Improve Diverse Commonsense Generation by Enhancing Subgraphs

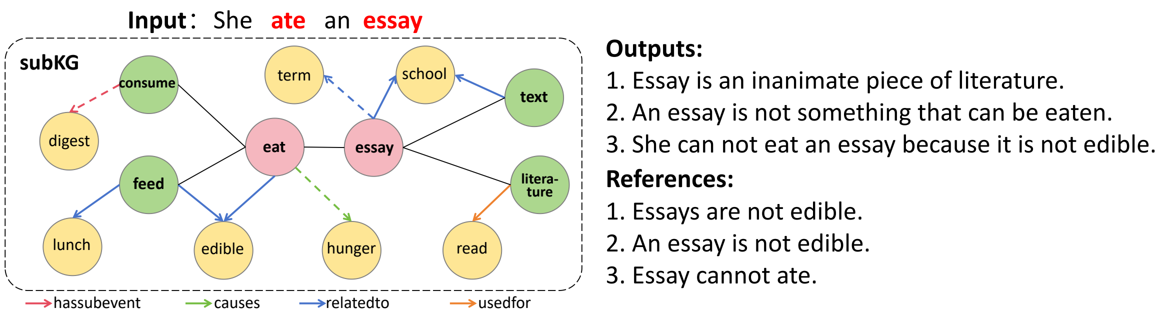
Jianman Tan TANJIANMAN@E.GZHU.EDU.CN and **Shuo Yang** * YANGSHUO@GZHU.EDU.CN
School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China

Editors: Vu Nguyen and Hsuan-Tien Lin

Abstract

Commonsense reasoning (CSR) requires rationale beyond the explicit knowledge mentioned in the context. Many existing methods use knowledge graphs (KGs) to generate rationale as additional evidence for CSR. However, rationale extracted from KGs (e.g., ConceptNet) often includes irrelevant information, which easily introduces noise and affects the evidential quality generated. Similar to brainstorming to generate diverse ideas, we introduce a synonym expansion method to expand input concepts, ultimately constructing a task-relevant knowledge subgraph. Additionally, we propose a pruning model that learns to score and prune the knowledge subgraph, removing parts that are not directly related to the input context. The proposed method improves the quality and diversity of rationale, which benefits generative commonsense reasoning tasks. Experiments on two datasets validated the effectiveness of our method, which demonstrates comparable performance with existing methods.

Keywords: Commonsense reasoning, Knowledge graph pruning, Concept expansion



Outputs:
1. Essay is an inanimate piece of literature.
2. An essay is not something that can be eaten.
3. She can not eat an essay because it is not edible.

References:
1. Essays are not edible.
2. An essay is not edible.
3. Essay cannot ate.

Figure 1: An example from ComVE (Wang et al., 2020). By adding synonyms and trimming the subgraph, we obtain a subgraph containing important information. Dotted lines indicate trimmed triples, while black solid lines connect the original sentence concepts and the synonym concepts.

* Corresponding author.

1. Introduction

Applying knowledge graphs to text generation tasks is a common and effective method in the domain of natural language generation (NLG). Recent studies have shown that integrating commonsense knowledge can enhance the performance of generation models, enabling them to generate text more accurately during reasoning processes (Wang et al., 2022; Nie et al., 2023; Wang et al., 2024). An important goal of NLG is to produce outputs that are both accurate and diverse (Tevet and Berant, 2020). Knowledge graphs have recently gained popularity for enhancing NLG. Ji et al. (2020) and Yu et al. (2022) expanded their methods using a predefined vocabulary of concepts and relationships, selecting nodes and relationships based on frequency statistics and distance. Nonetheless, their approach tends to overlook important but low-frequency relationships and is limited in expansion depth. Liu et al. (2021) filtered entities using part-of-speech tags and cosine similarity of word embeddings, which helps capture semantic connections but relies heavily on pre-trained word embeddings, ignoring complex contextual relationships and being inefficient when handling large-scale data. Yasunaga et al. (2021) and Tang et al. (2023) built knowledge subgraphs using entity retrieval, but even with pre-trained models filtering nodes, noise may still be introduced.

However, their method overlooks the fact that introducing knowledge graphs can also bring irrelevant and redundant information, which increases noise and affects the performance of the model.

Previous methods often introduce an external graph $\mathcal{G}_x$ to constrain the generation process, such as $P(y_i|x, \mathcal{G}_x)$, where x is the given sentence and y_i is the generated sentence. However, these graphs frequently contain redundant or irrelevant triples that can negatively impact the diversity and quality of the generation. To enhance generative commonsense reasoning, our method introduces synonym expansion and subgraph pruning techniques. As shown in Figure 2, we incorporate synonym expansion from WordNet when extracting subgraph concepts (Section 3.1). Subsequently, we train a subgraph pruning model to trim the triples in the subgraph (Section 3.2), resulting in a more relevant subgraph. Then, a multi-relational graph encoder updates the representation of each node by iteratively aggregating information from neighboring nodes and edges. It generates output by integrating the token embeddings of the input sequence and the top-ranked concepts (Section 3.3).

Figure 1 shows an example of the commonsense explanation generation (ComVE) task. The goal is to generate an explanation for why the input sentence ("She ate an essay") violates commonsense. The concepts "term" and "digest" are semantically unrelated to the input or reference output sentences. Including this irrelevant information introduces noise, which may affect the performance of the model. To address the irrelevant information in the extracted subgraph, we introduce a synonym expansion method to expand the input concepts and build a task-relevant knowledge subgraph. Additionally, we propose a pruning model that learns to score the knowledge subgraph and remove parts not directly related to the input context.

The contributions of this paper can be summarized as follows:

- Building upon the MoE (Mixture of Experts) model, we introduced synonym expansion for enhancing input concepts, ultimately expanding them into a task-specific knowledge subgraph.

- We propose a pruning model and introduce a balanced pruning strategy. This model learns to score and prune the knowledge subgraph that is not directly relevant to the input context.

- We show through experiments that our method achieves comparable or better performance on various evaluation metrics for two GCR benchmark datasets, and matches the quality and diversity of the large language model Vicuna-13b.

