often modeled in three different ways as follows: (i) linearized representation: linearize dependency graph and then use sequence model (e.g., RNN) to obtain systax-aware representation [1]; (ii) path-based representation: calculate attention weights based on the linear distance between a word and the aligned center position, i.e., the greater distance a word to the center position on the dependency graph is, the smaller contribution of the word to the context vector is [21]; and (iii) graph-based representation: use graph neural network (GNN) to aggregate information from dependency relations [6, 27].

4.3.3 Semantic dependency graph. Semantic dependency graph represents *predicate-argument* relations between content words in a sentence and have various semantic representation schemes (e.g., DM) based on different annotation systems. Nodes in a semantic dependency graph are extracted by semantic role labeling (SRL) or dependency parsing, and connected by different intra-semantic and inter-semantic relations [109]. Since semantic dependency graph introduces a higher level of information abstraction that captures commonalities between different realizations of the same underlying predicate-argument structures, it has been widely used to improve text generation [72, 87, 109, 131]. Jin et al. propose a semantic dependency guided summarization model [72]. They incorporate the semantic dependency graph and the input text by stacking encoders to guide summary generation process. The stacked encoders consist of a sequence encoder and a graph encoder, in which the sentence encoder first reads the input text through stacked multi-head self-attention, and then the graph encoder captures semantic relationships and incorporates the semantic graph structure into the contextual-level representation. Some recent work leverages abstract meaning representation (AMR) as a structured semantic representation to improving text generation performances [87, 131]. Compared with semantic roles, AMR is able to directly capture entity relations and abstract away inflections and function words. Song et al. demonstrate that making use of AMR helps in enforcing meaning preservation and handling data sparsity (i.e., many sentences correspond to one meaning) of machine translation models.

4.3.4 Keyword/sentence graph. Instead of using sequence encoder to represent extracted keywords, recent studies propose to construct a keyword interaction graph to represent documents especially when they are of great length [59, 84]. It has been shown that keyword interaction graph can

Table 9. Natural language generation methods that incorporate grounded text in text generation.

Task	Method	Description	
Dialogue system	KGNCM [52]	RNN-Seq2Seq + retrieve review snippets	M1
	WizardWiki [36]	Transformer + retrieve wiki articles + selection loss	M1
	DeepCopy [169]	KGNCM + hierarchical pointer network	M1
	CMR [114]	RNN-Seq2Seq + background document attention	M2
	RefNet [96]	CMR + two decoding modes, i.e., add background copy	M2
	GLKS [121]	CMR + a global-to-local knowledge selection	M2
Question answering	RAGE [23]	RNN-Seq2Seq + retrieve review snippets	M1
	PAAG [48]	RNN-Seq2Seq + retrieve review snippets	M1
	LatentQA [11]	CMR + stochastic selector network	M2
Summarization	Re^{3}Sum [18]	Retrieve and rerank summary template + rewrite	M1
	BiSET [145]	Re^{3}Sum + bi-directional selective template encoder	M1
Content paraphrase	FSET [73]	Retrieve most similar pair to paraphrase	M1

reflect the structure of a document. There are two steps constructing a keyword graph. First, the document is decomposed into several keyword centered clusters of text, each of which together with the keyword form a node in the graph. Second, each sentence of the documents is associated to one corresponding keyword node if the keyword appears in the sentence. The edges between nodes are built based on the semantic relationship between the nodes. Note that one sentence can be associated with multiple keyword nodes, which implicitly indicates connection between the two keywords. Sentences that do not contain any keyword are put into a special node called "Empty". Li et al. and Han et al. use graph-to-sequence (Graph2Seq) models based on the constructed keyword interaction graph [59, 84].

5 NLG ENHANCED BY GROUNDED TEXT

Knowledge grounded text refers to textual information that can provide additional knowledge relevant to the input sequence. The textual information may not be found in training corpora or structured databases (like knowledge bases and knowledge graphs that are mentioned in Sections 6 and 7), but can be obtained from massive textual data from online resources. These online resources include encyclopedia (e.g., Wikipedia), social media (e.g., Twitter), shopping websites (e.g., Amazon reviews). Knowledge grounded text plays an important role in understanding the input sequence and its surrounding contexts. For example, Wikipedia articles may offer textual explanations or background information for the input text. Amazon reviews may contain necessary descriptions and reviews needed to answer a product-related question. Tweets may contain people's comments and summaries towards an event. Therefore, knowledge grounded text is often taken as an important external knowledge source to help with a variety of NLG tasks.

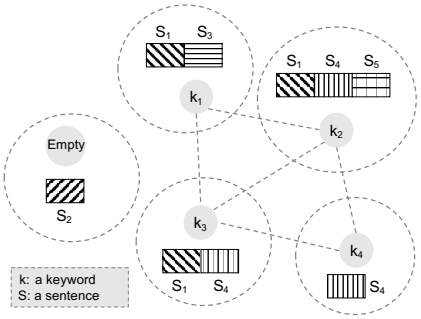

Fig. 2. An illustration of keyword graph.

