Investigating the true capability of pretrained language models in understanding causality in text is still an open question. More recently, Knowledge Graphs (KGs) have been used in combination with pretrained language models to address commonsense reasoning. Two examples are Causal-BERT (Li et al., 2020) for guided generation of Cause and Effect and the model introduced by Guan et al. (2020) for commonsense story generation. Motivated by the success of continual pretraining of PLMs for downstream tasks (Gururangan et al., 2020), we explore the impact of commonsense knowledge injection as a form of continual pretraining for causal reasoning and cause-effect relation classification. It is worth highlighting that even though there are studies to show the efficacy of knowledge injection with continual pretraining for commonsense reasoning (Guan et al., 2020), performance of these techniques is very dependent on the domain and downstream tasks (Gururangan et al., 2020). And, to the best of our knowledge, there are limited studies on the effect of commonsense knowledge injection with knowledge graph data on cause-effect relation classification (Dalal et al., 2020).
Our contributions are as follows:

- We study performance of PLMs augmented with knowledge graph data in the less investigated cause-effect relation classification task.

- We demonstrate that a simple masked language modeling framework using automatically verbalized knowledge graph triples, without any further model improvement (e.g., new architecture or loss function) or quality enhanced data for fine-tuning, can significantly boost the performance in cause-effect pair classification.

- We publicly release our knowledge graph verbalization codes and models that are continually pretrained on cloud TPUs.

2 Method

The overview of our method is shown in Figure 1. We first convert triples in ATOMIC20 (Hwang et al., 2021) knowledge graph to natural language texts. Then we continually pretrain BERT using Masked Language Modeling (MLM) and evaluate performance of the resulting model on different benchmarks. Samples in ATOMIC20 are stored as triples in form of (head/subject, relation, tail/target) in three splits including train, development, and test. ATOMIC20 has 23 relation types that are classified into three categorical types including commonsense relations of social interactions, physical-entity commonsense relations, and event-centric commonsense relations. In the rest of the paper, we refer to these three categories as social, physical, and event, respectively.

Filtering triples: We remove all duplicates and ignore all triples in which the target value is none. Moreover, we ignore all triples that include a blank. Since in masked language modeling we need to know the gold value of masked tokens, a triple that already has a blank (masked token/word) in it may not help our pretraining. For instance, in the triple: [PersonX affords another ____, xAttr, useful] it is hard to know why or understand what it means for a person to be useful without knowing what they afforded. This preprocessing step yields in 782,848 triples with 121,681, 177,706, and 483,461 from event, physical, and social categories, respectively.

Converting Triples: Each relation in ATOMIC20 is associated with a human-readable template. For example, xEffect’s and HasPrerequisite’s templates are as a result, PersonX will and to do this, one requires, respectively. We use these templates to convert triples in ATOMIC20 to sentences in natural language by concatenating the subject, relation template, and target. Examples of converting triples to text are shown in Figure 2.

Checking Grammar: When we convert triples to natural language text, ideally we want to have grammatically correct sentences. Human readable templates provided by ATOMIC20 are not necessarily rendered in a way to form error-free sentences when concatenated with subject and target in a triple. To address this issue, we use an open-source grammar and spell checker, LanguageTool,1 to double-check our converted triples to ensure they do not contain obvious grammatical mistakes or spelling errors. Similar approaches that include deterministic grammatical transformations were also previously used to convert KG triples to coherent sentences (Davison et al., 2019). It is worth pointing out that the Data-To-Text generation (KG verbalization) for itself is a separate task and there have been efforts to address this task (Agarwal et al., 2021). We leave investigating the effects of using other Data-To-Text and grammar-checking methods to future research.

3 Experiments

3.1 Benchmarks

We chose multiple benchmarks of commonsense causal reasoning and cause-effect relation classification to ensure we thoroughly test the effects of our newly trained models. These benchmarks include: 1) Temporal and Causal Reasoning (TCR) dataset (Ning et al., 2018), a benchmark for joint reasoning of temporal and causal relations; 2)...

