Data-Efficient Concept Extraction from Pre-trained Language Models for Commonsense Explanation Generation

Yanbo Fang  
Department of Computer Science  
Rutgers University  
New Brunswick, NJ, US  
yanbo.fang@rutgers.edu

Yongfeng Zhang  
Department of Computer Science  
Rutgers University  
New Brunswick, NJ, US  
yongfeng.zhang@rutgers.edu

Abstract

Predicting the key explanation concept is essential for generating commonsense explanations. This paper introduces a method to predict the concept from pre-trained language models for commonsense explanation generation. Our experiment found that adopting a language model as the concept extractor and fine-tuning it with 20% training data can improve the quality and accuracy of the generated explanations over multiple evaluation metrics. Compared with conventional methods that search concepts over knowledge graphs, our method does not require the preparation and training models to search through knowledge graphs. To better understand the results from pre-trained language models, we also designed a metric to evaluate the retrieved concepts. Through analysis and experiments, we show the correlation between this metric and the performance of the generators, and we also show the importance of attaching concepts for generating high-quality sentences.

1 Introduction

The natural language processing (NLP) field has gigantic advancements since the involvement of deep learning (Young et al., 2018) and large-size pre-training models (Liu et al., 2019; Devlin et al., 2019; Radford and Narasimhan, 2018). However, creating human-like AI systems requires models comprehending commonsense knowledge and reasoning, which is still a challenge for current NLP models (Jia and Liang, 2017). Among all the research about making language models’ responses fit common sense knowledge, we would like to focus on generating sentences to explain a statement with commonsense knowledge (Wang et al., 2019). An example of this task is shown in Table 1. Given a statement, the goal is to output an explanation to explain the statement.

There are two ways for this task at present. One way, Figure 1(a), is to let language models to generate an explanation directly given one statement without external assistance (Wang et al., 2019, 2020). The benefit is that this is an end-to-end pipeline and would be easy to use, but it would be hard to interpret why models output such sentences, and most times, the performance would be limited. Another method, Figure 1(b), is extracting bridge concepts from external structured knowledge graphs to connect statements and explanations (Ji et al., 2020). With the assistance of knowledge graphs, such as ConceptNet (Liu and Singh, 2004), this method can introduce ground truth knowledge to offer helpful information for generation, and the selected concepts can be used to interpret models’ outputs. Our experiments in Figure 3 confirm that informative concepts can assist the generation process. However, this method would need additional training to search concepts, and this method would be limited under a constrained environment where structured data would be rare.

This paper introduces a pre-trained language model driven method, Figure 1(c), to combine advantage of both methods. The key is that we can adopt

Statement: The school was open for summer

Template: Statement that the school was open for summer is wrong because concepts about [MASK].

Concepts: summer, school, education, the, schools, time, it, vacation, holidays, students, learning, kindergarten, that, teaching, children

Explanation: Summer time is typically vacation time for school.

Table 1: Example of the task and our method. The task is given a statement to produce an explanation. Our method adopts masked language modeling to firstly retrieve concepts from pre-trained language models by transferring statements into templates before generation.
Figure 1: This Figure gives illustrations of (a) standard method for generation without concepts augmentation, (b) method enhanced by concepts from knowledge graphs and trained models, and (c) method enhanced by concepts from pre-trained language models without training for this explanation generation task. KG is a knowledge graph. G is a generator model. PLM means a pre-trained language model.

Our approach is composed of two distinct models, conceptor and generator. The conceptor is used to provide bridge concepts for a given statement. The generator is responsible for generating explanations based on the combining statement and concepts. Following prompt-based tuning (Gao et al., 2021; Schick and Schütze, 2021), we design a template to transfer a generation question to a masked prediction task. Then we use data efficient finetuning to modify pre-trained language models such as BERT or RoBERTa to encode statements, and we pick up candidates with the highest scores in vocabulary as concepts by masked language modeling. An example is in Table 1. In the next step, the selected concepts are attached at the end of the statements as the generator’s input to generate explanations. This method arises with two motivations. One is that pre-trained language models could implicitly encode world knowledge (Niven and Kao, 2019; Talmor et al., 2020; Davison et al., 2019). Another is that while pre-trained models find it challenging to produce sentences that call for world knowledge, it would be simpler for them to output key concepts without taking syntax and grammar into account (Petroni et al., 2019).