Applications. Knowledge grounded text has been widely researched for improving the interactive experience in dialogue systems [36, 52, 96, 114, 169]. This is because knowledge grounded text provides supplementary information of a mentioned entity or background knowledge of the entire

dialogue. These works can be divided into two categories according to the source and usage of knowledge grounded text. Methods in the first category typically follow a two-stage process: retrieval and generation. The first stage retrieves snippets from a large collection of raw text entries (e.g., Wikipedia, Foursquare) or search engines (e.g., Google). It assumes that the top-k results cover the most related knowledge. The second stage feeds both dialogue history and retrieved snippets into a neural generation architecture [96, 114]. The second category is called background based conversation (BBC) [36, 52, 169]. BBC establishes a conversation mode in which relevant information can be obtained from the given document. In addition to dialogue systems, the interesting applications include question answering [11, 23, 48], summarization [18, 47, 110, 145], content paraphrase [22, 73] and machine translation [15, 165]. In summarization, Seq2Seq models that purely depend on the input text tend to "lose control" sometimes. For example, 3% of summaries contain less than three words, and 4% of summaries repeat a word for more than 99 times as mentioned in [18]. Furthermore, Seq2Seq models usually focus on copying source words in their exact order, which is often sub-optimal in abstractive summarization. Therefore, leveraging summaries of documents similar as the input document as templates can provide reference for the summarization process [18]. In the task of question answering (QA), it is often difficult to generate proper answers only based on the given question. For example, without knowing any information of an Amazon product, it is hard to deliver satisfactory answer to the user questions such as *"Does the laptop have a long battery life?"* or *"Is this refrigerator frost-free?"* So, the product description and customer reviews can be used as a reference for answering product-related questions [23, 48].

Table 9 summarizes representative NLG models that are enhanced by knowledge grounded text. They can be categorized into two methodologies:

- **M1: Guiding generation with retrieved information.** Because knowledge grounded text is not presented in the training corpora, an idea is to retrieve relevant textual information (e.g., a review, a summary template) from *external sources* based on the input text and to incorporate the retrieved grounded text into the generation process. This process is similar to designing knowledge acquisition and incorporation of KBs and KGs in NLG tasks. The difference is that ground text is unstructured and noisy. So, researchers design knowledge selection and incorporation methods to address the challenges.
- **M2: Modeling background document into response generation.** The retrieved snippets are locally related to the input sequence. Background document, with more global and comprehensive knowledge, has been used for generating informative responses and ensuring a conversation to not deviate from its topic. Keeping a conversation grounded on a background document is referred as background based conversation (BBC) [11, 100]. It is important to find an appropriate background snippet and generate response based on the snippet.

5.1 Guiding Generation with Retrieved Information

5.1.1 Retrieve relevant snippets. In dialogue system and question answering, people often search and acquire external information as needed to respond or answer questions. However, building a fully data-driven dialogue or QA system is difficult since most of the universal knowledge is not presented in the training corpora [52]. The lack of universal knowledge considerably limits the appeal of fully data-driven generation methods, as they are bounded to respond evasively or defectively and seldom include meaningfully factual contents. Take an example in [52],

- User input: *"Going to Kusakabe tonight."*
- Reviews on Foursquare: *"Consistently the best omakase in San Francisco."*
- Response given by vanilla Seq2Seq: *"Have a great time!"*
- Golden Response: *"You'll love it! Try omakase, the best in town."*

Though the response from a vanilla Seq2Seq is appropriate, it lacks content when compared to the golden response. To infuse the response with factual information relevant to input text, an intelligent machine has to first retrieve related fact or review snippets that contain necessary background information. Methods for retrieving fact or review snippets are various, including matching from a collection of raw text entries indexed by named entities [52]; scoring relevant documents within a large document collection by statistical approaches such as TF-IDF and BM25 [36].

Ghazvininejad et al. proposed a knowledge grounded neural conversation model (KGNCM), which is the first work to retrieve review snippets from Foursquare and Twitter. Then it incorporates the snippets into dialogue response generation [52]. It uses an end-to-end memory network [134] to generate responses based on the selected review snippets $K = \{k_1, \cdots, k_p\}$. Memory network calculates an attentive presentation $\mathbf{h}_K$ over all retrieved review snippets. The hidden state of the decoder is initialized with $\mathbf{h}_K$, i.e., $\mathbf{s}_0 = \mathbf{h}_K$. Formally, the attention over the knowledge relevant to the conversation context is given by

$$\mathbf{h}_K = \sum_{i=1}^{p} \alpha_i \mathbf{h}_{k_i}, \text{ where} \alpha_i = \frac{\exp(\eta(\mathbf{h}_X, \mathbf{h}_{k_i}))}{\sum_{j=1}^{p} \exp(\eta(\mathbf{h}_X, \mathbf{h}_{k_j}))}, \tag{41}$$

where $\mathbf{h}_X \in \mathbb{R}^d$ and $\mathbf{h}_{k_i} \in \mathbb{R}^d$ are embeddings of input sequence and i-th retrieved fact snippet. However, Ghazvininejad et al. did not enhance the decoding phase. In order to jointly attend and copy tokens from all available facts as external knowledge, Yavuz et al. used a hierarchical pointer network to determine the probability of copying a token from each fact snippet [169]. It not only attends over the knowledge facts around the decoding step to capture the importance of each snippet, but also adopts the copy mechanism to copy useful words from the snippet. Equipped with a soft switch mechanism between copy and generation modes, it allows to softly combine the copying probabilities with the generation probabilities into a final output probability distribution over an extended vocabulary. In QA scenarios, Chen et al. [23] and Gao et al. [48] incorporated online customer reviews to answer product-related questions.

