KGR^4: Retrieval, Retrospect, Refine and Rethink for Commonsense Generation

Xin Liu^{1,3} Dayiheng Liu^2 Baosong Yang^2 Haibo Zhang^2 Junwei Ding^2

Wenqing Yao^2 Weihua Luo^2 Haiying Zhang^1 Jinsong Su^{1,3,4*}

^{1}School of Informatics, Xiamen University ^{2}Alibaba Group

^{3} Key Laboratory of Digital Protection and Intelligent Processing of Intangible

Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism

^{4} Pengcheng Lab, Shenzhen

liuxin@stu.xmu.edu.cn

{liudayiheng.ldyh, yangbaosong.ybs, zhanhui.zhb, djw99219,

wenqing.ywq, weihua.luowh}@alibaba-inc.com

{zhang2002,jssu}@xmu.edu.cn

Abstract

Generative commonsense reasoning requires machines to generate sentences describing an everyday scenario given several concepts, which has attracted much attention recently. However, existing models cannot perform as well as humans, since sentences they produce are often implausible and grammatically incorrect. In this paper, inspired by the process of humans creating sentences, we propose a novel **K**nowledge-enhanced Commonsense **G**eneration framework, termed KGR^4, consisting of four stages: **R**etrieval, **R**etrospect, **R**efine, **R**ethink. Under this framework, we first perform retrieval to search for relevant sentences from external corpus as the prototypes. Then, we train the generator that either edits or copies these prototypes to generate candidate sentences, of which potential errors will be fixed by an autoencoder-based refiner. Finally, we select the output sentence from candidate sentences produced by generators with different hyper-parameters. Experimental results and in-depth analysis on the CommonGen benchmark strongly demonstrate the effectiveness of our framework. Particularly, KGR^4 obtains 33.56 SPICE points in the official leaderboard, outperforming the previously-reported best result by 2.49 SPICE points and achieving state-of-the-art performance. We release the code at https://github.com/DeepLearnXMU/KGR-4.

1   Introduction

Recently, integrating commonsense knowledge into artificial intelligence models has become increasingly attractive to researchers. To assess the ability of these models in understanding the commonsense knowledge from our daily life, various tasks (Zellers et al. 2018; Talmor et al. 2019a; Zellers et al. 2019; Lin et al. 2020) for commonsense reasoning have been proposed. Typically, SWAG (Zellers et al. 2018), HellaSWAG (Zellers et al. 2019), and CommonsenseQA (Talmor et al. 2019a) are designed to infer an upcoming event by selecting one of the listed choices. However, few of them focus on commonsense reasoning in a generative manner, which is considered as a basic ability of hu-

Concepts: {*hand, sink, wash, soap*}	
Outputs: The girl uses *soap* to *wash* her *hands* at the *sink*.	
GPT-2: *hands washing soap* on the *sink*.	
BERT-Gen: a woman *washes* her *hands* with **a *sink* of *soaps***.	
UniLM: *hands washing soap* in the *sink*	
BART: a man is *washing* his *hands* **in a *sink* a *sink***.	
T5: *hand washed* with *soap* in a *sink*.	
Ours: A man is *washing* his *hands* with *soap* in a *sink*.	

Table 1: Sentences produced by commonly-used pretrained models given some concepts. Those sentences generated by existing pretrained models are either implausible (e.g. *a sink of soaps*) or suffer from the repetition problem (e.g. *in a sink a sink*), while ours generates a more natural sentence.

man beings (Moore 2013). To deal with this issue, Lin et al. (2020) explore CommonGen, which requires the model to produce the sentence describing the daily life scenario given some concepts. As shown in the first two lines of Table 1, given concepts {*hand, sink, wash, soap*}, this task aims to generate a coherent sentence covering all of them, e.g. "*The girl uses soap to wash her hands at the sink*". Compared with previous tasks, CommonGen is able to better evaluate the generative commonsense reasoning ability of each model, thus attracting much attention recently.

However, CommonGen remains a difficult task due to the following challenges: 1) Generated sentences should be consistent with commonsense knowledge; 2) The model is required to possess the compositional generalization ability, so that it can deal with unseen combinations of concepts. From Table 1, we can observe that the sentences generated by commonly-used pretrained models are either not in line with commonsense knowledge (i.e., "*a sink of soaps*" produced by BERT-Gen (Bao et al. 2020)) or suffer from the repetition problem (i.e., "*in a sink a sink*" generated by BART (Lewis et al. 2020)).

To strengthen the ability of generative commonsense reasoning, many researchers resort to introducing external knowledge to refine pretrained models. Typically, Liu et al. (2021) exploit the knowledge graph ConceptNet (Liu and Singh 2004) to refine the BART-based model. However,