We discovered that concepts from pre-trained language models enhance the metric scores of generated sentences compared to baseline models. Compared with structured knowledge graphs, pre-trained language models are widely developed and convenient for processing unstructured texts. Moreover, fine-tuning the conceptor with more data points would further improve the efficiency of concept retrievals and metric scores. Furthermore, we design experiments to identify the advantages of obtaining informative concepts from pre-trained language models for explanation generations.

Our contributions can be summarized as follows:

- We proposed to replace annotated knowledge graphs with pre-trained language models for concept extraction. Our data-efficient training method demonstrates its advantages in the explanation generation tasks.
- We designed a metric to assess the retrieved concepts and showed the association between the metric scores and created sentence quality.
- We analyzed the challenges of retrieving concepts from current pre-trained language models and how these concepts would help generate better explanations.

2 Related Work

Knowledge of Pre-trained Language Model. Pre-trained language models are now the mainstream of
the Natural Language Processing community and demonstrate incredible strength in solving complex tasks, such as text understanding, generation, and domain generalization (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020; Hendrycks et al., 2020). However, these models make inferences in a human agnostic way and are easy to be attacked by attaching a trigger (Wallace et al., 2019). Therefore, some researchers are trying to interpret what these models have learned in their pre-training. For example, (Hewitt and Manning, 2019) shows it can recover syntactic dependency from token representations from BERT, and pre-trained language models contain a sense of world knowledge (Petroni et al., 2019; Li et al., 2021a).

Moreover, to reduce the gap between pre-training and fine-tuning, the prompt-based method is proved an efficient way to improve pre-trained language models’ performance on downstream tasks (Gao et al., 2021; Schick and Schütze, 2021; Le Scao and Rush, 2021). Thus, we adopt prompt-tuning and designed templates to express pre-trained language models’ internal knowledge to replace the external knowledge graphs.

**Reasonable Explanation Sentence Generation.** Some of the current works (Ji et al., 2020; Lin et al., 2020) focus on adjusting models to output sentences that fit the context and commonsense knowledge. Because investigations (Zhou et al., 2020; Richardson and Sabharwal, 2020) find that current pre-trained models have disadvantages in solving tasks needing inferences steps, some of these research (Moon et al., 2019; Zhou et al., 2018; Guan et al., 2019) propose using a knowledge graph to provide additional concepts to ensure controllable generation. Natural language explanation generation is also extensively explored in explainable recommender system research (Zhang et al., 2020), which aims to generate personalized and reasonable natural language sentences to explain why certain items are recommended to users. Early methods used manual templates to generate explanation sentences (Zhang et al., 2014), and more recent works explored neural-template generation methods (Li et al., 2020a,b), personalized transformers (Li et al., 2021c, 2022b), visual-enhanced explanation generation (Geng et al., 2022a; Chen et al., 2022a,b; Zhu et al., 2021), path-based explanations (Geng et al., 2022b; Xian et al., 2019, 2020), as well as large language models for explanation generation such as the P5 large recommendation model (Geng et al., 2022c). Following previous work, we use pre-trained language models as a knowledge base to replace annotated knowledge graphs so as to provide concepts for commonsense explanation sentence generation.

### 3 Method

#### 3.1 Problem Setup

In this task, conditioning on an statement $x = \{x_1, x_2, \ldots, x_m\}$ with length $m$, where $x_i$ denotes the $i$th tokens, a model is expected to generate a sentence $y = \{y_1, y_2, \ldots, y_n\}$ with length $n$ through model $f : x \rightarrow y$ to explain why the input statement is against common sense.

