ConstraintChecker: A Plugin for Large Language Models to Reason on Commonsense Knowledge Bases

Quyet V. Do, Tianqing Fang, Shizhe Diao, Zhaowei Wang, Yangqiu Song
Department of Computer Science and Engineering, HKUST, Hong Kong SAR, China
{vqdo, tfangaa, sdiaoaa, zwanggy, yqsong}@cse.ust.hk

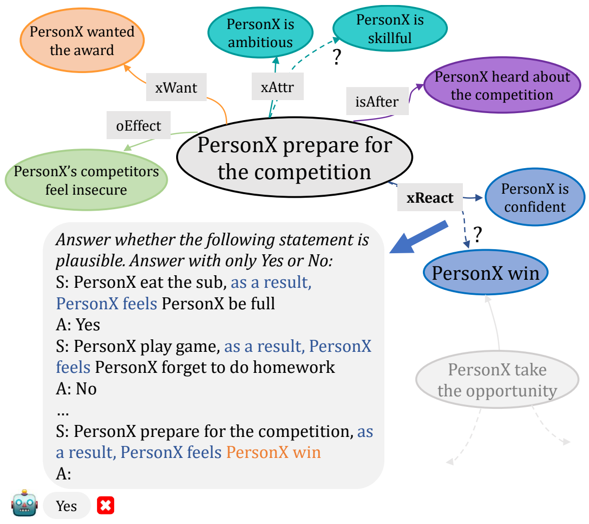
Abstract

Reasoning over Commonsense Knowledge Bases (CSKB), i.e., CSKB reasoning, has been explored as a way to acquire new commonsense knowledge based on reference knowledge in the original CSKBs and external prior knowledge. Despite the advancement of Large Language Models (LLM) and prompt engineering techniques in various reasoning tasks, they still struggle to deal with CSKB reasoning. One of the problems is that it is hard for them to acquire explicit relational constraints in CSKBs from only in-context exemplars, due to a lack of symbolic reasoning capabilities (Bengio et al., 2021). To this end, we proposed **ConstraintChecker**, a plugin over prompting techniques to provide and check explicit constraints. When considering a new knowledge instance, ConstraintChecker employs a rule-based module to produce a list of constraints, then it uses a zero-shot learning module to check whether this knowledge instance satisfies all constraints. The acquired constraint-checking result is then aggregated with the output of the main prompting technique to produce the final output. Experimental results on CSKB Reasoning benchmarks demonstrate the effectiveness of our method by bringing consistent improvements over all prompting methods. Codes and data are available at `https://github.com/HKUST-KnowComp/ConstraintChecker`.

1 Introduction

Commonsense Knowledge Bases (CSKB) Reasoning, as one of many commonsense reasoning tasks, has been well explored in Natural Language Processing for the past few years. As human-annotated CSKBs (Speer et al., 2017; Sap et al., 2019; Mostafazadeh et al., 2020) are usually incomplete and of a small coverage, reasoning over CSKBs, i.e., CSKB reasoning, is a way for expansion. CSKB reasoning is defined as determining whether a new knowledge triple *(head event, relation, tail event)* is commonsense (in other expressions, being plausible or having label 1) based on the reference knowledge in original CSKBs as well as external prior knowledge (Fang et al., 2023; Davison et al., 2019). Expanding CSKBs via such a reasoning process can lead to better and broader commonsense knowledge as valuable resources to augment AI models in various aspects, such as visual reasoning (Zellers et al., 2019), text generation (Zhou et al., 2021; Ilievski et al., 2021), or building more capable knowledge models for further downstream applications (Yu et al., 2022; Hwang et al., 2021; Wang et al., 2023a).