*Jinsong Su is the corresponding author. This work was done when Xin Liu was interning at DAMO Academy, Alibaba Group.
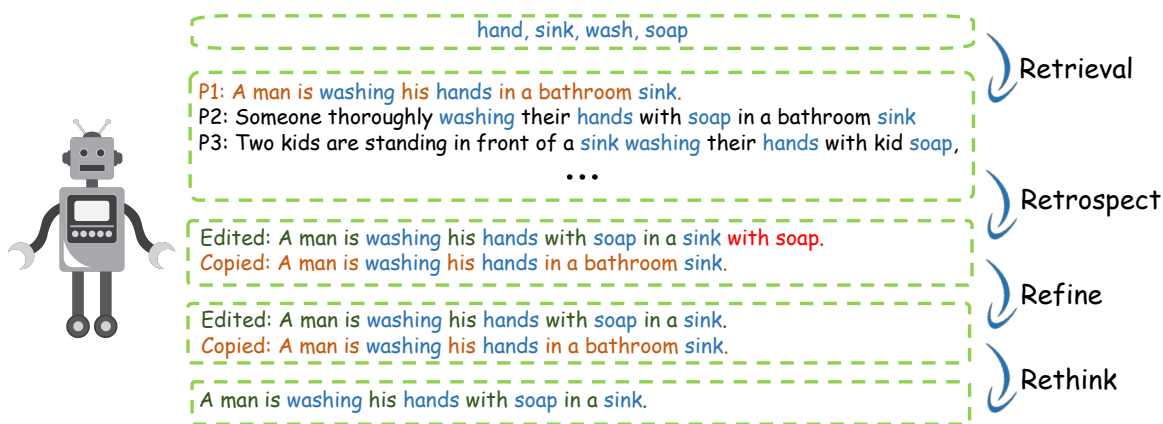
Figure 1: The process of our framework generating the sentence for input concepts *hand, sink, wash, soap*, which are marked in blue. P1, P2, P3 denote the prototypes retrieved from external corpora. The sentences followed by *Edited* are the sentence generated by editing prototypes, while those followed by *Copied* are copied from prototypes. The red chunk consists of repeated words and will be deleted at the refine stage. Our framework selects the better sentence as the final output at the rethink stage.

knowledge graphs are usually human-annotated and might fail to cover all commonsense in daily life. Therefore, the performance of this line of work would be limited. More recently, Fan et al. (2020) and Wang et al. (2021) introduce retrievers to search auxiliary information from external plain sentences, which contains enormous daily scenarios. Intuitively, the implicit commonsense knowledge within plain sentences is more abundant than that in human-annotated knowledge bases. In this work, we extend this idea and design a novel generation framework.

Suppose a user is asked to write a sentence mentioning the given concepts: they may attempt to search for an ideal sentence that can be directly copied as the answer. But if the searched sentence does not mention all concepts, they may further edit it to meet the requirements, where the potential errors should be corrected. Finally, they might consider several candidate sentences through the above process and pick the most satisfying one. Inspired by such process, in this paper, we propose a **K**nowledge-enhanced Commonsense **G**eneration framework, termed KGR[4]. As illustrated in Figure 1, our framework consists of four steps: 1) **Retrieval**: We first perform retrieval to obtain prototypes for generation, where candidates are roughly retrieved by concept mapping and then a trainable scorer is used to select satisfactory candidates as prototypes. 2) **Retrospect**: A BART-based seq2seq model is employed as the generator, which exploits prototypes for better sentence generations. To encourage the generator to edit or copy prototypes as output, we propose *retrospective training* and *retrospective augmentation* to enhance the training of the generator. 3) **Refine**: At this stage, we train a BART-based (Lewis et al. 2020) refiner to fix errors within generated sentences. 4) **Rethink**: Finally, we reuse the previously-trained scorer to select the best sentence from those produced by various generators.

To investigate the effectiveness of our framework, we conduct extensive experiments on the CommonGen benchmark, where experimental results and in-depth analysis demon-

strate the superiority of our framework. Specifically, KGR[4] significantly surpasses the previous best result (Wang et al. 2021) on the CommonGen v1.0 test set (34.40 vs. 39.70 SPICE points (Anderson et al. 2016)). Besides, on the official test set (v1.1) of the leaderboard, KGR[4] achieves 33.56 SPICE, outperforming the previous best model by 2.48 SPICE points and setting a new state-of-the-art.

2 Related Work

Our related work mainly includes the studies in two aspects: commonsense reasoning and utilizing commonsense knowledge in natural language generation (NLG).

Commonsense Reasoning Recently, a series of works have been proposed to investigate the machine commonsense reasoning ability. Typically, SWAG (Zellers et al. 2018), CODAH (Chen et al. 2019), HellaSWAG (Zellers et al. 2019), CommonsenseQA (Talmor et al. 2019b), Atomic (Sap et al. 2019) are proposed to assess the commonsense reasoning ability of models for question answering. To enhance such ability, Rajani et al. (2019) collect human explanations in the form of natural language for better commonsense reasoning. Lin et al. (2019) design an explainable inference framework to utilize external commonsense graphs for commonsense question answering. Besides, Shwartz et al. (2020) study pretrained language models as an alternative of external knowledge provider to benefit commonsense question answering. Furthermore, since writing sentence as natural as humans is a desired ability for machines, researchers also contributed to incorporating commonsense in NLG.