2. Related Work

Knowledge graphs (KGs) that structure human knowledge have garnered significant interest in both academia and industry. They have been used in various applications such as dialogue systems (Tang et al., 2023), story generation (Ji et al., 2020), and textual entailment (Kapanipathi et al., 2020). Unlike the use of isolated, separate triples, using KGs can enrich the semantics of text generation.

2.1. Mixture of Experts (MoE) Module

One approach is to use MoE combined with knowledge graphs to generate diverse outputs. MoE is a machine learning technique that integrates multiple expert models, each specializing in generating specific types of outputs, thereby enhancing overall diversity in generation. This method has been validated in various studies, as shown by Shen et al. (2019) and Cho et al. (2019). By integrating with knowledge graphs, MoE can leverage rich semantic information from the graphs, enabling each expert model to effectively generate diverse outputs relevant to the task.

2.2. Optimizing Task-specific Knowledge Graph Information

However, not all relationships within a KG are equally relevant or useful for reasoning tasks. Some concept-entity connections might introduce noise or lack informational value, and mislead the understanding of the task by the model. Lin et al. (2019) used TransE (Wang et al., 2014) to score each edge in a path, while Yasunaga et al. (2021) scored the generation probability of nodes based on inputs from a pre-trained language model. Nonetheless, the scoring mechanisms in these methods were not specifically trained to reflect the importance of nodes for a particular task.

We focus on two datasets for commonsense explanation tasks: ComVE (Wang et al., 2020) and α-NLG (Bhagavatula et al., 2019). Our goal is to produce outputs that are both diverse and high-quality. Hwang et al. (2023) built on MoKGE (Yu et al., 2022) by introducing a differentiable graph compression algorithm. This algorithm focuses on more significant and relevant task knowledge. Our approach extends MoKGE, a model for generating commonsense explanations. MoKGE diversifies the outputs of a MoE model by combining knowledge from KGs. However, the knowledge retrieved from KGs by MoKGE is unfiltered and may include loosely related, redundant, and irrelevant information, which negatively impacts the model's ability to generate high-quality, diverse outputs.

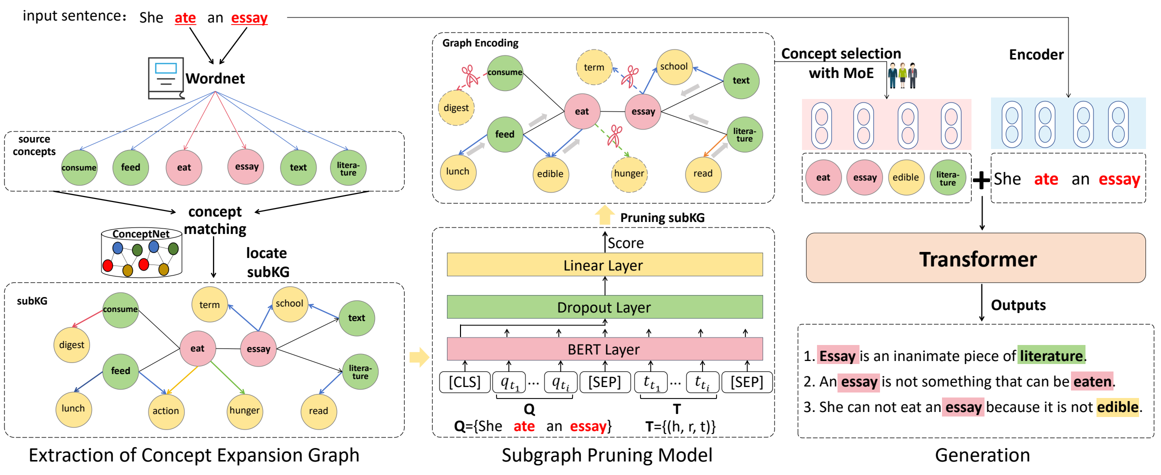


Figure 2: Overview of the proposed system.

3. Method

Problem Statement. Our goal is to generate diverse rationales for a given instance x in a generative commonsense reasoning task. The generated sentences $y_1, y_2, \ldots, y_k$ should be both correct (or valid) and natural for the given context. To facilitate the reasoning process, we utilize an external commonsense knowledge graph $G = \{(h, r, t) \mid h, t \in E, r \in R\}$, where E and R represent the sets of entities and relations, respectively. This graph can be viewed as auxiliary information. Within the instance x, there are k commonsense concepts $x = \{c_1, c_2, \ldots, c_k\}$, where each concept $c_i \in E$ is an object (noun) or an action (verb). We aim to maximize the generation probability $P(y_i|x)$ for each sentence y_i, while ensuring diversity in content, language style, and vocabulary.

3.1. Extraction of Concept Expansion Graph

To enrich the semantic representation of concepts and expand their semantic range, we use knowledge bases like WordNet to obtain synonyms for each extracted concept during subgraph extraction. This abundant synonym information enhances the ability of the model to understand different expressions. For each concept word, we traverse its set of synonyms and calculate the similarity of each synonym to the sentence. We use pre-trained word vector models like GloVe to represent words. We measure the semantic similarity between a given sentence x and a concept v using cosine similarity between their word vectors. Higher cosine similarity indicates stronger semantic relevance. For example, given the sentence $q =$ "She ate an essay", the key concepts in the sentence are $v_q =$ [eat, essay]. We select the top two words with the highest similarity for each concept as synonyms. The synonyms for "eat" are [feed, consume], and the synonyms for "essay" are [text, literature]. The integrated concept set is $v_{q'} =$ [eat, essay, feed, consume, text, literature]. Finally, we extract the concepts $v_{q'}$ that are connected within two hops.

3.2. Subgraph Pruning

We aim to first identify a subgraph relevant to the given sentence and then prune the irrelevant triples from the identified subgraph. To accomplish the pruning, we use a model to estimate the relevance of a sentence to its neighboring triples. As shown in Figure 2, the subgraph pruning model consists of a BERT layer, a dropout layer, and a linear layer. For each triple in the subgraph, the [CLS] token, representing the sentence as a set of tokens $\{q_{t_1}, \ldots, q_{t_i}\}$, and the triple as a set of tokens $\{t_{t_1}, \ldots, t_{t_i}\}$, are concatenated and fed into the BERT layer. The encoding of the [CLS] token, which represents the entire input, is passed through the dropout layer and then through the linear layer. We evaluate the relevance of a sentence to a triple each time. The output of the model is the pruning score for the given triple. After scoring all triples in the given subgraph, we adopt a balanced pruning strategy. We retain the top 70% and the bottom 20% of the triples to avoid over-pruning, reduce the loss of useful information, and ensure that low-priority but important content is not overlooked during the generation process.