Recently, inspired by the emergence of Large Language Models (LLMs) that can perform well in many commonsense reasoning tasks (Qin et al., 2023; Bian et al., 2023), Chan et al. (2023) attempted to use LLMs for a CSKB Reasoning bench-


PersonX wanted the award
PersonX is ambitious
PersonX is skillful
?
xWant
xAttr
isAfter
PersonX heard about the competition
oEffect
PersonX prepare for the competition
PersonX's competitors feel insecure
xReact
PersonX is confident
?
PersonX win
PersonX take the opportunity
Yes

Answer whether the following statement is plausible. Answer with only Yes or No:
S: PersonX eat the sub, as a result, PersonX feels PersonX be full
A: Yes
S: PersonX play game, as a result, PersonX feels PersonX forget to do homework
A: No
...
S: PersonX prepare for the competition, as a result, PersonX feels PersonX win
A:

Figure 1: Examples of CSKB Reasoning. Solid arrows represent existing triples (i.e., instances) in CSKBs, while the dashed arrows with question marks represent new knowledge triples which will be determined if they are commonsense. LLMs often fail to acquire the explicit relational constraints in CSKBs, hence making wrong predictions for many new knowledge triples.

mark named CSKB Population (CKBP) (Fang et al., 2023). However, the result shows that LLMs still fall short in the benchmark, even with a large number of in-context examples. One of the problems is that LLMs find it hard to acquire the explicit relational constraints in CSKBs, hence making wrong predictions. In the example in Figure 1, the `xReact`[1] relation in CSKBs requires the tail event of the knowledge triple to express a mental state, such as "*PersonX* is confident", instead of an action, such as "*PersonX* win". Meanwhile, LLMs fail to recognize the constraint from in-context exemplars, thus making the judgment mainly based on the semantics of the head and tail events. It leads to an incorrect prediction that the triple (*PersonX* prepare for the competition, `xReact`, *PersonX* win) is plausible. In light of this, many advanced prompting techniques, such as Chain-of-Thouht (CoT) (Wei et al., 2022), Least-to-Most (Zhou et al., 2023), Active-CoT (Diao et al., 2023), etc., can be possible alternatives for improvements. Nonetheless, they are task-agnostic and suffer from the inherent shortcoming of LLMs in inducing the rules in CSKBs (which we refer as symbolic reasoning ability), as current deep learning still struggles to deal with symbolic and high-level concepts reasoning tasks (Bengio et al., 2021; Huang and Chang, 2023; Pan et al., 2023).

To this end, we propose **ConstraintChecker**, **a plugin component** for LLMs to handle the problem of explicit constraints in CSKB reasoning. ConstraintChecker supports LLMs' reasoning as an independent component in addition to the **main-task component** that determines whether a knowledge triple is commonsense or not. There are two modules in this plugin. Given a knowledge triple *(head event, relation, tail event)*, we first employ a rule-based/symbolic module to produce a list of constraints based on the relation. The list is then passed to a zero-shot learning module, where we construct constraint-checking questions and use the same LLM as in the main-task component in a zero-shot manner to check whether the instance satisfies all constraints. The acquired constraint-checking result is then aggregated with the prediction from the main-task component by logical conjunction to produce the final prediction.

We implement ConstraintChecker and conduct extensive experiments on a CSKB Reasoning benchmark CKBPv2 (Fang et al., 2023) as well as a synthetic discriminative version of ATOMIC$^{20}_{20}$ (in short, SD-ATOMIC$^{20}_{20}$), over two large language models: ChatGPT (gpt-3.5-turbo-0301) and GPT3.5 (text-davinci-003). On both language models, ConstraintChecker improves over prompting techniques (as the main-task component) by a significant margin in different metrics, achieving the best result on both the benchmarks CKBPv2 and SD-ATOMIC$^{20}_{20}$. Further analyses and ablation studies show the effect of each of the considered constraints as well as different choices of prompt design in ConstraintChecker, and the superiority of its plug-and-play design over the single-prompt counterpart.

To summarize, our contribution is **two-fold**: (1) We propose ConstraintChecker, an independent plugin that handles the problem of explicit constraints in CSKB reasoning to improve over main-task prompt methods, and (2) We conduct extensive experiments on two CSKB Reasoning benchmarks CKBPv2 and SD-ATOMIC$^{20}_{20}$, to demonstrate our method's effectiveness and advantages over other advanced prompting techniques.