#### 3.2 Structure Overview

In our method, we adopt two separate pre-trained language models, one is for concepts extraction at the first stage, and we call this model as **conceptor** and denote as $C$. **Generator**, denoted as $G$, is used to generate explanations based on extracted concepts $c$ and the original statement $x$. The whole process can be formulated as

$$P(y, c | x) = \frac{P(c | M(x), \theta_C) \cdot P(y | x, c, \theta_G)}{P(c | \theta_C) \cdot P(x | \theta_G)}$$

(1)

#### 3.3 Concepts Extraction

This part is the $P(c | x, \theta_C)$ in equation 1. Firstly we modify inputs $x$ by attaching a template to create a masked sentence $M(x)$. The template we use is the template with Id 0 in Table 3 and an example is in Table 1. Then we use $C$ to encode $M(x)$. At this step, we can obtain the representation of the [MASK] tokens and sent it to a prediction head $W \in \mathbb{R}^{d \times N}$, where $d$ is the representation size and $N$ is the vocabulary size, to produce a probability of a word fits into the [MASK]. Subsequently, we select $k$ candidates $c = \{c_1, c_2, \ldots, c_k\}$ with the highest probability. $k$ is a hyperparameter and the default setting is 3. Concepts $c$ combined with input sentence $x$ are sent into model $G$ for explanation generation. In default setting, we only train our conceptor with 20% training data in a weak-supervision manner. We take the tokens that are in the ground truth explanations but not seen in
the input statements as the label. The detail about fine-tuning the model is described in section 5.4.

3.4 Explanation Generation

This section will describe the $P(y \mid x, c, \theta_G)$ process in equation 1. The concepts $c$ from previous step are concatenated at the end of the inputs $x$ to construct a new input $(x, c) = (x_1, x_2, \ldots, x_m, c_1, c_2, \ldots, c_k)$, and feed the model $G$ with the new input. The generator model follow the equation 2 to produce explanation.

$$P(y \mid x, c, \theta_G) = \prod_{t=1}^{n} P(y_t \mid x, c, y_{<t}, \theta_G) \tag{2}$$

The structure of the $G$ would be a sequence-to-sequence model, which contains an encoder and decoder. The Encoder would map inputs $(x, c)$ to a high-dimensional contextualized representations. The decoder would condition input representations and past generated sentences to generate the following sentences.

The generator $G$ would optimize its parameters towards minimizing the following loss function:

$$L = -\log P(y \mid x, c, \theta_G) \tag{3}$$

4 Experiments

4.1 Data

Similar to earlier work (Ji et al., 2020), we also evaluate the model on the dataset from Commonsense Validation and Explanation Competition. This challenge tests the commonsense knowledge through three subtasks: (1) selecting the statement with the proper commonsense knowledge from two similar options. (2) choosing the reason from three options to answer why a statement is against commonsense. (3) Coming up with explanations for why a certain statement is absurd. In this work, we focus on task 3, the explanation generation task.

This dataset collects 10,000 counter-commonsense claims, paired with three human-written explanations for each. We split each training data into three statement-explanation pairs. The final dataset is made up of 24,000(80%) training data, 1,000(10%) development data and 1,000(10%) testing data.

4.2 Model

We adopt various pre-trained language models as the concept extractor. Models we used are BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), with both base and large versions. Then we employ the T5-small (Raffel et al., 2020) model as the generator that accepts the combination of statements and concepts as inputs and outputs the explanation. The resulting explanations are evaluated using the automatic metrics, BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and METEOR (Banerjee and Lavie, 2005).

4.3 Implementation Details

We firstly trained the conceptor with 20% training data, then we use the trained conceptor to process all the data in training, development and testing sets to produce concepts for generator.

The default hyperparameter setting is following: number of concepts selected $k=3$, template is id 0 in Table 3, batch size is 8 for conceptor and 32 for generator training, training epoch is 5 for conceptor and 3 for generator, learning rate is $2e^{-5}$ for conceptor and $1e^{-3}$ for generator. The model with the best BLEU-4 score during evaluation would be saved and evaluated on test data.

4.4 Main Results

Table 2 is the main results. In comparison with the generator-only method (T5-small) and knowledge graph augmentation method, using a pre-trained language model as the extractor can improve most scores, and achieve the best results. Another observation is that large version models typically outperform their base version. We attribute the reason to that larger model tend to be more powerful (Liu et al., 2019; Hendrycks et al., 2020; Desai and Durrett, 2020; Brown et al., 2020). More analysis is located in section 5.4. Moreover, we can see that attaching concepts that were chosen randomly (Random) would reduce the result, indicating that only attaching appropriate concepts would best aid the generation process.

