The role of fine-tuning method in commonsense generation with pretrained language models

Anonymous ACL submission

Abstract

In this paper we study different methods for fine-tuning a pre-trained language model to generate commonsense knowledge. We particularly focus on the T5 model in few-shot settings. This model can be trained in unsupervised and supervised fashions. We investigate the effect of each type of these fine-tuning methods on the performance of the model. The results shows that the unsupervised method can generate more diverse and novel knowledge by relying more on the stored knowledge of the model. We also analyze parameter changes during few-shot fine-tuning to gain more insights on the way the model learns using these methods.

1 Introduction

Recently, pre-trained language models (PLMs) have been used as knowledge bases. Petroni et al. tried to retrieve factual and commonsense knowledge directly from these models by converting a query into cloze-style prompts (Petroni et al., 2020). This approach is more suitable for retrieving one-to-one factual knowledge. When prompted to complete commonsense declarative relationships, PLMs exhibit limited ability to map their language modeling abilities to this task (Petroni et al., 2020; Da et al., 2021b).

Other works attempted to fine-tune a PLM on a commonsense knowledge graph tuples for completion and generation of commonsense tuples (Bosset-Lu et al., 2020; Hwang et al., 2020; Da et al., 2021a). The tuples are typically in the form of (head, relation, tail). The model must generalize commonsense relationships to situations it has not seen during fine-tuning and provide a plausible tail.

In these works, it is assumed that commonsense knowledge is implicitly encoded in the pretrained model, and fine-tuning serves to learn an interface to the encoded knowledge. With this assumption, a commonsense knowledge model can be trained effectively in a few-shot setting to hypothesize commonsense knowledge (Da et al., 2021a). In (Da et al., 2021b), the authors tried to verify this assumption by analyzing the changes in model parameters during few-shot fine-tuning. They found that the changes are mainly in the direct of expressing the encoded knowledge.

One of the methods that can accelerates learning commonsense knowledge is the use of natural language prompts to elicit knowledge from PLMs (Feldman et al., 2020; Da et al., 2021a). Prompts are mainly effective in a few-shot setting where there is little signal to learn a relation embedding from scratch.

The related works usually employed encoder-decoder models such as T5 (Raffel et al., 2020) to generate the tail of a triplet by feeding the model with the head and the relation. The relation is either a special token appended to the head as a prefix, or is represented using natural language prompts.

However, the current works often fine-tune the model in a supervised fashion where a mapping must be made between the input and output. One drawback of this method is that the model may overfit to the training data and its generalization power decreases. As the result, it mainly memorize the most repeated patterns in the training data. We assumed that a fine-tuning method that is more similar to the self-supervised method in which the model was pretrained on a large amount of text data can provide better results and allows for a better use of the knowledge encoded in language model.

In this work, we investigate commonsense knowledge model performance under different training settings for few-shot learning. We especially try to understand if fine-tuning the model in an unsupervised fashion is effective on the results or not. Similar to (Da et al., 2021b), we analyze the parameter changes in different settings and compare our results with their findings.
2 Experimental Setup

2.1 Model

In the related works several generative models such as GPT2-LX as an autoregressive model, and BART and T5 as encoder-decoder models were employed. Among them, the BART and T5 models provided better results. The GPT model tended to copy the input or generate unnecessary full sentences (Da et al., 2021b,a). It was also observed in our experiments. Therefore, we decided to focus on the T5 model, which provided better results.

T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks. Each task is converted into a text-to-text format. T5 can be fine-tuned both in a supervised and unsupervised fashion. In the following, we review these methods.

• **Supervised Training** In this setup, the input sequence and output sequence are a standard sequence-to-sequence input-output mapping. A prefix representing the task instruction can be appended to each input example.

• **Unsupervised Training** The T5 model was pre-trained on a huge collection of unlabeled texts in an unsupervised method. In this setup, spans of the input sequence are randomly sampled and masked by so-called sentinel tokens (a.k.a unique mask tokens). The target then corresponds to all of the dropped-out spans of tokens, delimited by the same sentinel tokens used in the input sequence plus a final sentinel token to mark the end of the target sequence. The model is trained to reconstruct the original sequence by predicting the dropped-out spans of tokens.