5.1.2 Retrieve and rerank soft templates. In summarization, Seq2Seq models that purely depend on the input document to generate summaries tend to deteriorate with the accumulation of word generation, e.g., they generate irrelevant and repeated words frequently [18, 145]. Template-based summarization [18, 110, 145] assume the golden summaries of the similar sentences (i.e., templates) can provide a reference point to guide the input sentence summarization process. These templates are often called *soft templates* in order to distinguish from the traditional rule-based templates. Soft template-based summarization typically follows a three-step design: retrieve, rerank, and rewrite. The step of retrieval aims to return a few candidate templates from a summary collection. The reranking identifies the best template from the retrieved candidates. And the rewriting leverages both the source document and template to generate more faithful and informative summaries. In retrieval, similar with retrieving fact/review snippets, the candidate templates are ranked according to the text similarity between each candidate template and the input document [18, 145]. The reranking module is developed to identify the best template T that also resembles the actual summary Y as much as possible. In [18, 145], the actual saliency $S^*(T, Y)$ is calculated by ROUGE, while the predicted saliency of the template for the input sentence $S(T, X)$ is calculated by a bi-linear network. Importantly, the predicted saliency $S(T, X)$ should be close to the actual saliency $S^*(T, Y)$ as much as possible. So, the generation loss for soft template-based summarization is written as $\mathcal{L}_{ST}(\theta) = \mathcal{L}_{Rerank}(\theta) + \mathcal{L}_{ST-NLL}(\theta) = -\big[S^*(T, Y) \log S(T, X) + \log\left(p(y_t|y_{<t}, X, \text{argmax}_{T \in \mathcal{T}} p(T|X))\right)\big]$.

Follow-up work. Wang et al. proposed a bi-directional selective encoding with template (BiSET) model [145]. BiSET involves a novel bi-directional selective layer with two gates to mutually select key information from an article and its template to assist with summary generation.

5.2 Modeling Background Knowledge into Response Generation

Background Based Conversation (BBC, also known as grounded conversation), as an emerging topic, has been proposed for generating informative responses that are grounded on one or multiple background documents [11, 100]. Background knowledge plays an important role in human-human conversations. For example, when talking about a movie, people often recall important points (e.g., a scene or review about the movie) and appropriately mention them in the conversation context. The task of BBC is often compared with machine reading comprehension (MRC), in which a span is extracted from the background document as a response to a question [119]. However, since BBC needs to generate natural and fluent responses, the challenge lies in not only locating the right semantic units (i.e., snippets) in the background, but also referring to the right background information at the right time in the right place during the decoding phase.

As MRC models tie together multiple text segments to provide a unified and factual answer, many BBC models use the same idea to connect different pieces of information and find the appropriate background knowledge based on which the next response is to be generated [96, 114, 121]. For instance, Qin et al. proposed an end-to-end conversation model that jointly learned response generation together with on-demand machine reading [114]. The MRC models can effectively encode the input utterance X by treating it as a question in a typical QA task (e.g., SQuAD [119]) and encode the background document B as the context. They took the utterance-aware background representation $\{\mathbf{h}_i^{rc}\}_{i=1}^{|B|}$ as input into decoding phase. So, the context vector $\mathbf{c}_t^{rc}$ is no longer calculated by a weighted sum of hidden states $\{\mathbf{h}_i\}_{i=1}^{n}$ of input sequence, but by a weighted sum of utterance-aware background hidden states $\{\mathbf{h}_i^{rc}\}_{i=1}^{|B|}$,

$$\mathbf{c}_t^{rc} = \sum_{i=1}^{|B|} \alpha_{ti} \mathbf{h}_i^{rc}, \text{ where } \mathbf{h}_i^{rc} = \mathrm{RC}(X, B)[i], \tag{42}$$

where α_{ti} is the attention weight introduced in Eq.(9), B is a background document, $|B|$ is its number of words, $\mathrm{RC}(\cdot,\cdot)$ is a reading comprehension model. Additionally, $\mathrm{RC}(X, B) \in \mathbb{R}^{|B| \times d}$, $\mathrm{RC}(X, B)[i] \in \mathbb{R}^d$ is the i-th vector representation, and d is the vector dimension.

Follow-up work. Although utterance-aware background representation $\{\mathbf{h}_i^{rc}\}_{i=1}^{|B|}$ make soft alignment (i.e., cross attention) between utterance and background document, it does not accurately locate the background information and explicitly use snippets to guide the generation process. Therefore, Meng et al. proposed a reference-aware network (RefNet) to address the problem [96]. It not only obtained the context-aware background representation through a MRC-based model, but also incorporated a novel reference decoder learning to directly select a semantic unit (i.e., a snippet) from the background supervised by annotated spans. Furthermore, Ren et al. argued that selecting a snippet only leverages the background document from a *local* perspective [121]. It is problematic due to lack of the guidance from a *global* perspective. So, Ren et al. enhanced knowledge selection in background document by introducing a global-to-local knowledge selection (GLKS) mechanism. GLKS first learned a topic transition vector to encode the most likely text fragments to be used in the next response, which was then used to guide the local knowledge selection (i.e., snippet selection) at each decoding timestamp. Besides, Bi et al. extended background based conversation (BBC) to background based QA scenarios, specifically, to answer a certain question based on a background document [11]. They proposed a novel stochastic selector network that selects words to form a final

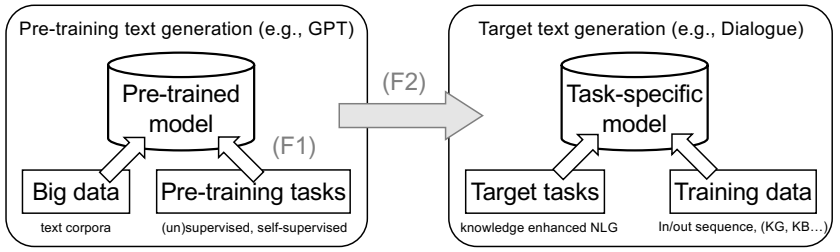

Fig. 3. Future direction of using pre-training methods. The first future direction (F1) is to design knowledge-enhanced pre-training tasks. The second direction (F2) is to discover knowledge from pre-trained models.

answer directly from the modeled relationship between the question and the background document. In the stochastic selector network, a sequence of discrete latent variables indicates which source produces an answer word. The word in a well-formed answer comes from one of three sources: the question, the background document, or the pre-defined vocabulary.