1https://tinyurl.com/yc77k3fb
Choice Of Plausible Alternatives (COPA) (Roemmele et al., 2011) dataset which is a widely used and notable benchmark (Rogers et al., 2021) for commonsense causal reasoning; And 3) BCOPA-CE (Han and Wang, 2021), a new benchmark inspired by COPA, that contains unbiased token distributions which makes it a more challenging benchmark. For COPA-related experiments, since COPA does not have a training set, we use COPA’s development set for fine-tuning our models and testing them on COPA’s test set (COPA-test) and BCOPA-CE. For TCR, we fine-tune and evaluate our models on train and test splits, respectively. In all experiments, we report the average performance of models using four different random seeds.

### 3.2 Models and Baseline

We use bert-large-cased pre-trained model in all our experiments as our baseline. For COPA and BCOPA-CE, we convert all instances to a SWAG-formatted data (Zellers et al., 2018) and use Huggingface’s BertForMultipleChoice—a BERT model with a multiple-choice classification head on top. And for TCR, we convert every instance by adding special tokens to input sequences as event boundaries and use the R-BERT \(^1\) model (Wu and He, 2019). We chose R-BERT for our relation classification since it not only leverages the pretrained embeddings but also transfers information of target entities (e.g., events in a relation) through model’s architecture and incorporates encodings of the targets entities.

Examples of COPA and TCR are shown in Figure 3. BCOPA-CE has the same format as COPA.

\[ \text{asks-for} = \text{cause} \]

\[ P: \text{The computer crashed.} \]

\[ H_1: \text{I backed up my files.} \]

\[ H_2: \text{I downloaded a virus.} \]

\[ \text{[CLS]} \text{The computer crashed.} \]

\[ \text{[SEP]} \text{I backed up my files.} \]

\[ \text{[SEP]} \text{I downloaded a virus.} \]

\[ \text{[CLS]} \text{The computer crashed.} \]

\[ \text{[SEP]} \text{A suicide car bomb exploded.} \]

\[ \text{[SEP]} \text{Friday in the middle of a group of men playing volleyball in northwest Pakistan, police said.} \]

\[ \text{Cause-Effect(e1,e2)} \]

Figure 3: COPA and TCR examples. The COPA instance is converted to Multiple Choice format.

### 3.3 Continual Pretraining

As mentioned earlier, we use (MLM) to continually pretrain our PLM, BERT-large-cased (Devlin et al., 2018). We follow the same procedure as BERT to create the input data to our pretraining (e.g. number of tokens to mask in input examples). We run the pretraining by default for 10 epochs on a Google Colab TPU v2 using PyTorch/XLA package with a maximum sequence length of 30 and batch size of 128 and save the checkpoints at every 500 steps. To avoid overfitting, we use early stopping with the patience of 3 on evaluation loss. We select the best model based on the lowest evaluation loss at the end of training. \(^4\)

### 4 Results and Discussion

Results of our experiments on TCR are shown in Table 1. As can be seen, our model significantly outperforms both our baseline and the joint inference framework by Ning et al. (2018) formulated as an integer linear programming (ILP) problem.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint system (Ning et al., 2018)</td>
<td>77.3</td>
</tr>
<tr>
<td>BERT-large (baseline) (\Phi)</td>
<td>75.0</td>
</tr>
<tr>
<td>ATOMIC-BERT-large (\Phi)</td>
<td>91.0</td>
</tr>
</tbody>
</table>

Table 1: TCR Accuracy results. \(\Phi\) Our models

Results of experiments on COPA-test are shown in Table 2. We initially observed that a continually pretrained model using all three types of relations has a lower performance than our baseline. By taking a closer look at each relation type, we decided to train another model, this time only using the event relations. The reason is that event-centric relations in ATOMIC\(^2\) specifically contain commonsense knowledge about event interaction for understating likely causal relations between events in the world (Hwang et al., 2021). In addition, event relations have a relatively longer context (# of tokens) than the average of all three relation types combined which means more context for a model to learn from. Our new pretrained model outperformed the baseline by \(\%\) 5 which shows the effect of augmented pretrained language model with commonsense reasoning knowledge.

We further experiment on the Easy and Hard question splits in COPA-test separated by Kavumba et al. (2019) to see how our best model performs on harder questions that do not contain superficial cues. Results are shown in Table 3. As can be seen, our ATOMIC-BERT model significantly out-

\(^2\)We use the following implementation of R-BERT: https://github.com/monologg/R-BERT

\(^3\)\%99.99 of ATOMIC\(^2\) instances have 30 tokens or less.