Additionally, to investigate the profit of concepts, we attached the tokens exclusive in references at the end of the statement and transmitted the combination to the generator. Consequently, we can observe a significant boost by attaching the correct concepts. Here, we define the “correct” of concepts as the tokens are not seen in the statement.
Table 2: Metric evaluation results for explanations from different methods. All these methods use T5-small as the generator. The bold number indicates the best results. **T5-small**: Using T5 to generate explanation without any concept augmentation. **Random**: Attaching concepts selected randomly. **Knowledge Graph**: Following the method in (Wang et al., 2019) that looking up concepts from ConceptNet. The model searching through ConceptNet is trained with 100% data. **BERT-base/large** and **RoBERTa-base/large** denote our method and the model we used as conceptor. The conceptor is trained with 20% data. **UpperBound**: Attaching the concepts that appear in references but do not exist in input statements. Therefore, we consider this result from **UpperBound** to be the upper limit of our method.

<table>
<thead>
<tr>
<th>Models</th>
<th>ROUGE-1/2/L/Lsum</th>
<th>METEOR</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-small</td>
<td>37.18/12.81/33.21/33.19</td>
<td>29.70</td>
<td>28.09</td>
<td>18.51</td>
</tr>
<tr>
<td>w/ Random concepts</td>
<td>34.15/11.64/29.93/29.16</td>
<td>27.37</td>
<td>26.44</td>
<td>16.37</td>
</tr>
<tr>
<td>w/ Knowledge Graph</td>
<td>37.01/12.60/33.23/33.21</td>
<td>29.96</td>
<td>28.36</td>
<td>19.33</td>
</tr>
<tr>
<td>w/ BERT-base</td>
<td>37.10/12.89/33.18/33.32</td>
<td>29.74</td>
<td>28.73</td>
<td>19.67</td>
</tr>
<tr>
<td>w/ BERT-large</td>
<td>37.20/13.52/33.52/33.51</td>
<td>30.02</td>
<td><strong>30.10</strong></td>
<td><strong>21.04</strong></td>
</tr>
<tr>
<td>w/ RoBERTa-base</td>
<td>37.15/13.38/33.25/33.20</td>
<td><strong>30.39</strong></td>
<td>29.39</td>
<td>20.10</td>
</tr>
<tr>
<td>w/ RoBERTa-large</td>
<td>37.33/13.26/33.38/33.37</td>
<td>30.34</td>
<td>29.62</td>
<td>20.23</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>43.75/25.74/38.26/38.31</td>
<td>-</td>
<td>47.93</td>
<td>42.55</td>
</tr>
</tbody>
</table>

but included in corresponding references. More discussion about retrieved concepts is in the section 5.5. This demonstrates how enhancing retrieving concept precision from language models would be beneficial for explanation generation. We have a deeper discussion in the section 5.5.

## 5 Analysis

### 5.1 Effects of Different Templates

Since template engineering is an important part of our method, we are curious to see whether these pre-trained models are sensitive to the design of templates. Table 3 shows the model’s performance with different manually created templates for prompt-tuning. From this table, although there are some variations of the BLEU-4 score, the standard deviation is only about 0.1259. Therefore, the choice of the template has a small impact on the metric score, which could indicate that pre-trained models are resistant to the difference in templates.

Moreover, we conduct additional ablation experiments. Results are in Table 4. We study what if adopting traditional fine-tuning method that takes the [CLS] representation for concept extraction rather than sitting a template. Even though models tend to promote under few-shot learning with prompt-based tuning (Gao et al., 2021), there is a small-scale decrease in the our result. The reason could be large amount of data would fill the gap between traditional fine-tuning and prompt-tuning. Next, we study the effects of ensemble of templates with the formula 4 and randomly selected templates with the formula 5.

\[
P(c \mid x, \theta_G) = \frac{1}{M} \sum_{i=1}^{M} P(c \mid \mathcal{M}_i(x), \theta_G) \quad (4)
\]

\[
P(c \mid x, \theta_G) = P(c \mid \mathcal{M}_i(x), \theta_G), 1 \leq i \leq M \quad (5)
\]

Where \( M \) is the number of distinct templates, and \( i \) denotes the Id of templates in 3. In formula 5, \( i \) is uniform randomly selected from the range \([1, 2, \ldots, M]\) for each training data. Table 4 shows these two methods can give enhancement compared to the default setting, and all single template results in Table 3. Therefore, one advantage of prompt-tuning data augmentations with templates.