6 DISCUSSION ON FUTURE DIRECTIONS

Many efforts have been conducted to tackle the problem of knowledge-enhanced text generation and its related applications. To advance the field, there remains several open problems and promising future directions. First, designing more effective ways to represent knowledge and integrate them into the generation process is still the most important trend in knowledge-enhanced text generation systems. Besides, more research effort should be spent on learning to discover knowledge more broadly and combine multiple forms of knowledge from different sources to improve the generation process. For examples, multi-task learning can make mutual enhancement between knowledge representation and text generation [18, 81]; structured knowledge and unstructured knowledge can play a complementary role in enhancing text generation [44].

From a broader perspective, we provide three future directions that make focusing such efforts worthwhile now: (i) pre-training, (ii) meta-learning, and (iii) lifelong learning.

6.1 Pre-training

The emergence of pre-training has brought natural language processing (NLP) to a new era. In NLG area, pre-trained language generation models (e.g., UniLM) have demonstrated superior performances on downstream tasks than many state-of-the-art generation models with simply fine-tuning on a small amount task-specific data [37]. Since text generation tasks are usually data-hungry, and many of them are low-resource in terms of labelled data [130], making use of pre-trained language generation models can alleviate the problem by leveraging massive unlabelled data. Pre-training allows model to learn universal language representations, provide a better parameter initialization, and avoid overfitting on downstream tasks with small training data.

In practice, however, the improvements on the downstream tasks contributed by the pre-trained generation models are not as much as expected in many real-world scenarios. As mentioned in [55], directly fine-tuning pre-trained language generation models on the story generation task still suffers from insufficient knowledge by representing the input text thorough a pre-trained encoder, leading to repetition, logic conflicts, and lack of long-range coherence in the generated output sequence. Therefore, incorporating knowledge into pre-training is one potential solution to combine the advantages of pre-training and knowledge enhancement. Existing work in NLP community has explored the idea of explicitly modeling task-specific knowledge by designing pre-training tasks on massive unlabelled data [55, 159, 178]. It is thus important to design novel

pre-training tasks and methods that incorporate knowledge for specific text generation tasks, which will certainly bring promising progress to the knowledge-enhanced text generation systems. On the other side, currently, fine-tuning is the dominant method to transfer pre-training knowledge to downstream tasks, but discovering knowledge from pre-trained models can be more flexible, such as knowledge distillation, data augmentation, using pre-trained models as external knowledge [113]. More efficient methods of obtaining knowledge from pre-trained models are expected.

6.2 Meta Learning

Most of current NLG research conduct on extensively labelled data to favor model training. However, this is in contrast to many real-world application scenarios, where only a few shots of examples are available for new domains. Limited data resources lead to limited knowledge that can be learnt in new domains. For examples, learning topical information of a dialogue occurring under a new domain is difficult since the topic may be rarely discussed before; constructing a syntactic dependency graph of a sequence in a low-resource language is hard since many linguistic features are of great uniqueness. Besides, external knowledge bases are often incomplete and insufficient to cover full entities and relationships due to the human costs of collecting domain-specific knowledge triples. Therefore, quick domain adaptation is an essential task in text generation tasks. One potential route towards addressing these issues is meta-learning [42], which in the context of NLG means a generation model develops a broad set of skills and pattern recognition abilities at training time, and quickly adapt to a new task given very few examples without retraining the model from scratch. Recently, there has been raising interests in both academia and industry to investigate meta-learning in different NLG tasks. Thus, it is a promising research direction to build efficient meta-learning algorithms that only need a few task-specific fine-tuning to learn the new task quickly. And for knowledge-enhanced text generation, it is of crucial importance to adapt the model quickly on new domains with limited new knowledge (e.g., only a few knowledge triples).

6.3 Lifelong Learning

Lifelong learning is an advanced machine learning paradigm that learns constantly, accumulates the knowledge learned in previous tasks, and uses it to assist future learning [28]. In the process, the intelligent machine becomes more and more knowledgeable and effective at learning new knowledge. To make an analogy, humans continuously acquire new knowledge and constantly update the knowledge system in the brain. However, existing knowledge-enhanced text generation systems usually do not keep updating knowledge in real time (e.g., knowledge graph expansion). Therefore, it is a promising research direction to continuously update knowledge obtained from various information sources, empowering intelligent machines with incoming knowledge and improving the performances on new text generation tasks.

7 CONCLUSIONS

In this survey, we present a comprehensive review of current representative research efforts and trends on knowledge-enhanced text generation, and expect it can facilitate future research. To summarize, this survey aims to answer two questions that commonly appears in knowledge-enhanced text generation: *how to acquire knowledge* and *how to incorporate knowledge to facilitate text generation*. Base on knowledge acquisition, the main content of our survey is divided into three sections according to different sources of knowledge enhancement. Based on knowledge incorporation, we first present general methods of incorporating knowledge into text generation and further discuss a number of specific ideas and technical solutions that incorporate the knowledge to enhance the text generation systems in each section. Besides, we review a variety of text generation applications in each section to help practitioners learn to choose and employ the methods.