The subgraph pruning model assigns high scores to relevant triples by fine-tuning a pre-trained BERT model. The fine-tuned BERT can compare the text representations of a given sentence and the triples (as shown in Figure 2) in the related subgraph. Given a sentence Q and a triple T, we define the following scoring function. *CosSim* computes the cosine similarity between the BERT embeddings of the sentence (represented as $ENC(Q)$) and the triple (represented as $ENC(T)$). This measures the similarity between the sentence and the triple. We score the triples from both relevance and similarity perspectives, with λ being a hyperparameter that balances the contribution of similarity and relevance in scoring the triples.

$$Score = \lambda \cdot CosSim(ENC(Q), ENC(T)) + (1 - \lambda) \cdot \rho(Q, T) \tag{1}$$

Where ρ is the Spearman's rank correlation coefficient, which measures the strength and direction of the monotonic relationship between two variables. Its range is $[-1, 1]$. The closer the absolute value of the correlation coefficient is to 1, the stronger the monotonic relationship between the two variables. Conversely, the closer it is to 0, the weaker the monotonic relationship. This method effectively evaluates the monotonic relationship between high-dimensional vectors. The calculation formula for Spearman's rank correlation coefficient is as follows:

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{2}$$

Given the sentence vector $Q = [Q_1, \ldots, Q_n]$ and the triple vector $T = [T_1, \ldots, T_n]$, where $n = 768$ dimensions, we rank the elements of vector Q and assign ranks $R(Q_i)$, and rank the elements of vector T and assign ranks $R(T_i)$. The rank difference is then calculated as:

$$d_i = R(Q_i) - R(T_i) \tag{3}$$

We then train the subgraph pruning model as a regression model. The loss function we use is the Mean Squared Error (MSE) loss function, which is a standard regression loss function:

$$\mathcal{L}_{\text{pruning}} = \frac{1}{n}\sum_{i=1}^{n}(\text{score}_i - \hat{\text{score}}_i)^2 \tag{4}$$

3.3. Graph Encoding, Training, and Generation

We follow the methodology outlined in previous work (Yu et al., 2022) for graph encoding, training, and generation.

Graph Encoding. Graph encoding (as shown in Figure 2) is performed by applying graph encoders, such as the method proposed by Wu et al. (2020), to the extracted subgraphs. We use the Relational Graph Convolutional Network R-GCN (Schlichtkrull et al., 2018) as our tool. This approach follows the research strategy of Yu et al. (2022). R-GCN generates concept node embeddings by iteratively aggregating the representations of neighboring nodes and considering the types of relations, capturing the structural patterns of the subgraph.

Training Loss Functions. For the commonsense explanation generation task, we use a sequence-to-sequence (seq2seq) architecture based on the BART-base model (Lewis et al., 2019) for training. The loss functions include a generation loss and an additional knowledge graph (KG) concept loss.

Generation Loss. In the sentence generation process, we aim to maximize the probability of the target sequence y given the input sequence x and the selected KG concepts $v_1, v_2, \ldots, v_N$. We use the standard autoregressive cross-entropy loss to measure generation accuracy, as shown below:

$$\mathcal{L}_{\text{generation}} = -\sum_{t=1}^{|y|} \log p(y_t | x, v_1, \ldots, v_N, y_{<t}) \tag{5}$$

KG Concept Loss. The effectiveness of concept selection is measured by the presence of selected concepts in the output sentence y (reference answer). For each concept $c \in V_x$ in the input sequence, we use a binary cross-entropy loss function $y_c = I(c \in V_x \cap C_y)$. $I(\cdot)$ is the indicator function, and C_y is the set of concepts present in the answer. We use a Multi-Layer Perceptron (MLP) to compute the probability p_c of each concept being selected. The resulting KG concept loss is as follows:

$$\mathcal{L}_{\text{concept}} = -\left(\sum_{c \in V_x \cap C_y} v \log p_c + \sum_{c \in V_x - C_y} (1-v)\log(1-p_c)\right) \tag{6}$$

Overall, our model is trained using the combined loss functions:

$$\mathcal{L} = \mathcal{L}_{\text{generation}} + \lambda \cdot \mathcal{L}_{\text{concept}} \tag{7}$$

where $\lambda = 0.3$ is a hyperparameter used to balance the two loss functions.

MoE-based Diverse Generation. To promote diversity in the output of the generative model, we use a MoE module trained with the hard Expectation-Maximization (EM) algorithm. Each expert is responsible for generating a set of unique KG concepts, prioritizing different key concepts during the generation process. We chose the EM algorithm to train the MoE mainly because it effectively distinguishes the task responsibilities of different expert models, enhancing the diversity of the generated outputs. The EM algorithm alternates between optimizing which experts to use and updating their parameters, allowing each expert to focus on different generation patterns. This helps reduce conflicts and avoid mode collapse. In contrast, training all expert models at once may be faster, but it can lead to overfitting, which reduces the diversity of the generated results.