2 Background and Related Works

2.1 CSKB Reasoning

Commonsense knowledge bases store commonsense knowledge in the format of *(head event, relation, tail event)* triples. Reasoning on CSKBs includes two main settings, a discriminative and a generative one. They are formally defined as: Given $B = \{(h,r,t)|h \in H, r \in R, t \in T\}$ (where $H/R/T$ is the set of head events/relations/tail events) the reference knowledge base, the discriminative setting (Fang et al., 2021b,a, 2023) assigns a binary label $y \in \{0,1\}$ to a new knowledge triple $T = (h,r,t)$ to indicate whether the triple T is commonsense or not; while the generative setting (Bosselut et al., 2019; Hwang et al., 2021) generates new commonsense knowledge triples $T' = (h,r,t')$ based on existing head events h and relations r. Although our method can be adapted to the generative setting, the evaluation of our method is more complex than that in the discriminative setting. Therefore, in this work, we only focus on the discriminative setting.

In terms of benchmarks with the discriminative setting, to our best knowledge, CKBPv2 (Fang et al., 2023) and its predecessor CKBPv1 (Fang et al., 2021a) stand as only comprehensive CSKB

[1]By definition in (Sap et al., 2019), a tail event of the `xReact` relation expresses how the protagonist feels after the corresponding head event.

Reasoning benchmarks, which cover the knowledge on four popular CSKBs (ConceptNet; Speer et al., 2017, ATOMIC; Sap et al., 2019, ATOMIC$^{20}_{20}$; Hwang et al., 2021, and GLUCOSE; Mostafazadeh et al., 2020). Nonetheless, since the reference CSKBs are popular and widely used among the NLP community, we believe CKBPv2 is representative enough in terms of CSKB reasoning. Besides, to ensure the reliability of our result in this work, we synthesize a discriminative-setting dataset from ATOMIC$^{20}_{20}$ which is designated for the generative setting. Indeed, the two benchmark datasets inherit similar head/tail events' formats and the same relation list from ATOMIC$^{20}_{20}$.

Meanwhile, in term of methodology, despite previous efforts to CSKB reasoning, most of them are based on knowledge base embeddings (Li et al., 2016; Malaviya et al., 2020; Hua and Zhang, 2022) or (lightweight) fine-tuning pre-trained language models (Yao et al., 2019; Fang et al., 2021a; Zhang et al., 2023), and less effort has been dedicated to studying how to use LLMs for CSKB reasoning via prompting. We address this research gap by studying a constraint-checking plugin to enhance the performance of LLMs.

2.2 Constraint Modelling in Traditional Knowledge Bases

Integrating rules or constraints into reasoning systems on traditional knowledge bases (KB) and knowledge graphs (KG) has long been studied (Wang et al., 2015; Krompaß et al., 2015; Ding et al., 2018; Zhang et al., 2019; Lan et al., 2023). While Wang et al. (2015) aimed to incorporate rules seamlessly into embedding models for KB completion during inference time by formulating inference as an integer linear programming (ILP) problem, Krompaß et al. (2015) studied the effect of type-constraints on the statistical modeling with latent variable models for large knowledge graphs. In a more recent time, other works such as Ding et al. (2018); Zhang et al. (2019); Lan et al. (2023) attempt to improve KG embedding by modelling rules and constraints in the learning objective. Our work, by contrast, introduces a novel augmentation paradigm which employs an explicit use of constraints in the inference time to improve the performance of large language models on CSKB reasoning.

2.3 Prompting Methods in LLMs

While simple prompt engineering and vanilla in-context learning have already witnessed a remarkable performance in terms of various NLP tasks, there are more sophisticated prompt paradigms to elicit better reasoning capabilities. One representative paradigm is chain-of-thought (CoT) prompting (Wei et al., 2022), which enriches the few-shot examples with explicit reasoning steps towards the final answers, leading to the emergence of many complex reasoning abilities such as arithmetic and commonsense reasoning. Following CoT, other techniques adopt self-consistency (Wang et al., 2023b), least-to-most that break down each question to sub-steps (Zhou et al., 2023), pre-trained verifier to validate the reasoning path (Li et al., 2023), diversity-based method for CoT selection (Zhang et al., 2022), restrict explicit and rigorous deductive reasoning of intermediate CoT reasoning process (Ling et al., 2023), uncertainty-based method for CoT selection and annotation (Diao et al., 2023), and automatic prompt augmentation and selection with CoT (Shum et al., 2023). Our work differs from those CoT-based prompt techniques in that we study add-on constraints to be applied to the result of any prompting technique.