REFERENCES

[1] Roee Aharoni and Yoav Goldberg. 2017. Towards String-To-Tree Neural Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[2] Reinald Kim Amplayo, Seonjae Lim, and Seung-won Hwang. 2018. Entity Commonsense Representation for Neural Abstractive Summarization. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *the Semantic Web*.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference for Learning Representation (ICLR)*.

[5] Suman Banerjee and Mitesh M Khapra. 2019. Graph convolutional network with sequential attention for goal-oriented dialogue systems. In *Transactions of the Association for Computational Linguistics (TACL)*.

[6] Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[7] Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for Generative Multi-Hop Question Answering Tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[8] Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[9] Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *International Conference for Learning Representation (ICLR)*.

[10] Bin Bi, Chen Wu, Ming Yan, Wei Wang, Jiangnan Xia, and Chenliang Li. 2019. Incorporating External Knowledge into Machine Reading for Generative Question Answering. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[11] Bin Bi, Chen Wu, Ming Yan, Wei Wang, Jiangnan Xia, and Chenliang Li. 2020. Generating Well-Formed Answers by Machine Reading with Stochastic Selector Networks. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[12] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. In *Journal of Machine Learning Research (JMLR)*.

[13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[14] Samuel Broscheit, Kiril Gashteovski, Yanjie Wang, and Rainer Gemulla. 2020. Can We Predict New Facts with Open Knowledge Graph Embeddings? A Benchmark for Open Link Prediction. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[15] Bram Bulté and Arda Tezcan. 2019. Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *57th Conference of the Association for Computational Linguistics (ACL)*.

[16] Hengyi Cai, Hongshen Chen, Yonghao Song, Xiaofang Zhao, and Dawei Yin. 2020. Exemplar Guided Neural Dialogue Generation. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[17] Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[18] Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[19] Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Annual Meeting of the Association of Computational Linguistics (ACL)*.

[20] Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. 2018. Hierarchical variational memory network for dialogue generation. In *World Wide Web Conference (WWW)*.

[21] Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[22] Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable Paraphrase Generation with a Syntactic Exemplar. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[23] Shiqian Chen, Chenliang Li, Feng Ji, Wei Zhou, and Haiqing Chen. 2019. Driven answer generation for product-related questions in e-commerce. In *International Conference on Web Search and Data Mining (WSDM)*.

[24] Shuang Chen, Jinpeng Wang, Xiaocheng Feng, Feng Jiang, Bing Qin, and Chin-Yew Lin. 2019. Enhancing Neural Data-To-Text Generation Models with External Background Knowledge. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[25] Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. In *Expert Systems with Applications*.

[26] Yongrui Chen, Huiying Li, Yuncheng Hua, and Guilin Qi. 2020. Formal Query Building with Query Structure Prediction for Complex Question Answering over Knowledge Base. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[27] Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2020. Reinforcement learning based graph-to-sequence model for natural question generation. In *International Conference of Learning Representation (ICLR)*.

[28] Zhiyuan Chen and Bing Liu. 2018. Lifelong machine learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers.

[29] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[30] Sajal Choudhary, Prerna Srivastava, Lyle Ungar, and Joao Sedoc. 2017. Domain aware neural dialog system. In *arXiv preprint arXiv:1708.00897*.

[31] Costanza Conforti, Matthias Huck, and Alexander Fraser. 2018. Neural morphological tagging of lemma sequences for machine translation. In *Conference of the Association for Machine Translation in the Americas (AMTA)*.

[32] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference for Learning Representation (ICLR)*.

[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 *International Conference for Learning Representation (ICLR)*.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[35] P Kingma Diederik, Max Welling, et al. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[36] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference for Learning Representation (ICLR)*.

[37] 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 *Advances in Neural Information Processing Systems (NeurIPS)*.

[38] Xiangyu Dong, Wenhao Yu, Chenguang Zhu, and Meng Jiang. 2020. Injecting Entity Types into Entity-Guided Text Generation. In *arXiv preprint arXiv:2009.13401*.

[39] Mihail Eric and Christopher D Manning. 2017. A Copy-Augmented Sequence-to-Sequence Architecture Gives Good Performance on Task-Oriented Dialogue. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

[40] Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using Local Knowledge Graph Construction to Scale Seq2Seq Models to Multi-Document Inputs. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[41] Xiaocheng Feng, Yawei Sun, Bing Qin, Heng Gong, Yibo Sun, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. Learning to Select Bi-Aspect Information for Document-Scale Text Content Manipulation. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*.

[43] Xiyan Fu, Jun Wang, Jinghan Zhang, Jinmao Wei, and Zhenglu Yang. 2020. Document Summarization with VHTM: Variational Hierarchical Topic-Aware Mechanism. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[44] Yao Fu and Yansong Feng. 2018. Natural answer generation with heterogeneous memory. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT))*.

[45] Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. In *Journal of Machine Learning Research (JMLR)*.

[46] Ce Gao and Jiangtao Ren. 2019. A topic-driven language model for learning to generate diverse sentences. In *Neurocomputing*.

[47] Shen Gao, Xiuying Chen, Piji Li, Zhaochun Ren, Lidong Bing, Dongyan Zhao, and Rui Yan. 2019. Abstractive text summarization by incorporating reader comments. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[48] Shen Gao, Zhaochun Ren, Yihong Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019. Product-aware answer generation in e-commerce question-answering. In *International Conference on Web Search and Data Mining (WDSM)*.