The algorithm aims to achieve diversified generation using a MoE model. Initially, the algorithm initializes model parameters θ and responsibilities r_z for each expert. Subsequently, training is conducted by alternately running the hard Expectation-Maximization (EM) algorithm. During the E-step (lines 4 to 8), for each input-output pair (x, y), it computes the responsibility r_z for each expert z under the current model, utilizing the knowledge graph G_x. In the M-step (lines 10 to 11), it updates the parameters θ, considering only those designated as responsible experts. This process iterates until convergence. Finally, outputs y_z are generated for each expert (lines 13 to 20), returned as K diversified generation results $y_1, y_2, \ldots, y_K$.

```
Algorithm: MoE for Diverse Generation
Input: input $x$, knowledge graph $G_x$, number of experts $K$
Output: diverse outputs $y_1, y_2, \ldots, y_K$
1   Initialize model parameters $\theta$.
2   Initialize experts' responsibilities $r_z$ for each $z \in \{1, \ldots, K\}$.
3   while not converged do
        Training using Hard EM:
4           E-step: Estimate responsibilities
5               for each example $(x, y)$ do
6                   Calculate $p(z|x, y, G_x)$ for $z = 1$ to $K$ using current $\theta$.
7                   if $z$ maximizes $p(z|x, y, G_x)$ then
                        Set $r_z = \mathbb{1}$.
8                   else
                        Set $r_z = 0$.
9               end for
10          M-step: Update parameters
11              Update $\theta$ using examples where $r_z = \mathbb{1}$ for expert $z$.
12      end while
13  Initialize empty list of outputs $y_1, y_2, \ldots, y_K$.
14  for $z = 1$ to $K$ do
15          Initialize $y_z$ as an empty sequence.
16          for each token in input sequence $x$ do
17              Greedily decode token $y_t$ using $p(y_t|y_{1:t-1}, z, x, G_x)$.
18              Append $y_t$ to $y_z$.
19          end for
20          Append $y_z$ to outputs $y_1, y_2, \ldots, y_K$.
21  end for
22  return $y_1, y_2, \ldots, y_K$ as $K$ diverse outputs.
```

4. Experiments

4.1. Datasets

We used two commonsense generation tasks (Zhang et al., 2024) to evaluate the performance of our generative model: ComVE and α-NLG. **ComVE** (Wang et al., 2020) is part of the SemEval 2020 commonsense validation task, which focuses on generating explanations for absurd sentences. The main goal of this dataset is to generate explanations for given absurd sentences, explaining why they are nonsensical. **α-NLG** (Bhagavatula et al., 2019) dataset is used for hypothetical commonsense reasoning tasks. The goal of this task is to generate explanations for reasonable events that could have happened between given past and future observations.

4.2. Baselines

In the experimental section of this paper, we selected various baseline methods, following the baseline methods from previous work (Hwang et al., 2023).

MoE-based Methods. First, we compared our approach with two methods based on MoE, which generate diverse outputs by sampling from different mixture components. Specifically, we used the MoE-prompt method proposed by Shen et al. (2019) and the MoE-embed method proposed by Cho et al. (2019). These methods provide an effective approach to generate diverse responses using multiple expert models.

Methods for Improving Diversity. To further demonstrate the advantages of our subgraph pruning method, we considered common pruning methods that remove irrelevant paths from potentially noisy subgraphs, such as KagNet (Lin et al., 2019). KagNet decomposes paths into a set of triples, scores each triple using the scoring function from the knowledge graph embedding technique TransE, and uses the product of the scores of each triple as the overall score of the path. We also compared our method with the MoKGE+SAG+OT method based on compressed graphs (Hwang et al., 2023).

Large Language Model (LLM). Finally, we compared our model with Vicuna-13b (Chiang et al., 2023), a representative of modern large-scale pre-trained language models that perform excellently in generation tasks. Vicuna-13b is a large language model built on LLaMA-13b (Touvron et al., 2023), which is based on the transformer architecture, and specifically trained to handle trillions of tokens from open datasets.

4.3. Implementation Details

For a fair comparison, our method and all baseline methods use BART-base to initialize the Transformer parameters, while R-GCN parameters are randomly initialized.

The model training used a batch size of 50, a learning rate of 3e-5, and L2 weight decay of 0.01. The learning rate was warmed up for the first 10,000 steps and then linearly decayed. Training was conducted on a single Nvidia 4090 GPU (24GB memory) using PyTorch and the Huggingface Transformers library (Wolf et al., 2020). All Transformer-based methods were trained for 30 epochs on the ComVE dataset, taking about 2 hours on the ComVE dataset and 2-4 hours on the α-NLG dataset.

4.4. Metrics

We follow evaluation setup as in previous work (Yu et al., 2022; Hwang et al., 2023). We evaluate the performance of the generative model based on diversity and quality using common evaluation metrics: Self-BLEU-3/4, Distinct-k, BLEU, ROUGE, and Entropy.

Quality Evaluation. BLEU score is an N-gram precision metric used to evaluate the similarity between generated sentences and reference sentences. It calculate the N-gram overlap between each generated sentence and its corresponding reference sentence. A higher BLEU score indicates a better match between generated and reference sentences. The ROUGE score is another N-gram-based metric that measures the coverage and recall of generated text by comparing the N-gram overlap between generated and reference sentences. A higher ROUGE score indicates that the generated sentences better cover the content of the reference sentences.

Diversity Evaluation. Self-BLEU (Zhu et al., 2018) evaluates the diversity within a set of generated sentences. Self-BLEU-3/4 measures the similarity between generated sentences based on 3-gram and 4-gram overlaps. It calculates the BLEU score for each generated sentence against all other generated sentences and averages them. Lower Self-BLEU scores indicate higher diversity, with less repetition and similarity among sentences. Distinct-k (Li et al., 2015) measures the number of unique k-grams in the generated sentences, normalizing by the total number of words to prevent bias toward longer sentences. Higher Distinct-k values indicate higher lexical diversity, with less repetition of common words. Entropy-k (Zhang et al., 2018) evaluates the diversity and uniformity of generated sentences by calculating the entropy of the k-gram distribution within the sentences. Higher Entropy-k values indicate greater uniformity and diversity.

5. Results and Analysis

Comparison with Baseline Methods (Table 1). Our method shows similar trends across two datasets and two model series (embedding-based and prompt-based). In terms of diversity, our approach (including synonym addition and subgraph pruning) significantly outperforms the Mixture of Experts methods, while also achieving comparable or better performance in quality. Compared to MoKGE+SAG+OT (Hwang et al., 2023), our method also demonstrates comparable or superior results in both diversity and quality.

Our method shows clear advantages when compared to filtering and pruning baselines. By adding synonyms and pruning subgraphs, our approach effectively suppresses redundant information in the knowledge graph and introduces more relevant knowledge in the given context, thereby enhancing overall performance.