Utilizing Commonsense Knowledge in NLG To assess the machine commonsense reasoning ability in a generative manner, Lin et al. (2020) propose CommonGen which asks models to generate a reasonable and fluency sentence, which mentions all the given concepts. To enhance generative commonsense reasoning, many researchers devote
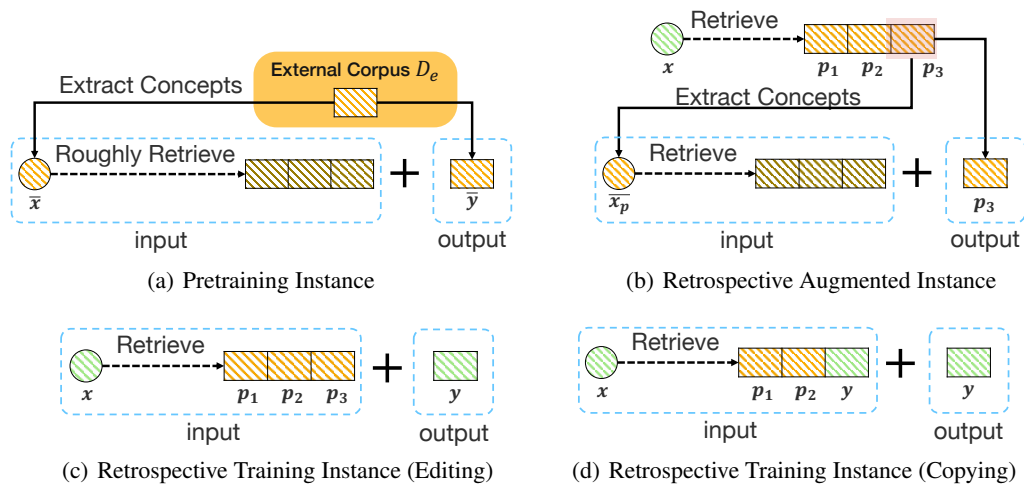
Figure 2: Four types of training instances at the retrospect stage. (a) A pretraining instance, of which target sentence $\bar{y}$ is extracted from external corpus $\mathcal{D}_e$ and input concepts $\bar{x}$ are extracted from $\bar{y}$; (b) A retrospective augmented instance, where both pseudo concept set $\bar{x}_p$ and pseudo target sentence p_3 are extracted from the prototype of external corpus $\mathcal{D}_e$; (c) A retrospective training instance encouraging the generator to edit prototypes; (d) A retrospective training instance encouraging the generator to copy prototypes, where one of its prototypes (randomly chosen) is replaced with the target sentence.

to incorporating existing knowledge bases into pretrained models. For instance, Liu et al. (2021) additionally introduce an encoder and a decoder customized to exploit the knowledge graph related to the given concepts. Zhou et al. (2021) infuse concept-centric commonsense knowledge into pretrained models via generative and contrastive objectives. Nevertheless, the performance of these models is limited due to the low coverage and quality of used knowledge bases. Besides, many researchers focus on utilizing implicit commonsense knowledge within plain sentences and adopt the retrieval-and-generation paradigm. Typically, Fan et al. (2020) equip the BART-based generation model with an enhanced knowledge injection module, which exploits the commonsense knowledge extracted from retrieved sentences. Furthermore, Wang et al. (2021) propose a trainable retriever to search auxiliary sentences for given concepts, and concatenate them with concepts as input for T5 (Raffel et al. 2020a).

Obviously, our framework is an extension of retrieval-and-generation methods. We additionally propose three novel stages. At the retrospect stage, the generator produces sentences by copying or editing the retrieved prototypes. Generated sentences will be refined by our refiner to fix the errors at the refine stage, and the best sentence will be picked among candidate sentences produced by generators with different hyper-parameters at the rethink stage.

3 Our Framework

In this section, by simulating the process of humans writing sentences, we propose a novel knowledge-enhanced commonsense generation framework, which aims to generate a high-quality target sentence y containing all the concepts in set $x = \{x_1, x_2, ..., x_N\}$, where x_i denotes the i-th concept. As illustrated in Figure 1, our framework mainly consists of

four stages: Retrieval, Retrospect, Refine and Rethink. In the following subsections, we describe these stages in detail.

3.1 Retrieval

Intuitively, when asked to write a sentence with given concepts, humans will always call to mind the scenarios associated with these concepts, then conceive logical sentences. Similarly, our framework retrieves prototypes related to the given concepts from external corpora, which contain abundant scenario knowledge, then uses these prototypes as auxiliary information for the commonsense generation.

We firstly use the concept mapping (Wang et al. 2021) and roughly search candidate prototypes for the concept set x from an external corpus $\mathcal{D}_e$, which consists of caption sentences describing the daily scenario. Then, we train a RoBERTa-based binary classifier as a scorer to semantically evaluate candidate prototypes. The scorer takes the concatenation of x and a candidate prototype p' as input and outputs a score $f_s(x, p')$ indicating whether p' is related to x. To optimize the scorer, we construct a temporary training set from $\mathcal{D}_e$ and the CommonGen training set $\mathcal{D}$, which consists of $\langle x, y \rangle$. More specifically, for each concept set x from $\mathcal{D}$, we take its corresponding sentence y to construct a positive sample and randomly select sentences from $\mathcal{D}_e$ to produce negative samples. Finally, we apply this scorer to semantically evaluate candidate prototypes for each concept set. As implemented in Wang et al. (2021), we select the top-3 scored candidate prototypes as the final prototype set.

3.2 Retrospect

We then construct a generator to exploit prototypes for sentence generations. Note that given the prototypes, humans might glance at these prototypes before sentence writing. If one of these prototypes is good enough, they could directly

copy it as output. Otherwise, they might write the sentence by editing prototypes. Thus, we believe both editing and copying prototypes are crucial for generating high-quality sentences. For that purpose, we propose two novel strategies to enhance the training of the generator.