3 ConstraintChecker

An overview of our proposed ConstraintChecker is shown in Figure 2. The CSKB reasoning task we focus on is inherently a binary classification task and the expected outputs are either *plausible* or *implausible*. ConstraintChecker consists of two modules, entitled Module 1 and Module 2. For each instance, Module 1 queries *preset rules* to get all relational constraints corresponding to the instance's relation. Module 2 then constructs questions accordingly to ask whether each constraint is satisfied, and passes these questions to the backbone LLM to get predictions. Together with the prediction from the main-task component, we use the logical conjunction (AND operator) to aggregate the final prediction. In fact, ConstraintChecker only has the effect on instances that are predicted as commonsense (or "Yes", corresponding to *plausible*) by the main-task component, and can only change the prediction from "Yes" to "No", in view of the nature of logical conjunction. Thus, it targets and corrects False-Positive predictions.

In this section, we elaborate on how we select the pool of constraints and the preset rules to map

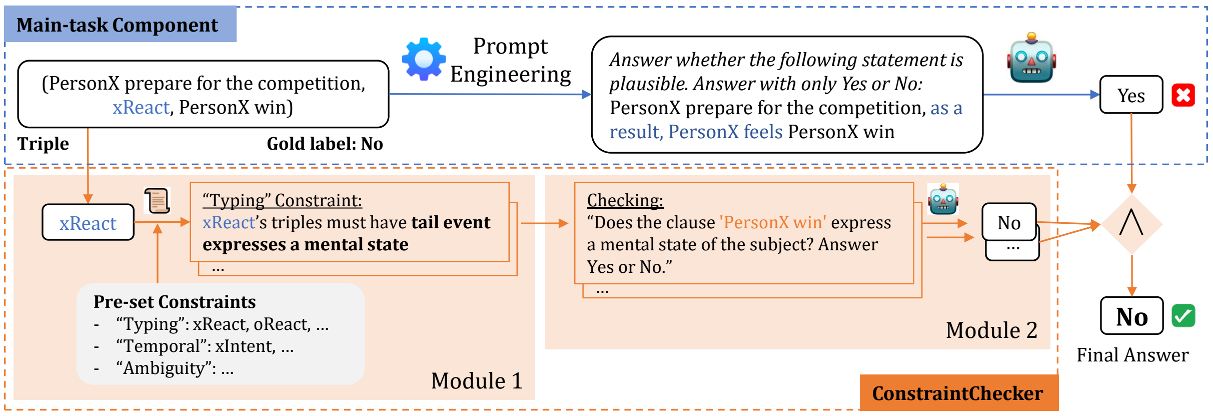

Main-task Component
(PersonX prepare for the competition, xReact, PersonX win)
Triple
Gold label: No
Prompt Engineering
Answer whether the following statement is plausible. Answer with only Yes or No: PersonX prepare for the competition, as a result, PersonX feels PersonX win
Yes
xReact
"Typing" Constraint: xReact's triples must have tail event expresses a mental state
...
Pre-set Constraints
- "Typing": xReact, oReact, ...
- "Temporal": xIntent, ...
- "Ambiguity": ...
Module 1
Checking: "Does the clause 'PersonX win' express a mental state of the subject? Answer Yes or No."
...
No
...
Module 2
∧
No
Final Answer
ConstraintChecker

Figure 2: Illustration of ConstraintChecker. For each instance, Module 1 queries *preset rules* to get all relational constraints corresponding to the instance's relation. Module 2 then constructs questions accordingly to ask whether each constraint is satisfied, and passes these questions to the backbone LLM to get predictions. Together with the prediction from the main-task component, we use the logical conjunction (i.e., ∧ or AND operator) to aggregate the final prediction. Note that "Prompt Engineering" refers to baseline prompting methods (subsection 4.2).

relations to constraints in Module 1, as well as the constraint-checking prompt design in Module 2 concerning the two benchmarks.