[49] Cristina Garbacea and Qiaozhu Mei. 2020. Neural Language Generation: Formulation, Methods, and Evaluation. In *arXiv preprint arXiv:2007.15780*.

[50] Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. In *Journal of Artificial Intelligence Research (JAIR)*.
[51] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*.
[52] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *AAAI Conference on Artificial Intelligence (AAAI)*.
[53] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*.
[54] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[55] Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. 2020. A knowledge-enhanced pretraining model for commonsense story generation. In *Transactions of the Association for Computational Linguistics (TACL)*.
[56] Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In *AAAI Conference on Artificial Intelligence (AAAI)*.
[57] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the Unknown Words. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[58] Dandan Guo, Bo Chen, Ruiying Lu, and Mingyuan Zhou. 2020. Recurrent Hierarchical Topic-Guided RNN for Language Generation. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.
[59] Fred X Han, Di Niu, Kunfeng Lai, Weidong Guo, Yancheng He, and Yu Xu. 2019. Inferring Search Queries from Web Documents via a Graph-Augmented Sequence to Attention Network. In *The World Wide Web Conference (WWW)*.
[60] Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[61] Frances Horibe. 1999. Managing knowledge workers: New skills and attitudes to unlock the intellectual capital in your organization. In *John Wiley & Sons*.
[62] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[63] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning (ICML)*.
[64] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, Xiaodan Liang, Lianhui Qin, Haoye Dong, and Eric Xing. 2018. Deep Generative Models with Learnable Knowledge Constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[65] Luyang Huang, Lingfei Wu, and Lu Wang. 2020. Knowledge Graph-Augmented Abstractive Summarization with Semantic-Driven Cloze Reward. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[66] Touseef Iqbal and Shaima Qureshi. 2020. The Survey: Text Generation Models in Deep Learning.. In *Journal of King Saud University-Computer and Information Sciences*.
[67] Hayate Iso, Chao Qiao, and Hang Li. 2020. Fact-based Text Editing. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[68] Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, and Hiroya Takamura. 2019. Learning to Select, Track, and Generate for Data-to-Text. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[69] Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, and Minlie Huang. 2020. Generating Commonsense Explanation by Extracting Bridge Concepts from Reasoning Paths. In *Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and International Joint Conference on Natural Language (AACL-IJCNLP)*.
[70] Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, Xiaoyan Zhu, and Minlie Huang. 2020. Language Generation with Multi-Hop Reasoning on Commonsense Knowledge Graph. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
[71] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. In *arXiv preprint arXiv:2002.00388*.
[72] Hanqi Jin, Tianming Wang, and Xiaojun Wan. 2020. SemSUM: Semantic Dependency Guided Neural Abstractive Summarization. In *AAAI Conference on Artificial Intelligence (AAAI)*.
[73] Amirhossein Kazemnejad, Mohammadreza Salehi, and Mahdieh Soleymani Baghshah. 2020. Paraphrase Generation by Learning How to Edit from Samples. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[74] Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. 2020. Sequential Latent Knowledge Selection for Knowledge-Grounded Dialogue. In *International Conference for Learning Representation (ICLR)*.
[75] Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies (NAACL-HLT).

[76] Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Conference on Empirical Methods in Natural Language Processing (EMNLP).*

[77] Ni Lao, Jun Zhu, Liu Liu, Yandong Liu, and William W Cohen. 2010. Efficient relational learning with hidden variable detection. In *Advances in Neural Information Processing Systems (NeurIPS).*

[78] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. In *Nature*. Nature Publishing Group.

[79] 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 *Annual Meeting of the Association for Computational Linguistics (ACL).*

[80] Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding generation for abstractive text summarization based on key information guide network. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).*

[81] Haoran Li, Junnan Zhu, Jiajun Zhang, Chengqing Zong, and Xiaodong He. 2020. Keywords-guided abstractive sentence summarization. In *AAAI Conference on Artificial Intelligence (AAAI).*

[82] Jingyuan Li and Xiao Sun. 2018. A Syntactically Constrained Bidirectional-Asynchronous Approach for Emotional Conversation Generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP).*

[83] Miao Li, Beihong Jin, et al. 2019. A Topic Augmented Text Generation Model: Joint Learning of Semantics and Structural Features. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).*

[84] Wei Li, Jingjing Xu, Yancheng He, ShengLi Yan, Yunfang Wu, and Xu Sun. 2019. Coherent Comments Generation for Chinese articles with a Graph-to-Sequence Model. In *Annual Meeting of the Association for Computational Linguistics (ACL).*

[85] Rongzhong Lian, Min Xie, Fan Wang, Jinhua Peng, and Hua Wu. 2019. Learning to select knowledge for response generation in dialog systems. In *International Joint Conference on Artificial Intelligence (IJCAI).*

[86] Percy Liang, Michael I Jordan, and Dan Klein. [n.d.]. Learning from measurements in exponential families. In *International Conference on Machine Learning (ICML).*

[87] Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for Multi-Document Summarization. In *International Conference on Computational Linguistics (COLING).*

[88] Hui Lin and Vincent Ng. 2019. Abstractive Summarization: A Survey of the State of the Art. In *AAAI Conference on Artificial Intelligence (AAAI).*

[89] Zehao Lin, Xinjing Huang, Feng Ji, Haiqing Chen, and Yin Zhang. 2019. Task-Oriented Conversation Generation Using Heterogeneous Memory Networks. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).*