Figure 1 shows an example of the input and target of the unsupervised pre-training objective.

2.2 Dataset

One of the main commonsense knowledge bases used in the related works is the ATOMIC knowledge base (Sap et al., 2019). It is an atlas of everyday commonsense reasoning, organized through 877k textual descriptions of inferential knowledge. The tuples are in the form of a \{head h, relation r, tail t\} triplets. The head is the description of a situation involving social agents and the tails are social commonsense relating to them along nine dimensions, such as the causes and the effects of the event, and the reaction of the participants. The tuples can be organized in the form of typed if-then relations with variables (e.g., “if PersonX goes to a library, he probably wants to borrow a book”). Each head can have multiple tails for each relation.

ATOMIC split into training, development, and test subsets such that no head entities in one set appear in any other. Models trained on this knowledge base can generate a plausible sequence by receiving a situation and a relation.

2.3 Training

To train the model, the head and the relation of each example are fed to the model and the tail is used as the target. The model is trained to minimize the negative log-likelihood of the tokens of the tail entity for each tuple. We use the AdaFactor optimizer with a constant learning rate of 0.001, a mini-batch size of 4, and train the model for 3 epochs. Unless stated otherwise, we use T5-large as a seed language model for all experiments. In fewshot settings, we set the number of examples per relation $n=30$ and 300 examples per relations.

2.4 Evaluation

The generated tails for a given head and relation can be evaluated in terms of plausibility as well as variety and novelty with both automatic metrics and human evaluation. Due to the cost of human annotation, and the number of experiments pursued, we use automated metrics on the a random subset of the test set with 1000 examples. Also, since in the related works and our experiments, different metrics exhibit a linear relationship, here we just report the results of the ROUGE score.

2.5 Methods

Table 1 shows different methods to format the input and the output of the model and the type of training.
Table 1: Different Methods to format the input and output of a T5 model

<table>
<thead>
<tr>
<th>Name</th>
<th>Format Training setup</th>
<th>Training setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>sup</td>
<td>Input: <code>&lt;xIntent&gt;</code> PersonX goes to a library to borrow a book</td>
<td>Supervised</td>
</tr>
<tr>
<td></td>
<td>Target:</td>
<td></td>
</tr>
<tr>
<td>sup-nat</td>
<td>Input: PersonX goes to a library <code>because he wants</code> to borrow a book</td>
<td>Supervised</td>
</tr>
<tr>
<td></td>
<td>Target:</td>
<td></td>
</tr>
<tr>
<td>unsup-nat</td>
<td>Input: PersonX goes to a library <code>because he wants</code> <code>&lt;X&gt;</code> to borrow a book</td>
<td>Unsupervised</td>
</tr>
<tr>
<td></td>
<td>Target:</td>
<td><code>&lt;X&gt;</code></td>
</tr>
<tr>
<td>unsup</td>
<td>Input: PersonX goes to a library <code>&lt;Intent&gt;</code> <code>&lt;X&gt;</code> to borrow a book</td>
<td>Unsupervised</td>
</tr>
<tr>
<td></td>
<td>Target:</td>
<td><code>&lt;X&gt;</code></td>
</tr>
</tbody>
</table>

setup. These methods are as the following:

- **sup**: This method is the same used in (Bose-lut et al., 2020) in which the model is fine-tuned in a supervised fashion and a unique token is appended to the the tokens of head entity for each relation. This token maps to a unique learnable embedding for that relation.

- **sup-nat**: This method is similar to the above, but the input tuples are formatted into natural language prompts to represent the relations. We used the templates introduced in (Da et al., 2021b) for each relation.

- **unsup-nat**: This method is again similar to sup-nat method in that the relations are mapped to natural language prompts. However, according to T5 unsupervised training format, we use a unique mask token `<X>` as a placeholder for the tail. The input and the missing tail when replaced form a natural language sentence. We named it unsup because it is similar to the unsupervised method used in the pretraining stage of the T5 model. To augment data in this setup, other tokens of the full sentence can also be omitted, however we leave it to a future work.