3.1 Constraints Selection

We follow the definitions of CSKB relations in previous works, including the taxonomy of *if-then* reasoning types in Sap et al. (2019) and human-readable templates for crowdsourced evaluation in Hwang et al. (2021)[2], to derive the set of considered constraints and the rule to apply constraints. For example, the readable template "as a result, *PersonX* feels" of the `xReact` relation suggests the "temporal" constraint, in which the head event must happen before the tail event, and the taxonomy of `xReact` implies the "typing" constraint, in which the tail event must express a mental state. Note that the template "as a result, *PersonX* feels" of `xReact` may not strictly impose the typing constraint on the tail event, due to a subtle problem in natural language. For example, for human, two text sequences "as a result, *PersonX* feels *PersonX* will win" and "as a result, *PersonX* feels *PersonX* is confident" all make sense, although "*PersonX* will win" completely does not express a mental state of *PersonX*.

In addition, as Davis (2023) suggests that many commonsense datasets have significant portions of ambiguous instances, we also consider the "ambiguity" constraint.

Among of possible constraints, we shortlisted the most likely needed constraints, namely typing, temporal, and ambiguity constraints. The formal definition of each constraint is as follows:

- **Typing:** The tail event has to express the type of content (one among three types: activity, mental state, persona) that the relation expects. For example, `xReact`'s tail events need to express a mental state, while `xAttr`'s tail events need to express a persona.
- **Temporal:** The (estimated) temporal order (i.e. before or after) of the head event and the tail event must follow the order derived from the definition/human-readable template of the relation. For example, for `HinderedBy` relation, the head event must happen after the tail event.
- **Ambiguity:** The meaning of the head and tail events must be grammatically complete and semantically informative. For example, "*PersonX* order a salad" is not ambiguous, while "*PersonX* would like" is ambiguous.

3.2 Preset Rules

Each relation will be mapped into a set of constraints based on the aforementioned taxonomy and templates, as well as human-readable templates used by the main-task component in terms of how well the template of the relation semantically reflects the constraints of that relation. For example, the template of `xReact` "as a result, *PersonX* feels" does contain the phrase "as a result" representing the temporal constraint which is needed to check. To refine the rule, we conduct a pilot study with ChatGPT on the development set of CKBPv2 to estimate the effectiveness of designated constraints on each relation. According to the results of the pilot study (Appendix A.2), we remove ineffective

[2]The templates are shown in Table 9 and 10 in the Appendix.

Constraint	Relations on effect (pre-pilot-study)	Relations on effect (post-pilot-study)
Typing	`xReact (m), oReact (m), xAttr (p)`	`xReact (m), oReact (m), xAttr (p)`
Temporal	`xIntent (a), xNeed (a), Causes (b), HinderedBy (a)`	`xIntent (a), xNeed (a)`
Ambiguity	All relations	No relation

Table 1: Relations on range of effect of each constraint. For Typing constraint, the notation `(m)` and `(p)` denote the required type "mental state" and "person" of the tail event. For Temporal constraint, the notation `(a)` and `(b)` denote the required estimated order "after" and "before" of the head and tail events.

constraint-relation pairs to refine the rule, as shown in Table 1. In fact, the ineffectiveness of Ambiguity constraint suggests that *randomly taking a constraint and then cherry-picking its effective constraint-relation pairs would not work*, instead we need to derive the rules from prior knowledge about relations. We further conduct an ablation study on the main experiments w.r.t. ChatGPT on CKBPv2 to show the ineffectiveness of removed relation-constraint pairs (Section 4.4).

3.3 Constraint-Checking Prompt Design

As we use a zero-shot LLM to check constraints, we construct questions for derived constraints in a direct question-answering manner. For example, for the typing constraint, which requires the tail event of the triple to express a mental state, we design a prompt as "Does the clause <*tail event*> express a mental state of the subject? Answer Yes or No". Thanks to the robustness of LLMs and the fact that constraint satisfaction is a relatively simple task that does not require complex reasoning, exemplars for constraint-checking questions are not needed. For each constraint, we design two templates to seek the best one. We provide an analysis of different prompt choices later in Section 4.4. Overall, we choose the following prompt designs for typing and temporal constraints respectively.
Typing: "Which aspect (among three options 1. event/activity, 2. persona, 3. mental state) of the subject does the clause <*tail event*> express. Answer the choice only."
Temporal: "Which one of the following two statements is more plausible: 0. <*tail event*> before <*head event*>, 1. <*tail event*> after <*head event*>. Answer 0 or 1 only."