\(^4\)We use Huggingface’s BertForMaskedLM implementation. All codes and trained models will be publicly available.
Table 2: COPA-test Accuracy results. ✫ Our models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMI (Roemmele et al., 2011)</td>
<td>58.8</td>
</tr>
<tr>
<td>b-l-reg (Han and Wang, 2021)</td>
<td>71.1</td>
</tr>
<tr>
<td>Google T5-base (Raffel et al., 2019)</td>
<td>71.2</td>
</tr>
<tr>
<td>BERT-large (Kavumba et al., 2019)</td>
<td>76.5</td>
</tr>
<tr>
<td>CausalBERT (Li et al., 2020)</td>
<td>78.6</td>
</tr>
<tr>
<td>BERT-SocialIQA (Sap et al., 2019)*</td>
<td>80.1</td>
</tr>
<tr>
<td>BERT-large (baseline) ✫</td>
<td>75.9</td>
</tr>
<tr>
<td>ATOMIC-BERT-large ✫</td>
<td>74.3</td>
</tr>
<tr>
<td>- Event, Physical, Social</td>
<td>80.9</td>
</tr>
<tr>
<td>- Event only</td>
<td>58.1</td>
</tr>
<tr>
<td>Google T5-11B (Raffel et al., 2019)</td>
<td>94.8</td>
</tr>
<tr>
<td>DeBERTa-1.5B (He et al., 2020)</td>
<td>96.8</td>
</tr>
</tbody>
</table>

Table 3: COPA-test Accuracy results on Easy and Hard question subsets. ✫ Our models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Easy ↑</th>
<th>Hard ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Han and Wang, 2021)</td>
<td>-</td>
<td>69.7</td>
</tr>
<tr>
<td>(Kavumba et al., 2019)</td>
<td>83.9</td>
<td>71.9</td>
</tr>
<tr>
<td>BERT-large (baseline) ✫</td>
<td>84.1</td>
<td>69.7</td>
</tr>
<tr>
<td>ATOMIC-BERT-large ✫</td>
<td>89.1</td>
<td>75.9</td>
</tr>
</tbody>
</table>

Table 4: BCOPA-CE Accuracy results. ✫ Our models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b-l-aug (Han and Wang, 2021)</td>
<td>51.1</td>
</tr>
<tr>
<td>b-l-reg (Han and Wang, 2021)</td>
<td>64.1</td>
</tr>
<tr>
<td>BERT-large (baseline) ✫</td>
<td>55.8</td>
</tr>
<tr>
<td>ATOMIC-BERT-large ✫</td>
<td>74.3</td>
</tr>
<tr>
<td>- Event, Physical, Social</td>
<td>54.1</td>
</tr>
<tr>
<td>- Event only</td>
<td>80.9</td>
</tr>
</tbody>
</table>

4.1 BCOPA-CE: Prompt vs. No Prompt

Results of experiments on BCOPA-CE are shown in Table 4. As expected based on the results also reported by Han and Wang (2021), we initially observed that our models are performing nearly as random baseline. Since we do not use the type of question when encoding input sequences, we decided to see whether adding question type as a prompt to input sequences will improve the performance. We added It is because and As a result, as prompt for asks-for="cause" and asks-for="effect", respectively. Interestingly, the new model outperforms the baseline and Han and Wang (2021)'s b-l-aug model that is fine-tuned with the same data as ours, when question types are added as prompts to input sequences of correct and incorrect answers in the test set. We also ran a similar experiment on COPA-test (Table 5) in which adding prompt did not help with performance improvement.

5 Conclusion

We introduced a simple framework for augmenting PLMs with commonsense knowledge created by automatically verbalizing ATOMIC \(^{20}\). Our results show that commonsense knowledge-augmented PLMs outperform the original PLMs on cause-effect pair classification and answering commonsense causal reasoning questions. As the next step, it would be interesting to see how the previously proposed model improvement methods or using unbiased fine-tuning datasets can potentially enhance the performance of our knowledge-augmented models.
References


Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and...


Zhongyang Li, Xiao Ding, Ting Liu, J Edward Hu, and Benjamin Van Durme. 2020. Guided generation of cause and effect. IJCAI.