We compared our method with Vicuna-13b, and the experimental results are presented in Table 2. Compared to MoKGE+SAG+OT (Hwang et al., 2023), our method also outperforms Vicuna-13b in terms of Distinct-2 and Entropy-4 metrics.

We focused on investigating the impact of using different types of concept expansion methods on model performance during training. Specifically, we explored three types of expansion methods: hypernyms, hyponyms, and synonyms. We evaluated the effectiveness of each method through multiple comparative tests.

The appropriate addition of synonyms can effectively enhance model performance, while adding too many or too few synonyms can have the opposite effect. Using synonyms for

Table 1: Evaluation of diversity and quality on the ComVE and α-NLG datasets. Each experiment is conducted three times with different random seeds, and the standard deviations are reported as subscripts. The symbol ‡ represents the results of previous studies. The best results are **bold**, and the second best results are $\underline{\text{underlined}}$.

ComVE	**self-bleu-3** ($\Downarrow$)	**self-bleu-4** ($\Downarrow$)	**distinct-2** ($\Uparrow$)	**entropy-4** ($\Uparrow$)	**bleu-4** ($\Uparrow$)	**rouge-l** ($\Uparrow$)
MoE, embed ‡	$33.64_{0.20}$	$28.21_{0.10}$	$46.57_{0.20}$	$9.61_{0.10}$	$18.66_{0.50}$	$\underline{43.72_{0.20}}$
MoKGE, embed ‡	$35.36_{1.10}$	$29.71_{1.20}$	$47.51_{0.40}$	$9.63_{0.10}$	$\underline{19.13_{0.10}}$	$43.70_{0.10}$
+ SAG + OT ‡	$\mathbf{32.19_{0.60}}$	$\mathbf{26.28_{0.60}}$	$\mathbf{49.05_{0.10}}$	$\mathbf{9.69_{0.00}}$	$19.08_{0.20}$	$43.65_{0.30}$
+ pruning(ours)	$33.29_{0.91}$	$27.20_{0.94}$	$\underline{48.32_{0.32}}$	$\underline{9.68_{0.00}}$	$18.92_{0.57}$	$43.78_{0.44}$
+ 2 synonyms(ours)	$33.08_{0.34}$	$27.57_{0.38}$	$47.62_{0.02}$	$9.63_{0.01}$	$\mathbf{19.21_{0.40}}$	$43.46_{0.01}$
+ 2 synonyms+pruning(ours)	$\underline{33.04_{0.56}}$	$\underline{27.15_{0.69}}$	$48.30_{0.36}$	$9.66_{0.00}$	$18.94_{0.15}$	$\mathbf{43.93_{0.13}}$
MoE, prompt ‡	$33.42_{0.30}$	$28.40_{0.30}$	$46.93_{0.20}$	$9.60_{0.20}$	$18.91_{0.40}$	$43.71_{0.50}$
MoKGE, prompt ‡	$30.93_{0.90}$	$25.30_{1.10}$	$\underline{48.44_{0.02}}$	$9.67_{0.20}$	$19.13_{0.10}$	$43.83_{0.30}$
+ filtering ‡	$34.01_{0.50}$	$28.92_{0.50}$	$47.49_{0.90}$	$9.64_{0.10}$	$19.02_{0.40}$	$43.48_{0.60}$
+ pruning ‡	$33.43_{2.00}$	$28.27_{2.20}$	$48.26_{0.70}$	$9.64_{0.00}$	$18.67_{0.20}$	$43.10_{0.30}$
+ SAG + OT ‡	$\mathbf{27.32_{0.30}}$	$\underline{21.94_{0.40}}$	$\mathbf{48.94_{0.10}}$	$\mathbf{9.69_{0.00}}$	$19.31_{0.30}$	$44.16_{0.10}$
+ pruning(ours)	$29.54_{0.14}$	$24.50_{0.20}$	$47.93_{0.04}$	$9.66_{0.00}$	$\mathbf{20.20_{0.02}}$	$\underline{44.39_{0.16}}$
+ 2 synonyms(ours)	$27.93_{0.37}$	$22.14_{0.43}$	$48.42_{0.08}$	$\underline{9.69_{0.05}}$	$19.64_{0.15}$	$44.36_{0.02}$
+ 2 synonyms+pruning(ours)	$\underline{27.43_{0.20}}$	$\mathbf{21.82_{0.20}}$	$48.33_{0.04}$	$9.66_{0.00}$	$\underline{19.55_{0.39}}$	$\mathbf{44.41_{0.25}}$
α-NLG	**self-bleu-3** ($\Downarrow$)	**self-bleu-4** ($\Downarrow$)	**distinct-2** ($\Uparrow$)	**entropy-4** ($\Uparrow$)	**bleu-4** ($\Uparrow$)	**rouge-l** ($\Uparrow$)
MoE, embed ‡	$29.02_{1.00}$	$24.19_{1.00}$	$36.22_{0.30}$	$10.84_{0.00}$	$\mathbf{14.31_{0.20}}$	$\mathbf{38.91_{0.20}}$
MoKGE, embed ‡	$29.17_{1.50}$	$24.04_{1.60}$	$38.15_{0.30}$	$10.90_{0.10}$	$\underline{13.74_{0.20}}$	$38.06_{0.20}$
+ SAG + OT ‡	$\mathbf{24.98_{0.20}}$	$\mathbf{19.83_{0.20}}$	$\mathbf{38.93_{0.30}}$	$\mathbf{10.93_{0.00}}$	$13.06_{0.30}$	$37.77_{0.30}$
+ pruning(ours)	$27.09_{0.43}$	$\underline{21.93_{0.39}}$	$39.06_{0.14}$	$\mathbf{10.93_{0.00}}$	$13.47_{0.06}$	$\underline{38.33_{0.12}}$
+ 2 synonyms(ours)	$28.38_{0.12}$	$23.13_{0.11}$	$38.5_{0.05}$	$10.9_{0.0}$	$12.74_{0.06}$	$37.83_{0.05}$
+ 2 synonyms+pruning(ours)	$\underline{27.67_{0.07}}$	$22.45_{0.10}$	$\underline{38.66_{0.05}}$	$\underline{10.91_{0.00}}$	$13.27_{0.03}$	$37.96_{0.05}$
MoE, prompt ‡	$28.05_{2.00}$	$23.18_{1.90}$	$36.71_{0.10}$	$10.85_{0.00}$	$\mathbf{14.26_{0.30}}$	$38.78_{0.40}$
MoKGE, prompt ‡	$27.40_{2.00}$	$22.43_{2.40}$	$38.01_{0.60}$	$10.88_{0.20}$	$14.17_{0.20}$	$38.82_{0.70}$
+ filtering ‡	$31.38_{2.90}$	$26.36_{2.80}$	$37.95_{0.60}$	$10.78_{0.60}$	$13.89_{0.20}$	$38.07_{0.10}$
+ pruning ‡	$31.84_{2.30}$	$26.72_{2.40}$	$38.21_{0.20}$	$10.78_{0.00}$	$13.73_{0.10}$	$38.01_{0.20}$
+ SAG + OT ‡	$\underline{23.99_{0.70}}$	$\underline{18.80_{0.60}}$	$39.02_{0.70}$	$\underline{10.88_{0.00}}$	$\underline{14.21_{0.50}}$	$\underline{38.93_{0.20}}$
+ pruning(ours)	$\mathbf{23.70_{0.23}}$	$\mathbf{18.44_{0.22}}$	$\mathbf{39.60_{0.02}}$	$\mathbf{10.92_{0.02}}$	$13.91_{0.00}$	$\mathbf{39.08_{0.04}}$
+ 2 synonyms(ours)	$24.69_{0.01}$	$19.43_{0.01}$	$\underline{39.43_{0.02}}$	$\mathbf{10.92_{0.00}}$	$13.90_{0.01}$	$38.88_{0.03}$
+ 2 synonyms+pruning(ours)	$30.21_{0.65}$	$24.97_{0.68}$	$38.68_{0.16}$	$10.83_{0.01}$	$13.73_{0.18}$	$38.32_{0.24}$