Generator Following previous studies (Liu et al. 2021; Fan et al. 2020), we adopt BART-large (Lewis et al. 2020) to establish our generator, which takes the concept set and retrieved prototypes as input and generates a target sentence mentioning all concepts. To make the model fully absorb the commonsense knowledge from the external corpus $\mathcal{D}_e$, we pretrain the generator using a large-scale pretraining dataset $\mathcal{D}_{pt}$ constructed from $\mathcal{D}_e$. In specific, the construction steps of $\mathcal{D}$ are as follows: taking the sentence $\bar{y}$ in $\mathcal{D}_e$ as the target sentence, we use spaCy * to perform POS tagging on $\bar{y}$, and sample several lemmatized Verbs, Nouns, and Proper Nouns as the pseudo concepts $\bar{x}$. We pair $\bar{x}$ with $\bar{y}$ to form a pretraining instance (Figure 2(a)). Considering the training efficiency of the generator, we only roughly retrieve the prototypes via concept matching during pretraining. Formally, we use $\mathcal{D}_{pt}$ to pretrain our generator in the following way:

$$\mathcal{L}_G^{pt} = -\sum_{t=1}^{|\bar{y}|} \log p(\bar{y}_t|\bar{x}, \{\bar{p}_1, \bar{p}_2, \bar{p}_3\}, \bar{y}_{<t}), \qquad (1)$$

where $\bar{p}_*$ is the roughly retrieved prototypes, and $\bar{y}_t$ denotes the t-th token of $\bar{y}$.

Afterwards, we propose two novel strategies to finetune the generator so as to enhance the training of the generator. Since the retrieved prototypes related to the given concepts are usually human-created, they are natural and coherent. We argue that letting the generator learn more about how to generate such prototypes will help the generator better capture the meaning of these concepts, and further benefits the subsequent commonsense generation. Thus, we propose **retrospective augmentation** strategy to expand $\mathcal{D}$ into a new training set $\mathcal{D}_{ft}$. As shown in Figure 2(b), following the concepts extracting process mentioned before, we extract pseudo concepts $\bar{x}_p$ from the prototype p_3, and pair $\bar{x}_p$ with p_3 to form a retrospective augmented instance, which will be added into $\mathcal{D}_{ft}$.

Most importantly, we propose **retrospective training** strategy to enhance the editing and copying ability of our generator. Unlike the previous pretraining only based on the MLE training objective, our training objective involves two loss items. Formally, given a training instance $\langle x, y \rangle$ from $\mathcal{D}_{ft}$ and its retrieved prototypes $\{p_1, p_2, p_3\}$, we define the joint training objective as follows:

$$\mathcal{L}_G^{ft} = (1-\lambda)\mathcal{L}_{edit} + \lambda\mathcal{L}_{copy}, \qquad (2)$$

$$\mathcal{L}_{edit} = -\sum_{t=1}^{|y|} \log p(y_t|x, \{p_1, p_2, p_3\}, y_{<t}), \qquad (3)$$

$$\mathcal{L}_{copy} = -\sum_{t=1}^{|y|} \log p(y_t|x, \{p_1, p_2, y\}, y_{<t}), \qquad (4)$$

---

*https://spacy.io/

where λ is the hyper-parameter to balance $\mathcal{L}_{edit}$ and $\mathcal{L}_{copy}$. $\mathcal{L}_{edit}$ denotes the editing training objective. As illustrated in Figure 2(c), the generator generates the target sentence y by editing the retrieved prototypes $\{p_1, p_2, p_3\}$. $\mathcal{L}_{copy}$ is the copying training objective. We randomly replace one of the prototypes in $\{p_1, p_2, p_3\}$ with the target sentence y, which is illustrated in Figure 2(d). By using $\mathcal{L}_{copy}$, we expect that the generator is able to detect the high-quality prototype and copy it as output.

3.3 Refine

Similar to previous works (Lewis et al. 2020; Fan et al. 2020) focusing on CommonGen, we find that our generator suffers from the *degeneration problem* (Welleck et al. 2020). To ensure these sentences are grammatically correct, we propose a refiner to fix potential errors within generated sentences.

Refiner To better model our refiner, we first analyze the errors within the generated sentences, which can be classified into the following two types: 1) *Repetition error*. The generator sometimes pays too much attention to what it has recently produced (Fan, Lewis, and Dauphin 2018), and thus it tends to generate similar text multiple times. For example, "*in a sink a sink*" shown in Table 1; 2) *Misspelling*. Since the BART-based generator tokenizes the sentences using the same byte-pair encoding as GPT-2 (Radford et al. 2019), some characters and spaces in sentence should be predicted, while the generator occasionally misses them, leading to misspelling words (i.e., "*wash hands*" is incorrectly output as "*wsh hands*" and "*washhands*").

To deal with the errors mentioned above, we construct a BART-based refiner, which takes the the candidate target sentence generated by our generator as input and outputs the corrected target sentence. To this end, we construct training instances based on $\mathcal{D}_e$. Given the sentence $\bar{y}$ from $\mathcal{D}_e$, we generate a perturbed sentence $\hat{y}$ by incorporating above-mentioned errors into $\bar{y}$: either randomly repeat a word sequence of $\bar{y}$ to simulate repetition errors, or randomly remove characters or spaces in $\bar{y}$ to introduce the misspelling words. Then, we require the refiner to recover $\bar{y}$ from $\hat{y}$ via the following auto-encoding training objective:

$$\mathcal{L}_R = -\log p(\bar{y}|\hat{y}) = -\sum_{t=1}^{|\bar{y}|} \log p(\bar{y}_t|\hat{y}, \bar{y}_{<t}). \qquad (5)$$

In this way, our refiner is trained to correct potential errors within candidate sentences, especially repetition and misspelling errors.