[90] Yuanxin Liu, Zheng Lin, Fenglin Liu, Qinyun Dai, and Weiping Wang. 2019. Generating Paraphrase with Topic as Prior Knowledge. In *International Conference on Information and Knowledge Management (CIKM).*

[91] Zhibin Liu, Zheng-Yu Niu, Hua Wu, and Haifeng Wang. 2019. Knowledge aware conversation generation with reasoning on augmented graph. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).*

[92] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP).*

[93] Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems. In *Annual Meeting of the Association for Computational Linguistics (ACL).*

[94] Gideon S Mann and Andrew McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *International Conference on Machine Learning (ICML).*

[95] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Annual Meeting of the Association for Computational Linguistics: System Demonstration (ACL).*

[96] Chuan Meng, Pengjie Ren, Zhumin Chen, Christof Monz, Jun Ma, and Maarten de Rijke. 2020. RefNet: A reference-aware network for background based conversation. In *AAAI Conference on Artificial Intelligence (AAAI).*

[97] Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In *International Conference on Machine Learning (ICML).*

[98] Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International conference on machine learning (ICML).*

[99] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS).*

[100] Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. 2018. Towards Exploiting Background Knowledge for Building Conversation Systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[101] Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[102] Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to Backward and Forward Sequences: A Content-Introducing Approach to Generative Short-Text Conversation. In *Conference on Computational Linguistics: Technical Papers (COLING)*.

[103] Diego Moussallem, Tommaso Soru, and Axel-Cyrille Ngonga Ngomo. 2019. THOTH: Neural Translation and Enrichment of Knowledge Graphs. In *International Semantic Web Conference (ISWC)*.

[104] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Conference on Computational Natural Language Learning (SIGNLL)*.

[105] Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[106] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A Survey on Open Information Extraction. In *International Conference on Computational Linguistics (COLING)*.

[107] Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. In *Transactions of the Association for Computational Linguistics (TACL)*.

[108] Zheng-Yu Niu, Hua Wu, Haifeng Wang, et al. 2019. Knowledge Aware Conversation Generation with Explainable Reasoning over Augmented Graphs. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[109] Liangming Pan, Yuxi Xie, Yansong Feng, Tat-Seng Chua, and Min-Yen Kan. 2020. Semantic Graphs for Generating Deep Questions. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[110] Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text Generation with Exemplar-based Adaptive Decoding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[111] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[112] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[113] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[114] Lianhui Qin, Michel Galley, Chris Brockett, Xiaodong Liu, Xiang Gao, Bill Dolan, Yejin Choi, and Jianfeng Gao. 2019. Conversing by Reading: Contentful Neural Conversation with On-demand Machine Reading. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[115] Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Backpropagation-based Decoding for Unsupervised Counterfactual and Abductive Reasoning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[116] Delai Qiu, Yuanzhe Zhang, Xinwei Feng, Xiangwen Liao, Wenbin Jiang, Yajuan Lyu, Kang Liu, and Jun Zhao. 2019. Machine Reading Comprehension Using Structural Knowledge Graph-aware Network. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[117] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. In *Science China Technological Sciences*.

[118] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *OpenAI Blog*.

[119] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[120] Revanth Gangi Reddy, Danish Contractor, Dinesh Raghu, and Sachindra Joshi. 2019. Multi-Level Memory for Task Oriented Dialogs. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[121] Pengjie Ren, Zhumin Chen, Christof Monz, Jun Ma, and Maarten de Rijke. 2020. Thinking Globally, Acting Locally: Distantly Supervised Global-to-Local Knowledge Selection for Background Based Conversation. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[122] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference (ESWC)*.
[123] Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[124] Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Conference on Machine Translation (WMT)*.
[125] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[126] Lei Shen and Yang Feng. 2020. CDL: Curriculum Dual Learning for Emotion-Controllable Response Generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[127] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[128] Sifatullah Siddiqi and Aditi Sharan. 2015. Keyword and keyphrase extraction techniques: a literature review. In *International journal of Computer Applications (IJCA)*. Foundation of Computer Science.
[129] Haoyu Song, Wei-Nan Zhang, Yiming Cui, Dong Wang, and Ting Liu. 2019. Exploiting persona information for diverse generation of conversational responses. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
[130] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked Sequence to Sequence Pre-training for Language Generation. In *International Conference on Machine Learning (ICML)*.
[131] Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. In *Transactions of the Association for Computational Linguistics (TACL)*.
[132] Zhenqiao Song, Xiaoqing Zheng, Lu Liu, Mu Xu, and Xuan-Jing Huang. 2019. Generating responses with a specific emotion in dialog. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[133] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence (AAAI)*.
[134] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[135] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[136] Bowen Tan, Lianhui Qin, Eric Xing, and Zhiting Hu. 2020. Summarizing Text on Any Aspects: A Knowledge-Informed Weakly-Supervised Approach. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
[137] Jianheng Tang, Tiancheng Zhao, Chenyan Xiong, Xiaodan Liang, Eric Xing, and Zhiting Hu. 2019. Target-Guided Open-Domain Conversation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[138] Yi-Lin Tuan, Yun-Nung Chen, and Hung-yi Lee. 2019. DyKgChat: Benchmarking Dialogue Generation Grounding on Dynamic Knowledge Graphs. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.
[139] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[140] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference for Learning Representation (ICLR)*.
[141] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[142] Hao Wang, Bin Guo, Wei Wu, and Zhiwen Yu. 2020. Towards information-rich, logical text generation with knowledge-enhanced neural models. In *arXiv preprint arXiv:2003.00814*.
[143] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*.
[144] Jian Wang, Junhao Liu, Wei Bi, Xiaojiang Liu, Kejing He, Ruifeng Xu, and Min Yang. 2020. Improving Knowledge-aware Dialogue Generation via Knowledge Base Question Answering. In *AAAI Conference on Artificial Intelligence (AAAI)*.
[145] Kai Wang, Xiaojun Quan, and Rui Wang. 2019. BiSET: Bi-directional Selective Encoding with Template for Abstractive Summarization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
[146] Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
[147] Qingyun Wang, Lifu Huang, Zhiying Jiang, Kevin Knight, Heng Ji, Mohit Bansal, and Yi Luan. 2019. PaperRobot: Incremental Draft Generation of Scientific Ideas. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[148] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*.