- **unsup**: To differentiate the effect of natural language prompt and the special tokens used for the unsupervised training, we introduced this method in which the relation is represented by a special token (like sup method) and the mask token `<X>` is appended to the input.

### 2.6 Performance in fewshot settings

We evaluated the few-shot commonsense learning capability of the T5-large model. Table 2 shows the results for different methods described in the previous section using $n = 30$ and 300 examples per relations.

**Findings** Using $n = 30$ examples per relation, the sup-nat method provides better results. This finding is in agreement with the related works (Da et al., 2021a) and shows that using natural language prompts for relation is effective in a few-shot setting. However, this advantage vanishes when the number of examples increases.

The unsupervised methods (unsup-nat and unsup) produce results similar to the sup method in the few shot settings. However, when we use natural language prompts together with the placeholder token `<X>`, the diversity of generated tails increases. The model tries to complete each sentences with probable compliments. This feature can balance between memorizing the training set and generating novel tails. However, it seems the model needs more data to converge to an optimum. When the number of examples increases, the model produces better results, and at the same time preserve their diversity relative to the other methods.

In the next section we investigate to see whether there are difference between the way these methods learn.

### 2.7 Analyzing Learning of Methods

To analyse and compare different methods during fine-tuning, we use the measures presented in (Da et al., 2021b).

**Absolute Parameter Change** This measure quantifies the average change in each parameter matrix during fine-tuning by computing a normalized $\ell_1$ distance for each set of parameter matrices in the transformer blocks. The normalized $\ell_1$ distance between these two matrices for each parame-
we found that in the supervised methods, most parameter changes occur in the decoder cross attention, while the unsupervised methods attend more to tokens of input. These methods can produce more diverse and novel knowledge by relying more on the stored knowledge in a declarative form.

However, we found that the supervised and unsupervised methods exhibit similar patterns for all parameter changes. Contrary to (Da et al., 2021b), we found that in the supervised methods, most changes occur in the feedforward layers of middle layers which are proposed to serve as an an implicit memory storage for information learned during pretraining (Geva et al., 2020). This can be expected because the supervised methods try to map inputs to the outputs.

Similar to (Da et al., 2021b), we also found that the absolute and angular distance heatmaps for prompt and embedding relation inputs are similar, implying that most of the parameter change during fine-tuning is not linked to the way the relation is represented.

However, in the case of unsupervised methods, the main changes occur in the self and cross attention layers. However the changes are not as high as supervised methods. This can shows that similar to the pre-training stage, the model tires to fill the masked tokens by attending to the input and previous tokens. Interestingly, in these methods, the angular change occurs more in the early layers of decoder cross attention ($v_x$ and $o_x$), while the $\ell_1$ distance change occurs in the final layers of these parameters, suggesting that the model attend more to the input to generate the output at early layers.

### 3 Conclusion

In this paper we compared different methods for finetuning a T5 model to generate commonsense knowledge. We proposed that converting the training examples to a format similar to the unsupervised training of the model can provide results which are different from a supervised fashion. These methods can produce more diverse and novel knowledge by relying more on the stored knowledge in the PLM. We also analyzed the model parameters during fine-tuning using these methods. We found that supervised methods regardless of the type of relation tend to map the inputs to outputs, while the unsupervised methods attend more to tokens of input.

### References


Jeff Da, Ronan Le Bras, Ximing Lu, Yejin Choi, and Antoine Bosselut. 2021a. Understanding few-shot
Figure 2: Comparison of parameter changes during fine-tuning of the T5 large model using different methods and \( n = 30 \) examples per relation. (from left, sup, sup-nat, unsup, unsup-nat) The blue ones are the change in normalized \( \ell_1 \) distance for each parameter matrix, and the green ones are the change in the angular distance. The decoder part is on top and the encoder part is on bottom commonsense knowledge models. *arXiv preprint arXiv:2101.00297*.