3.4 Rethink

The above processes describe in detail how to generate a target sentence for a given concept set. However, this process is not entirely consistent with how humans write sentences: humans usually produce multiple sentences in different ways (copying or writing) and then select the best one from them. To further improve quality of the generated sentences, we propose to employ the generator with different λs to generate multiple target sentences, all of which are then refined by our proposed refiner. Afterwards, we reuse the

Statistics	Train	Validation	Test
#Concept Sets	32,651	993	1,497
-Size = 3	25,020	493	-
-Size = 4	4,240	250	747
-Size = 5	3,391	250	750
#Sentences	67,389	4,018	7,644
Unseen Concepts	-	6.53%	8.97%
Unseen Concept-Pairs	-	96.31%	100.00%
Unseen Concept-Triples	-	99.60%	100.00%

Table 2: The basic statistics of the CommonGen dataset. #Concept is the number of concepts each concept set contains. Unseen concept compositions (i.e., concept, concept-pair, concept-triple) denote the ratio of unseen compositions in the training set, which propose challenges on the generalization of models.

previously-trained scorer (See Section 3.1) to semantically evaluate these target sentences. During this process, we feed the concatenation of the given concept set and the refined sentence to the scorer, and select the final outputted sentence with the highest score.

4 Experiments

4.1 Settings

Dataset and Metrics Following previous studies, we use the CommonGen dataset constructed by Lin et al. (2020). We show the basic statistics of this dataset in Table 2. Note that all concept-pairs and concept-triples of test set are unseen in the training set, which requires models to generalize well on the unseen combinations of concepts.

As described above, our framework involves an external corpus, which serves three purposes: 1) Being the retrieval pool where the prototypes come (See Section 3.1); 2) Constructing pseudo instances for retrospective augmentation and pretraining (See Section 3.2); 3) Establishing training and validation sets to train the refiner (See Section 3.3). We construct this corpus by combining 3M image and video captions of several datasets: Activity (Krishna et al. 2017), MultiNLI (Williams, Nangia, and Bowman 2018), SNLI (Bowman et al. 2015), Vatex (Wang et al. 2019), MSCOCO (Lin et al. 2014) and (Young et al. 2014).

Following Lin et al. (2020), we use `BLEU-4` (Papineni et al. 2002), `CIDEr` (Vedantam, Zitnick, and Parikh 2015), `SPICE` (Anderson et al. 2016) as our evaluation metrics. Since Lin et al. (2020) claim that `SPICE` is most relevant to human evaluation, we use it as our prior metric.

Baselines We compare KGR[4] with several competitive generation models:

- **EKI-BART** (Fan et al. 2020). It is a knowledge-enhanced model based on vanilla BART, which retrieves a prototype for better sentence generation;
- **KG-BART** (Liu et al. 2021). It pretrains BART via a masked concept prediction task, and further leverages the knowledge graph to enhance the encoder and decoder;
- **CLAM** (Zhou et al. 2021). It introduces generative and contrastive objectives into pretrained text generation

models, so as to better learn concept-centric commonsense knowledge;

- **RE-T5** (Wang et al. 2021). It is a T5-based model equipped with a trainable retriever to retrieve prototypes as the auxiliary input.

Moreover, we report the performance of several commonly-used pretrained generation models, including: **GPT-2** (Radford et al. 2019), **BERT-Gen** (Bao et al. 2020), **UniLM** (Dong et al. 2019), **BART** (Lewis et al. 2020), **T5-base** (Raffel et al. 2020b), and **T5-large** (Raffel et al. 2020b). Except T5-base and T5-large that add a prompt into the beginning of input sequence, all other models take the combination of concepts as input and output the description sentence.

Implementation Details At the retrieval stage, we select 3 negative samples for each positive sample. We optimize the RoBERTa-based scorer using the Adam optimizer (Kingma and Ba 2015) with a learning rate of 2e-5 for 3 epochs, and set the batch-size to 32. At the retrospect stage, we pretrain the generator for 80,000 steps using the pseudo instances constructed from the external corpus and then finetune the model parameters for 2,000 steps, where the learning rate of the Adam optimizer is set as 2e-5, and the batch size is 16. In both pretraining and retrospective augmentation, we sample 5 concepts from each sentence. At the refine stage, we construct the training and validation set for refiner from the external corpus. In both training and validation sets, 5% of instances are sampled to produce the perturbed sentences, 50% of the perturbed sentences contain misspelling errors, while the others contain repetition errors. During this process, we remove 1% characters and 10% spaces from the instances containing misspelling errors, and repeat sentence segments to construct instances with repetition errors. We update the parameters of refiner for 2,000 steps and keep the rest hyper-parameters same as the generator. Particularly, we employ early-stopping when training scorer, generator, and refiner.

4.2 Overall Results

Table 3 lists the overall results of various models. We can observe that KGR[4] performs best among all models. Compared with the previous best model RE-T5, KGR[4] surpasses it by 1.955 `BLEU-4` and 0.760 `CIDEr` points. Meanwhile, KGR[4] achieves the highest `SPICE` point 33.564, setting a new SOTA on the official leaderboard[†]. Please note that both architectures of EKI-BART and KG-BART are modified to exploit external knowledge, while our framework is completely data-driven, independent on specific model architecture. Thus, our framework can be directly applied to any task-oriented models, such as EKI-BART and KG-BART, to gain further improvements.