[149] Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. 2019. Topic-Guided Variational Auto-Encoder for Text Generation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[150] Wentao Wang, Zhiting Hu, Zichao Yang, Haoran Shi, Frank Xu, and Eric Xing. 2019. Toward Unsupervised Text Content Manipulation. In *arXiv preprint arXiv:1901.09501*.

[151] Wei Wei, Jiayi Liu, Xianling Mao, Guibing Guo, Feida Zhu, Pan Zhou, and Yuchong Hu. 2019. Emotion-aware Chat Machine: Automatic Emotional Response Generation for Human-like Emotional Interaction. In *ACM International Conference on Information and Knowledge Management (CIKM)*.

[152] Xiangpeng Wei, Yue Hu, Luxi Xing, Yipeng Wang, and Li Gao. 2019. Translating with Bilingual Topic Knowledge for Neural Machine Translation. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[153] Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019. Global-to-local memory pointer networks for task-oriented dialogue. In *International Conference for Learning Representation (ICLR)*.

[154] Sixing Wu, Ying Li, Dawei Zhang, Yang Zhou, and Zhonghai Wu. 2020. Diverse and Informative Dialogue Generation with Context-Specific Commonsense Knowledge Awareness. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[155] Sixing Wu, Ying Li, Dawei Zhang, Yang Zhou, and Zhonghai Wu. 2020. TopicKA: Generating Commonsense Knowledge-Aware Dialogue Responses Towards the Recommended Topic Fact. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[156] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. In *IEEE Transactions on Neural Networks and Learning Systems*.

[157] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[158] Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[159] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model. In *International Conference of Learning Representation (ICLR)*.

[160] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[161] Hongcai Xu, Junpeng Bao, and Gaojie Zhang. 2020. Dynamic Knowledge Graph-based Dialogue Generation with Improved Adversarial Meta-Learning. In *arXiv preprint arXiv:2004.08833*.

[162] Jun Xu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. 2020. Conversational Graph Grounded Policy Learning for Open-Domain Conversation Generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[163] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *International Conference for Learning Representation (ICLR)*.

[164] Minghong Xu, Piji Li, Haoran Yang, Pengjie Ren, Zhaochun Ren, Zhumin Chen, and Jun Ma. 2020. A Neural Topical Expansion Framework for Unstructured Persona-oriented Dialogue Generation. In *European Conference on Artificial Intelligence (ECAI)*.

[165] Jian Yang, Shuming Ma, Dongdong Zhang, Zhoujun Li, and Ming Zhou. 2020. Improving Neural Machine Translation with Soft Template Prediction. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[166] Min Yang, Qiang Qu, Wenting Tu, Ying Shen, Zhou Zhao, and Xiaojun Chen. 2019. Exploring human-like reading strategy for abstractive text summarization. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[167] Pengcheng Yang, Lei Li, Fuli Luo, Tianyu Liu, and Xu Sun. 2019. Enhancing Topic-to-Essay Generation with External Commonsense Knowledge. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[168] Shuoheng Yang, Yuxin Wang, and Xiaowen Chu. 2020. A Survey of Deep Learning Techniques for Neural Machine Translation. In *arXiv preprint arXiv:2002.07526*.

[169] Semih Yavuz, Abhinav Rastogi, Guan-Lin Chao, Dilek Hakkani-Tür, and Amazon Alexa AI. 2019. DEEPCOPY: Grounded Response Generation with Hierarchical Pointer Networks. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

[170] Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[171] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[172] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[173] Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2020. Grounded conversation generation as guided traverses in commonsense knowledge graphs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[174] Jian Zhang, Liangyou Li, Andy Way, and Qun Liu. 2016. Topic-Informed Neural Machine Translation. In *International Conference on Computational Linguistics: Technical Papers (COLING)*.

[175] Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017. Prior Knowledge Integration for Neural Machine Translation using Posterior Regularization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[176] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too?. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[177] Yang Zhao, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2020. Knowledge Graphs Enhanced Neural Machine Translation. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[178] Yinhe Zheng, Rongsheng Zhang, Minlie Huang, and Xiaoxi Mao. 2020. A Pre-Training Based Personalized Dialogue Generation Model with Persona-Sparse Data.. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[179] Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[180] Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[181] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *Conference on Natural Language Processing and Chinese Computing (NLPCC)*.

[182] Xianda Zhou and William Yang Wang. 2018. MojiTalk: Generating Emotional Responses at Scale. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[183] Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2020. Boosting Factual Correctness of Abstractive Summarization with Knowledge Graph. In *arXiv preprint arXiv:2003.08612*.

[184] Jun Zhu, Ning Chen, and Eric P Xing. 2014. Bayesian inference with posterior regularization and applications to infinite latent SVMs. In *Journal of Machine Learning Research (JMLR)*.

[185] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. In *Journal of the royal statistical society*. Wiley Online Library.