### 5.2 The Number Of Attached Concepts

To further investigate the best number of retrieved concepts for models, we designed an experiment by gradually changing the hyperparameter \( k \). \( k \) denotes the number of tokens selected from masked language modelings and attached with statements. The results are shown in Figure 2. We use BLEU-3 and BLEU-4 to evaluate the model’s outputs, and the pre-trained models selected for retrieving concepts are the base version of BERT. The x-axis denotes the number of concepts \( k \), and the y-axis represents the evaluation scores on test data.

In this figure, the model provides the best scores over all metrics with \( k = 3 \). For \( k \) small than 3, the
### 5.3 Evaluating Retrieved Concepts

In this section, we would like to talk about how to evaluate retrieved concepts and introduce an automatic evaluation metric. We also provide human analysis in section 5.6.

Inspired by the concept F1 metric from (Ji et al., 2020) to evaluate sentences, tokens that in the grounded truth explanations but do not appear in the input sentences are considered the positive class. All other tokens in the vocabulary are regarded as the negative class.

In detail, we assume all unique tokens in tokenized input $x$ as $C_x$, unique tokens in tokenized ground truth explanation $y$ as $C_y$, and extracted concepts $\hat{y}$ from conceptor as $C_{\hat{y}}$. $C_y - C_x$ represents the desired concepts. Then we use precision to measure the match conditions among references and outputs.

$$\text{Precision} = \frac{|C_{\hat{y}} \cap (C_y - C_x)|}{|C_{\hat{y}}|}$$

$C_y \cap (C_y - C_x)$ are overlapped concepts, and $|\cdot|$ denotes the number, $|C_{\hat{y}}|$ is usually the same as $k$. This metric result shows how many retrieved concepts are aligned with the grounded truth explanations. We choose not to use recall, equation 7, for evaluation as we think the recall in our experiment setting can not properly measure the retrieved concepts and ground truth.

$$\text{Recall} = \frac{|C_{\hat{y}} \cap (C_y - C_x)|}{|C_y|}$$

Then we use the precision in equation 6 to evaluate the concepts collected from BERT-base and RoBERTa-large. The result is shown in Table 5. We can observe these scores are low when zero-shot, but the precision is much improved by just

---

**Table 3: Templates used in our method. [DATA] wold replaced by original data. The template are created manually and have semantic meanings.**

<table>
<thead>
<tr>
<th>Id</th>
<th>Template</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Statement that [DATA] is unreasonable because of knowledge about [MASK]</td>
<td>19.67</td>
</tr>
<tr>
<td>1</td>
<td>knowledge about [MASK] can refute that [DATA]</td>
<td>19.72</td>
</tr>
<tr>
<td>2</td>
<td>[DATA] is wrong because bridge concept of [MASK]</td>
<td>19.84</td>
</tr>
<tr>
<td>3</td>
<td>[DATA] can be explained by knowledge about [MASK]</td>
<td>19.58</td>
</tr>
<tr>
<td>4</td>
<td>knowledge about [MASK] disproves statement that [DATA]</td>
<td>19.89</td>
</tr>
</tbody>
</table>

**Table 4: Template Ablation Study with BERT-base performance is proportional to $k$. However, when $k$ exceeds 3, the results decrease as $k$ increases. We argue that although our method can bring some concepts that would benefit the explanation generation stage, it also would inevitably include some noisy tokens that would confuse the generation model. As long $k$ is less than 3, the benefit from retrieved concepts outperforms the noise. However, when the $k$ becomes larger than $k$, the noise would reduce the advantage of these concepts, thus decreasing the scores. Therefore, in our future work, the key to enhancing the method would be raising the accuracy of concept retrieval and reducing the noise.**

![Figure 2: Performance by changing the number of top-$k$ concepts selected.](image)

1The average number of concepts in each data point is 7.26.
Table 5: The precision metrics scores for concepts from pre-trained language models with different top-k. the number before % means the percent of data used to train the conceptor. 0% is zero-shot and 100% is fully fine-tuning. The higher the better.