Performance on Instances with Various Difficulties Following Lin et al. (2020), we extract 5-size concept-sets (each concept-set consists of five concepts) from the test set, and classify them into 3 categories: **easy**, **normal**,

---

[†]https://inklab.usc.edu/CommonGen/leaderboard.html

Model	BLEU-4(lb)	CIDEr(lb)	**SPICE(lb)**	SPICE(v1.0)
GPT-2 (Radford et al. 2019)	26.833	12.187	23.567	25.90
BERT-Gen (Bao et al. 2020)	23.468	12.606	24.822	27.30
UniLM (Dong et al. 2019)	30.616	14.889	27.429	30.20
BART (Lewis et al. 2020)	31.827	13.976	27.995	30.60
T5-base (Raffel et al. 2020b)	18.546	9.399	19.871	22.00
T5-large (Raffel et al. 2020b)	31.962	15.128	28.855	31.60
EKI-BART (Fan et al. 2020)	35.945	16.999	29.583	32.40
KG-BART (Liu et al. 2021)	33.867	16.927	29.634	32.70
CALM(T5-base) (Zhou et al. 2021)	-	-	-	33.00
RE-T5 (Wang et al. 2021)	40.863	17.663	31.079	34.30
KGR^4	**42.818**	**18.423**	**33.564**	**39.70**

Table 3: Experimental results on the CommonGen benchmark. *(lb) means the results are shown on the official leaderboard. *(v1.0) indicates the evaluation using the old evaluation protocol. Please note that SPICE is our most important metric.
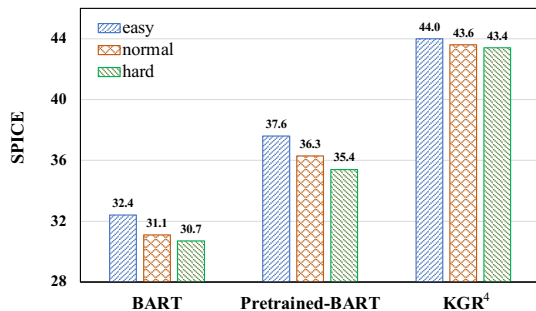

Figure 3: SPICE points on test sets with various difficulties.

model	SPICE(v1.0)
BART	30.60
+pretraining	33.10
+retrieval	36.60
+retrospective training	38.30
+retrospective augmentation	39.20
+refine	39.40
+rethink	**39.70**

Table 4: Ablation study of KGR^4.

hard, representing difficulties of generating sentences about them. Here we estimate the difficulty of each concept-set based on the following fact: if there are more concepts directly connected on ConceptNet, then it is easier to write sentences about them. For each 5-size concept-set, there are 10 concept pairs (pick two among five concepts). We count the one-hop connections of concept pairs that are directly connected on ConceptNet, and then empirically divide concept-sets into **hard|normal|easy** categories if they contain [0,2]|[3,5]|[6,10] one-hop connections.

We illustrate the performance of BART, BART pretrained using the external corpus (Pretrained-BART), and KGR^4 on the test sets with various difficulties in Figure 3. We can find that our framework gains the best results on all test sets, strongly demonstrating its superiority. Besides, it is interesting to observe that our framework reaches the lowest performance decline when switching the test set from the easy to the hard one. This indicates our framework can better deal with all the given concepts regardless of their difficulties.

4.3 Analysis

We further conduct in-depth analyses to investigate the following problems: **Q1**: Do all strategies and stages of KGR^4 take effects? **Q2**: How the hyper-parameter λ affects the generator? **Q3**: Can the refine stage alleviate repetition and misspelling errors?

Ablation Study To explore the effectiveness of different stages and strategies, we report the performance of different variants of our framework.

- **BART**. It is a BART-large model directly supervised by the CommonGen training set.

- **+pretraining**. It is the variant firstly pretrained using the pretraining instances constructed from the external corpus, and then finetuned using the CommonGen training set.

- **+retrieval**. To construct this variant, we further apply retrieval during both pretraining and finetuning for the generator (See Section 3.1).

- **+retrospective training**. In this variant, the pretrained generator is finetuned via our proposed retrospective training strategy using the CommonGen training set (See Section 3.2).

- **+retrospective augmentation**. The generator of this variant is further enhanced with retrospective augmentation strategy.

- **+refine**. It means the generated sentences are further refined through the refine stage.

- **+rethink**. indicates the output sentences are selected among predicted sentences of models using various λs.