Table 2: Following the settings of prior work (Hwang et al., 2023), Vicuna-13b is compared with our method under the prompts of MoKGE.

ComVE	**self-bleu-3** ($\Downarrow$)	**self-bleu-4** ($\Downarrow$)	**distinct-2** ($\Uparrow$)	**entropy-4** ($\Uparrow$)	**bleu-4** ($\Uparrow$)	**rouge-l** ($\Uparrow$)
Vicuna-13b ‡	$\mathbf{18.10_{0.00}}$	$\mathbf{12.74_{0.00}}$	$48.40_{0.00}$	$9.65_{0.00}$	$17.65_{0.00}$	$43.97_{0.00}$
MoKGE + SAG + OT ‡	$\underline{27.32_{0.30}}$	$21.94_{0.40}$	$\mathbf{48.94_{0.10}}$	$\mathbf{9.69_{0.00}}$	$19.31_{0.30}$	$44.16_{0.10}$
MoKGE + pruning (ours)	$29.54_{0.14}$	$24.50_{0.20}$	$47.93_{0.04}$	$9.66_{0.00}$	$\mathbf{20.20_{0.02}}$	$\underline{44.39_{0.16}}$
MoKGE + 2 synonyms (ours)	$27.93_{0.37}$	$22.14_{0.43}$	$\underline{48.42_{0.08}}$	$\mathbf{9.69_{0.05}}$	$19.64_{0.15}$	$44.36_{0.02}$
MoKGE + 2 synonyms + pruning (ours)	$27.43_{0.20}$	$\underline{21.82_{0.20}}$	$48.33_{0.04}$	$9.66_{0.00}$	$\underline{19.55_{0.39}}$	$\mathbf{44.41_{0.25}}$
α-NLG	**self-bleu-3** ($\Downarrow$)	**self-bleu-4** ($\Downarrow$)	**distinct-2** ($\Uparrow$)	**entropy-4** ($\Uparrow$)	**bleu-4** ($\Uparrow$)	**rouge-l** ($\Uparrow$)
Vicuna-13b ‡	$33.23_{0.00}$	$27.39_{0.00}$	$37.97_{0.00}$	$10.38_{0.00}$	$\mathbf{17.30_{0.00}}$	$\mathbf{40.58_{0.00}}$
MoKGE + SAG + OT ‡	$\underline{23.99_{0.70}}$	$\underline{18.80_{0.60}}$	$39.02_{0.70}$	$10.88_{0.00}$	$\underline{14.21_{0.50}}$	$38.93_{0.20}$
MoKGE + pruning (ours)	$\mathbf{23.70_{0.23}}$	$\mathbf{18.44_{0.22}}$	$\mathbf{39.60_{0.02}}$	$\mathbf{10.92_{0.02}}$	$13.91_{0.00}$	$\underline{39.08_{0.04}}$
MoKGE + 2 synonyms (ours)	$24.69_{0.01}$	$19.43_{0.01}$	$\underline{39.43_{0.02}}$	$\mathbf{10.92_{0.00}}$	$13.90_{0.01}$	$38.88_{0.03}$
MoKGE + 2 synonyms + pruning (ours)	$30.21_{0.65}$	$24.97_{0.68}$	$38.68_{0.16}$	$10.83_{0.01}$	$13.73_{0.18}$	$38.32_{0.24}$

concept expansion is more effective than hypernyms and hyponyms. In particular, adding around two synonyms yields the best improvement in model performance.

Table 3: Ablation Study. Concept expansion was performed separately using two hypernyms, hyponyms, and synonyms.