<table>
<thead>
<tr>
<th>Data</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>2.40</td>
<td>1.87</td>
<td>2.16</td>
<td>2.36</td>
<td>2.08</td>
</tr>
<tr>
<td>20%</td>
<td>20.00</td>
<td>16.16</td>
<td>13.02</td>
<td>9.55</td>
<td>6.71</td>
</tr>
<tr>
<td>100%</td>
<td>30.60</td>
<td>22.20</td>
<td>17.84</td>
<td>12.61</td>
<td>8.59</td>
</tr>
<tr>
<td>RoBERTa-large</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>2.10</td>
<td>2.03</td>
<td>2.28</td>
<td>2.12</td>
<td>2.23</td>
</tr>
<tr>
<td>20%</td>
<td>38.20</td>
<td>26.73</td>
<td>21.18</td>
<td>14.78</td>
<td>9.84</td>
</tr>
<tr>
<td>100%</td>
<td>47.50</td>
<td>34.80</td>
<td>28.42</td>
<td>20.15</td>
<td>13.83</td>
</tr>
</tbody>
</table>

5.4 Fine-tuning The Conceptor

In last section, we observe that training with more data and larger models can increase the precision of concepts retrieval, but we also wonder whether these more accurate concepts would benefit the final results. The same as the precision described in section 5.3, we set all the concepts in \((C_y - C_k)\) as desired concepts for masked language modeling and denote it as label \(C\). Because this task can be regarded as multi-label learning, we adopt the sigmoid function to map the output scores into probability and keep each value independent. Then we optimize conceptors by cross-entropy loss.

As the results in Table 6, fine-tuning increases metric scores with percent of data used. Comparing zero-shot and fully-training, there is 1.31 BLEU-4 increase for BERT and 3.12 BLEU-4 increase for RoBERTa. Therefore, this evidence underlines the connections between the concept precision metric and model performance. In additional, RoBERTa is better than BERT over all metrics with trained with the same percent of data, the same as observations before.

5.5 How Retrieved Concepts Affect Generations

Our proposed method can make concept extraction simpler and efficient and benefit the generation process. However, current methods still face some difficulty in guiding pre-trained language models to return correct concepts as the scores in Table 5 are still deficient.

Therefore, we design experiments in which \(m\) of \(k\) concepts are fixed as correct and \(k - m\) concepts as noisy tokens, where \(1 \leq m \leq k\). Specifically, we randomly sample \(m\) concepts that appear in the ground truth explanations but are not included in the input sentence, as well as pick up \(m - k\) noisy concepts from the vocabulary at random. Then shuffling these \(k\) concepts followed by attaching these concepts at the tail of statements for generation. We ran experiments with \(k \in \{1, 3, 5, 8\}\) and \(m\) increases from 1 to \(k\) for each \(k\), and evaluate with BLEU-3 and BLEU-4 metrics.

Figure 3 displays the results. We see that the score rises with the number of correct concepts increasing. Additionally, smaller \(k\) typically perform better when having the same \(m\). Models only need an expectation of 2 to easily surpass our best results in Table 2. These observations suggest that enhancing models’ capacity to identify better bridge concepts could be the focus of future work for generating commonsense sentences.

5.6 Human Analysis

As these automatic evaluation metrics has the difficulty of heavily relying on the quality of human references, we also provide a human investigation to see whether our method can help generation compared with baseline. We collect 50 statements paired with concepts, generated explanations and grounded truth explanations. Then one human evaluate these data with 4 standards. **Correct**: whether retrieved concepts occur in the grounded truth explanations, no matter whether these concepts are
Table 6: Consequences after fine-tuning conceptors with $k = 3$. Models represents different concept extractors. % denotes the percent of data used for training.