From Table 4, we can observe that all our stages and strategies take effects in our framework. Particularly, when sequentially applying our retrospective training and retrospective augmentation strategies, our framework achieves 1.70 and 0.90 SPICE point improvements over its previous version, respectively. These results strongly demonstrate the effectiveness of these two strategies.
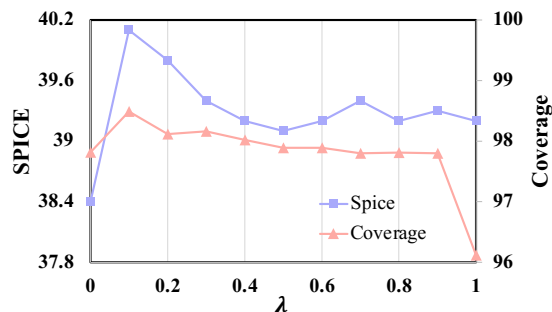
Figure 4: `SPICE`(v1.0) and `Coverage` points computed on CommonGen test set with various λs.
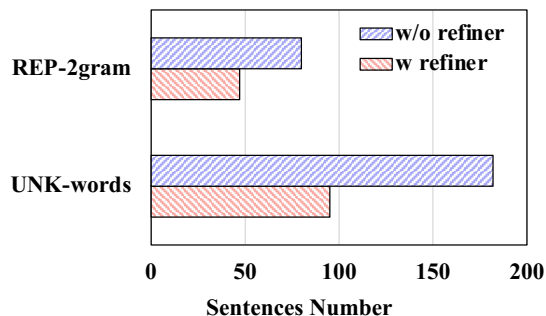

Figure 5: REP-2gram, and UNK-words without (w/o) and with (w) our refiner. Our refiner significantly reduces the numbers of sentences containing two types of errors. w/o refiner is the variant +*retrospective augmentation* mentioned in ablation study.

Effect of Hyper-parameter λ As described in Section 3.2, our generator introduces an important hyper-parameter λ to control the joint training objective listed (See Section 2), which has a crucial effect on the performance of our framework. Thus, we vary λ from 0 to 1 with an increment of 0.1 at each step, and inspect the performance of our framework using different λs. In addition to our prior metric `SPICE`, we introduce another metric `Coverage` that is defined as the average percentage of input concepts occurring within output target sentences.

As shown in Figure 4, our framework always achieves satisfactory performance with any λ in terms of `SPICE` and `Coverage`. Note that when λ is set to 0, our framework obtains 38.4 `SPICE` points, which is significantly inferior to those of other settings, indicating that editing retrieved prototypes regardless of copying may fail to produce high-quality sentences. Meanwhile, we find that the `Coverage` value of our framework drops sharply when λ increases from 0.9 to 1. The underlying reason is that the generator with $\lambda = 1$ tends to copy rather than edit prototypes, while the copied one may not mention all the given concepts, leading to the lower `coverage`. Particularly, when setting λ to 0.1, our framework reaches the highest `SPICE` and `Coverage` values, demonstrating the effectiveness of our retrospect stage.

w/o refiner:	
A *dog splashes* through a *puddle* of *water* <u>in a *puddle*</u> in the *rain* .	
w refiner:	
A *dog splashes* through a *puddle* of *water* in the *rain* .	
w/o refiner:	
Bearded <u>manin</u> white shirt *demonstrates steps* to *tying necktie*.	
w refiner:	
Bearded man in white shirt *demonstrates steps* to *tying necktie*.	

Table 5: Two examples of sentences produced by our framework without and with the refine stage. Concepts appealing in sentences are marked in *italics*. Erroneous words are underlined.

Impact of the Refine Stage   To quantify impact of our refiner, we conduct statistics on the two kinds of errors occurring within the sentences generated by our refiner. Concretely, we count the number of sentences containing repeated n-grams to measure repetition errors (**REP-ngram**). Besides, we roughly consider the output words, which are unseen in both the CommonGen dataset and the external corpus, as misspelling ones. We count the number of sentences containing those words (**UNK-words**). The results of two metrics without and with our refine stage are shown in Figure 5. Obviously, through refine stage, both types of errors are significantly reduce.Specifically, 40% of the repetition and 50% of the misspelling errors are corrected at this stage, strongly demonstrating the effectiveness of our refiner.

Table 5 lists two examples of the sentences without and with our refiner. In the first example, we can find that the phrase "*a puddle*" is repeated before the refine stage. By contrast, our refiner detects this error and removes the repeated phrase as well as the word "*in*", generating a more plausible sentence. In the second example, the space between "*man*" and "*in*" is missed, while our refiner not only adds it but also keeps the rest of the sentence unchanged.

5   Conclusion

In this paper, we have proposed KGR[4], a commonsense generation framework. Our framework mainly separates four key stages, i.e., retrieval, retrospect, refine, and rethink, to imitate corresponding human behaviors in writing. On the commonly-used dataset, KGR[4] outperforms several competitive baselines, setting a new state of the art on the official board. Besides, our study suggests following points: 1) Each component in our framework has its unique function in commonsense generation, and worth to be further explored; 2) The retrieval and retrospect stages play more crucial roles on improving the commonsense reasoning ability; and 3) The refine and rethink stages enable our framework to proofread the generated candidates. In the future, we plan to explore variational models (Zhang et al. 2016a,b; Su et al. 2018a,b) and bidirectional decoding (Zhang et al. 2018; Su et al. 2019) to refine our framework. Besides, we will generalize KGR[4] to other conditional text generation tasks.

Acknowledgments

References

Anderson, P.; Fernando, B.; Johnson, M.; and Gould, S. 2016. SPICE: Semantic Propositional Image Caption Evaluation. In *ECCV 2016*.

Bao, H.; Dong, L.; Wei, F.; Wang, W.; Yang, N.; Liu, X.; Wang, Y.; Gao, J.; Piao, S.; Zhou, M.; and Hon, H.-W. 2020. UniLMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training. In *ICML 2020*.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *EMNLP 2015*.