	ComVE						α-NLG					
	SB-3 (⇓)	**SB-4 (⇓)**	**D-2 (⇑)**	**E-4 (⇑)**	**B-4 (⇑)**	**R-L (⇑)**	**SB-3 (⇓)**	**SB-4 (⇓)**	**D-2 (⇑)**	**E-4 (⇑)**	**B-4 (⇑)**	**R-L (⇑)**
MoKGE, prompt ‡	30.93$_{0.90}$	25.30$_{1.10}$	**48.44$_{0.20}$**	9.67$_{0.20}$	19.13$_{0.10}$	43.83$_{0.30}$	27.40$_{2.00}$	22.43$_{2.40}$	38.01$_{0.60}$	10.88$_{0.20}$	14.17$_{0.20}$	38.82$_{0.70}$
+ 2 hypernyms	34.45$_{0.39}$	29.50$_{0.36}$	48.06$_{0.22}$	9.60$_{0.0}$	19.00$_{0.03}$	43.52$_{0.0}$	31.72$_{0.19}$	26.52$_{0.16}$	38.75$_{0.09}$	10.74$_{0.00}$	**15.03$_{0.00}$**	**38.90$_{0.03}$**
+ 2 hyponyms	35.41$_{0.31}$	30.04$_{0.32}$	48.03$_{0.10}$	9.65$_{0.0}$	18.63$_{0.15}$	43.35$_{0.04}$	31.48$_{0.32}$	26.25$_{0.31}$	39.20$_{0.34}$	10.78$_{0.03}$	14.43$_{0.52}$	38.71$_{0.19}$
+ 2 synonyms	**27.93$_{0.37}$**	**22.14$_{0.43}$**	48.42$_{0.08}$	**9.69$_{0.05}$**	**19.64$_{0.15}$**	**44.36$_{0.02}$**	**24.69$_{0.01}$**	**19.43$_{0.01}$**	**39.43$_{0.02}$**	**10.92$_{0.00}$**	13.90$_{0.01}$	38.88$_{0.03}$

Impact of Synonyms. Experimental results (Table 3) indicate that using synonyms for concept expansion significantly improves model performance. By adding synonyms, the model can acquire more semantic information about sentences, resulting in more accurate understanding and processing of natural language. This improvement is likely due to the high semantic relatedness of synonyms, which provides more information about semantic similarity, enabling the model to handle different expressions of the same concept more flexibly and robustly.

Impact of Hypernyms and Hyponyms. In comparison to synonyms, the use of hypernyms and hyponyms for concept expansion did not meet expectations. The experimental results (Table 3 and 4) show that these methods introduce more noise, leading to a decrease in model performance. This may be because hypernyms generally represent more abstract concepts, while hyponyms represent more specific concepts. The semantic distance between hypernyms and hyponyms from the original concept increases the complexity of understanding and processing, interfering with the judgment of the model and resulting in decreased performance.

Table 4: Ablation Study. Concept expansion was performed using 1 to 3 hypernyms, hyponyms, and synonyms.

ComVE	**self-bleu-3 (⇓)**	**self-bleu-4 (⇓)**	**distinct-2 (⇑)**	**entropy-4 (⇑)**	**bleu-4 (⇑)**	**rouge-l (⇑)**
MoKGE, prompt ‡	30.93$_{0.90}$	$\underline{25.30}_{1.10}$	$\underline{48.44}_{0.20}$	$\underline{9.67}_{0.20}$	$\underline{19.13}_{0.10}$	$\underline{43.83}_{0.30}$
+ 1 hypernyms	35.71$_{0.23}$	30.85$_{0.34}$	47.50$_{0.02}$	9.65$_{0.00}$	17.02$_{0.11}$	41.32$_{0.02}$
+ 2 hypernyms	34.45$_{0.39}$	29.50$_{0.36}$	48.06$_{0.22}$	9.60$_{0.00}$	19.00$_{0.03}$	43.52$_{0.00}$
+ 3 hypernyms	$\underline{30.71}_{0.16}$	25.67$_{0.23}$	**48.51$_{0.13}$**	9.64$_{0.00}$	18.32$_{0.06}$	43.00$_{0.09}$
+ 1 hyponyms	35.00$_{0.12}$	29.88$_{0.10}$	47.29$_{0.02}$	9.62$_{0.00}$	18.54$_{0.05}$	43.23$_{0.10}$
+ 2 hyponyms	35.41$_{0.31}$	30.04$_{0.32}$	48.03$_{0.10}$	9.65$_{0.00}$	18.63$_{0.15}$	43.35$_{0.04}$
+ 3 hyponyms	34.10$_{0.53}$	29.32$_{0.60}$	47.05$_{0.18}$	9.61$_{0.01}$	18.07$_{0.08}$	42.72$_{0.09}$
+ 1 synonyms	32.84$_{0.05}$	27.64$_{0.02}$	48.24$_{0.05}$	9.61$_{0.00}$	18.20$_{0.14}$	42.83$_{0.14}$
+ 2 synonyms	**27.93$_{0.37}$**	**22.14$_{0.43}$**	48.42$_{0.08}$	**9.69$_{0.05}$**	**19.64$_{0.15}$**	**44.36$_{0.02}$**
+ 3 synonyms	37.13$_{0.24}$	31.75$_{0.18}$	46.59$_{0.27}$	9.63$_{0.00}$	18.75$_{0.11}$	43.71$_{0.08}$

Impact of the Number of Synonyms. In further experiments (Table 4), we compared the effects of adding different numbers of synonyms on model performance. Specifically, we tested the addition of 1, 2, and 3 synonyms. The results show that the addition of 2 synonyms yields the best performance, while the addition of 1 or 3 synonyms results in decreased performance.

This phenomenon can be explained for the following reasons: When using synonyms for concept expansion, the model treats synonyms as source concepts and searches for neighboring concepts in the knowledge graph. Adding 1 synonym may provide insufficient information to significantly enhance the semantic understanding of the model. Adding 3 synonyms, on the other hand, may introduce too much noise, reducing the number of neighboring concepts found by the model and potentially encountering more interfering information during the expansion process. This reduces the coverage of knowledge by the model. Therefore, an optimal number of synonyms can effectively improve model performance, while too many or too few synonyms can have the opposite effect.