<table>
<thead>
<tr>
<th>Models</th>
<th>ROUGE-1/2/L/Lsum</th>
<th>METEOR</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-base</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>37.15/12.84/33.11/33.13</td>
<td>29.31</td>
<td>28.38</td>
<td>19.16</td>
</tr>
<tr>
<td>20%</td>
<td>37.10/12.89/33.18/33.24</td>
<td>29.74</td>
<td>28.73</td>
<td>19.67</td>
</tr>
<tr>
<td>50%</td>
<td>37.30/13.07/33.80/33.75</td>
<td>29.71</td>
<td>28.86</td>
<td>19.83</td>
</tr>
<tr>
<td>100%</td>
<td>37.50/13.15/33.91/33.89</td>
<td>29.84</td>
<td>29.61</td>
<td>20.47</td>
</tr>
<tr>
<td>RoBERTa-large</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>37.09/12.77/33.16/33.18</td>
<td>29.78</td>
<td>28.30</td>
<td>18.60</td>
</tr>
<tr>
<td>20%</td>
<td>37.33/13.16/33.38/33.37</td>
<td>30.24</td>
<td>29.62</td>
<td>20.23</td>
</tr>
<tr>
<td>50%</td>
<td>37.58/13.63/33.62/33.68</td>
<td>30.28</td>
<td>30.00</td>
<td>20.78</td>
</tr>
<tr>
<td>100%</td>
<td>37.84/13.88/34.97/33.95</td>
<td>30.46</td>
<td>30.69</td>
<td>21.72</td>
</tr>
</tbody>
</table>

Table 7: Human evaluation of generated explanation and selected concepts

<table>
<thead>
<tr>
<th>Model</th>
<th>Correct</th>
<th>Reasonable</th>
<th>Dependent</th>
<th>Relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-small</td>
<td>-</td>
<td>0.400</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>w/ KG</td>
<td>0.147</td>
<td>0.440</td>
<td>0.12</td>
<td>0.253</td>
</tr>
<tr>
<td>w/ BERT</td>
<td>0.180</td>
<td>0.520</td>
<td>0.133</td>
<td>0.293</td>
</tr>
</tbody>
</table>

6 Conclusion

In this paper, we present a combining prompt-tuning and pre-trained language models concept extraction method for commonsense explanation generation task. We demonstrate that our method is more data-efficient for concept extractions than methods that selecting concepts from knowledge graph, and our method improves evaluation metric scores compared with baseline models. Additionally, we analyze the method of selecting concepts from pre-trained language model and challenges to solve in future work.

Limitation

In our experiment, we find no very efficient methods and pre-trained language models to retrieve concepts with sufficient precision. We believe this limitation would restrict the generator’s capacity to generate explanations, and future work would focus on proposing methods to improve the concept retrieval precision. Additional limitations would be we haven’t tested our method with alternative generation models for the generator, while the generation stage is not our focus of this work. Our method still can not explain how the generator understands statements and concepts to generate explanations.

Acknowledgement

This work was supported in part by NSF IIS-1910154, 2007907, and 2046457. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

References

Example 1

<table>
<thead>
<tr>
<th>Statement</th>
<th>mustard must go on all sandwiches.</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-small</td>
<td>mustard is not a condiment.</td>
</tr>
<tr>
<td>w/ KG</td>
<td>mustard is not edible.</td>
</tr>
<tr>
<td>w/ BERT</td>
<td>it is unnecessary to have mustard</td>
</tr>
<tr>
<td>Concepts KG</td>
<td>sandwich cook edible</td>
</tr>
<tr>
<td>Concepts BERT</td>
<td>people unnecessary food</td>
</tr>
</tbody>
</table>

Example 2

| T5-small  | he was taking bath in the kitchen. |
| Baseline  | you can’t take a bath.             |
| w/ KG     | kitchens are not a place for bathing. |
| w/ BERT   | you take bath is bath room.        |
| Concepts KG | truth bath power                |
| Concepts BERT | bath meat room                  |

Example 3

| T5-small  | I want to learn how to play a dog |
| Baseline  | dogs are not for play             |
| w/ KG     | you should not play a dog.        |
| w/ BERT   | dogs are not instruments.         |
| Concepts KG | game learning animal          |
| Concepts BERT | instrument fun video         |

Table 8: Sample cases of extracted concepts and generated explanations. Red color means concepts appeared in generated explanations.


Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In AAAI.