Chen, M.; D'Arcy, M.; Liu, A.; Fernandez, J.; and Downey, D. 2019. CODAH: An Adversarially Authored Question-Answer Dataset for Common Sense. *CoRR*, abs/1904.04365.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H.-W. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *NeurIPS 2019*.

Fan, A.; Lewis, M.; and Dauphin, Y. N. 2018. Hierarchical Neural Story Generation. In *ACL 2018*.

Fan, Z.; Gong, Y.; Wei, Z.; Wang, S.; Huang, Y.; Jiao, J.; Huang, X.; Duan, N.; and Zhang, R. 2020. An Enhanced Knowledge Injection Model for Commonsense Generation. In *COLING 2020*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR 2015*.

Krishna, R.; Hata, K.; Ren, F.; Fei-Fei, L.; and Carlos Niebles, J. 2017. Dense-Captioning Events in Videos. In *ICCV 2017*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL 2020*.

Lin, B. Y.; Chen, X.; Chen, J.; and Ren, X. 2019. KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. In *EMNLP-IJCNLP 2019*.

Lin, B. Y.; Zhou, W.; Shen, M.; Zhou, P.; Bhagavatula, C.; Choi, Y.; and Ren, X. 2020. CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning. In *EMNLP 2020*.

Lin, T.; Maire, M.; Belongie, S. J.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common Objects in Context. In *ECCV 2014*.

Liu, H.; and Singh, P. 2004. ConceptNet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4).

Liu, Y.; Wan, Y.; He, L.; Peng, H.; and Yu, P. S. 2021. KG-BART: Knowledge Graph-Augmented BART for Generative Commonsense Reasoning. In *AAAI 2021*.

Moore, C. 2013. *The development of commonsense psychology*. Psychology Press.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL 2002*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020a. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020b. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21.

Rajani, N. F.; McCann, B.; Xiong, C.; and Socher, R. 2019. Explain Yourself! Leveraging Language Models for Commonsense Reasoning. In *ACL 2019*.

Sap, M.; Bras, R. L.; Allaway, E.; Bhagavatula, C.; Lourie, N.; Rashkin, H.; Roof, B.; Smith, N. A.; and Choi, Y. 2019. ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning. In *EAAI 2019*.

Shwartz, V.; West, P.; Bras, R. L.; Bhagavatula, C.; and Choi, Y. 2020. Unsupervised Commonsense Question Answering with Self-Talk. In *EMNLP 2020*.

Su, J.; Wu, S.; Xiong, D.; Lu, Y.; Han, X.; and Zhang, B. 2018a. Variational Recurrent Neural Machine Translation. In *AAAI 2018*.

Su, J.; Wu, S.; Zhang, B.; Wu, C.; Qin, Y.; and Xiong, D. 2018b. A neural generative autoencoder for bilingual word embeddings. *Inf. Sci.*, 424.

Su, J.; Zhang, X.; Lin, Q.; Qin, Y.; Yao, J.; and Liu, Y. 2019. Exploiting reverse target-side contexts for neural machine translation via asynchronous bidirectional decoding. *Artif. Intell.*, 277.

Talmor, A.; Herzig, J.; Lourie, N.; and Berant, J. 2019a. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *NAACL-HLT 2019*.

Talmor, A.; Herzig, J.; Lourie, N.; and Berant, J. 2019b. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *NAACL-HLT 2019*.

Vedantam, R.; Zitnick, C. L.; and Parikh, D. 2015. CIDEr: Consensus-based image description evaluation. In *CVPR 2015*.

Wang, H.; Liu, Y.; Zhu, C.; Shou, L.; Gong, M.; Xu, Y.; and Zeng, M. 2021. Retrieval Enhanced Model for Commonsense Generation. In *Findings of ACL/IJCNLP 2021*.

Wang, X.; Wu, J.; Chen, J.; Li, L.; Wang, Y.; and Wang, W. Y. 2019. VaTeX: A Large-Scale, High-Quality Multilingual Dataset for Video-and-Language Research. In *ICCV 2019*.

Welleck, S.; Kulikov, I.; Roller, S.; Dinan, E.; Cho, K.; and Weston, J. 2020. Neural Text Generation With Unlikelihood Training. In *ICLR 2020*.

Williams, A.; Nangia, N.; and Bowman, S. R. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *NAACL-HLT 2018*.

Young, P.; Lai, A.; Hodosh, M.; and Hockenmaier, J. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Trans. Assoc. Comput. Linguistics*, 2.

Zellers, R.; Bisk, Y.; Schwartz, R.; and Choi, Y. 2018. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *EMNLP 2018*.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *ACL 2019*.

Zhang, B.; Xiong, D.; Su, J.; Duan, H.; and Zhang, M. 2016a. Variational Neural Machine Translation. In *EMNLP 2016*.

Zhang, B.; Xiong, D.; Su, J.; Liu, Q.; Ji, R.; Duan, H.; and Zhang, M. 2016b. Variational Neural Discourse Relation Recognizer. In *EMNLP 2016*.

Zhang, X.; Su, J.; Qin, Y.; Liu, Y.; Ji, R.; and Wang, H. 2018. Asynchronous Bidirectional Decoding for Neural Machine Translation. In *AAAI 2018*.

Zhou, W.; Lee, D.; Selvam, R. K.; Lee, S.; and Ren, X. 2021. Pre-training Text-to-Text Transformers for Concept-centric Common Sense. In *ICLR 2021*.