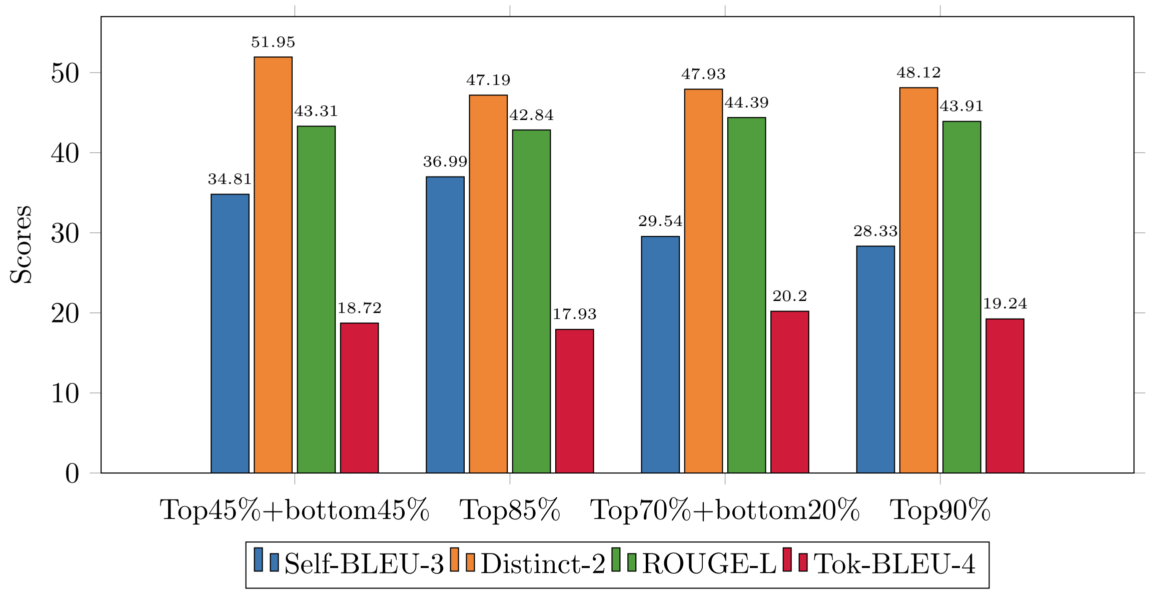


Figure 3: The scores of Self-BLEU-3, Distinct-2, ROUGE-L, and Tok-BLEU-4 on the ComVE dataset. The experiment uses MoKGE-prompt after pruning triples with the pruning model.

Pruning Strategy. As shown in Figure 3, we experimented with retaining the top 90% highest-scoring triples, the top 85% highest-scoring triples, both the top 70% highest-scoring triples and the bottom 20% lowest-scoring triples, and retaining both the top 45% highest-scoring triples and the bottom 45% lowest-scoring triples.

During the subgraph pruning process, selecting to retain the top k highest-scoring triples and pruning the rest does not always yield optimal performance. Compared to only retaining the top k triples, the strategy of retaining the top 70% of triples avoids over-pruning, thereby reducing the loss of valuable information. Retaining the bottom 20% of triples provides balance, ensuring that the generation process does not overlook lower-priority but still important content.

6. Conclusions and Future Work

This paper introduces a method to expand input concepts using synonym expansion and constructing a task-specific knowledge subgraph. We also develop a pruning model to assess and remove knowledge subgraph parts irrelevant to the input context. These methods significantly enhance the quality and diversity of generated text, offering a new solution for generative commonsense reasoning tasks. They demonstrate that effectively leveraging knowledge graphs to optimize the selection of nodes and their relations can notably improve task performance.

Future work can further optimize the expansion and pruning model algorithms, explore their applications in other natural language generation tasks, and investigate the dynamic adjustments of knowledge subgraphs to adapt to different contexts and task requirements.

References

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*, 2019.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, march 2023. *URL https://lmsys. org/blog/2023-03-30-vicuna*, 3(5), 2023.

Jaemin Cho, Minjoon Seo, and Hannaneh Hajishirzi. Mixture content selection for diverse sequence generation. *arXiv preprint arXiv:1909.01953*, 2019.

EunJeong Hwang, Veronika Thost, Vered Shwartz, and Tengfei Ma. Knowledge graph compression enhances diverse commonsense generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 558–572, 2023.

Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, Xiaoyan Zhu, and Minlie Huang. Language generation with multi-hop reasoning on commonsense knowledge graph. *arXiv preprint arXiv:2009.11692*, 2020.

Pavan Kapanipathi, Veronika Thost, Siva Sankalp Patel, Spencer Whitehead, Ibrahim Abdelaziz, Avinash Balakrishnan, Maria Chang, Kshitij Fadnis, Chulaka Gunasekara, Bassem Makni, et al. Infusing knowledge into the textual entailment task using graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8074–8081, 2020.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*, 2019.

Ye Liu, Yao Wan, Lifang He, Hao Peng, and S Yu Philip. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6418–6425, 2021.

Weizhi Nie, Yuru Bao, Yue Zhao, and Anan Liu. Long dialogue emotion detection based on commonsense knowledge graph guidance. *IEEE Transactions on Multimedia*, 26:514–528, 2023.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer, 2018.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. Mixture models for diverse machine translation: Tricks of the trade. In *International conference on machine learning*, pages 5719–5728. PMLR, 2019.

Chen Tang, Hongbo Zhang, Tyler Loakman, Chenghua Lin, and Frank Guerin. Enhancing dialogue generation via dynamic graph knowledge aggregation. *arXiv preprint arXiv:2306.16195*, 2023.

Guy Tevet and Jonathan Berant. Evaluating the evaluation of diversity in natural language generation. *arXiv preprint arXiv:2004.02990*, 2020.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. Semeval-2020 task 4: Commonsense validation and explanation. *arXiv preprint arXiv:2007.00236*, 2020.

Ruijie Wang, Luca Rossetto, Michael Cochez, and Abraham Bernstein. Qagcn: A graph convolutional network-based multi-relation question answering system. *arXiv preprint arXiv:2206.01818*, 2022.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214, 2024.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*, 2021.

Wenhao Yu, Chenguang Zhu, Lianhui Qin, Zhihan Zhang, Tong Zhao, and Meng Jiang. Diversifying content generation for commonsense reasoning with mixture of knowledge graph experts. *arXiv preprint arXiv:2203.07285*, 2022.

Tianhui Zhang, Bei Peng, and Danushka Bollegala. Improving diversity of commonsense generation by large language models via in-context learning. *CoRR*, abs/2404.16807, 2024.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. Generating informative and diverse conversational responses via adversarial information maximization. *Advances in Neural Information Processing Systems*, 31, 2018.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 1097–1100, 2018.