Since the chosen prompts do not standardly question whether the constraint is satisfied, we use a snippet of code to convert the acquired prediction into the Yes/No answer for the standard constraint-checking question.

4 Experiments

4.1 Benchmarks

We use CKBPv2 (Fang et al., 2023) and SD-ATOMIC$^{20}_{20}$ as the CSKB reasoning benchmarks for evaluation. For CKBPv2, to reduce the computational cost while keeping the same data distribution w.r.t two attributes *relation* and *label*, we use stratified sampling to down-scale the test set, hence forming a set of 979 instances. Meanwhile, SD-ATOMIC$^{20}_{20}$ is curated from the test set of ATOMIC$^{20}_{20}$ using random relation/tail event replacement as the negative sampling strategy, resulting in a set of 1000 instances with the equal numbers of positive and negative instances. More details about the two benchmarks can be found in Appendix A.1.

4.2 Setups

The following prompting methods serve as the main-task component's baselines in our experiments.

- Zero-shot: standard zero-shot prompting, which directly asks the task question and "forces" LLMs to only return the final answer.
- Few-shot: standard prompting which uses exemplars to facilitate in-context learning. Here, we consider three variants which are different in the way they select exemplars. In detail, we use (1) Random exemplar selection. (2) KATE (Liu et al., 2022), which chooses exemplars that are the most semantically similar to the test instance using sentence embedding models. (3) KATE-s, a special version of Few-shot KATE for CSKB reasoning, in which the selected exemplars must have the same relation as the test instance.
- Zero-shot-CoT (Kojima et al., 2022): the zero-shot prompting technique which uses the phrase "Let's think step by step" to stimulate LLMs to generate rationales before the final answer.
- Few-shot-CoT: chain-of-thought methods which use exemplars. Here, we employ a simple method that randomly selects CoT exemplars.

Method	CKBPv2				SD-ATOMIC$^{20}_{20}$			
	ChatGPT		GPT3.5		ChatGPT		GPT3.5	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Zero-shot	67.58	47.48	73.92	37.16	59.6	61.3	59.7	62.72
+ ConstraintChecker	69.19	48.45 (+0.97)	74.97	38.09 (+0.93)	63.4	63.62 (+2.32)	62.2	64.2 (+1.48)
Few-shot (Random)	69.94	48.84	72.12	49.76	59.4	60.43	57.4	61.13
+ ConstraintChecker	71.67	50.15 (+1.31)	73.82	**51.07** (+1.31)	62.0	62.0 (+1.57)	60.2	62.73 (+1.6)
Few-shot (KATE)	68.0	44.94	69.87	47.41	61.3	60.55	58.5	61.4
+ ConstraintChecker	69.73	46.01 (+1.07)	71.81	48.89 (+1.48)	63.8	62.13 (+1.58)	61.7	63.28 (+1.88)
Few-shot (KATE-s)	67.35	45.99	69.25	47.52	59.0	59.08	60.3	62.51
+ ConstraintChecker	69.19	46.95 (+0.96)	71.09	48.68 (+1.16)	61.7	60.72 (+1.64)	63.0	64.15 (+1.64)
Zero-shot-CoT	62.14	42.37	77.25	39.49	49.9	50.25	58.4	57.89
+ ConstraintChecker	64.35	43.58 (+1.21)	77.93	40.21 (+0.72)	52.7	51.69 (+1.44)	60.8	59.34 (+1.45)
Few-shot-CoT	76.92	48.41	62.0	45.71	60.1	61.52	57.0	62.74
+ ConstraintChecker	77.46	48.67 (+0.26)	63.84	46.74 (+1.03)	62.2	62.8 (+1.28)	60.1	**64.47** (+1.73)

Table 2: Main experimental results on the test data of CKBPv2 and SD-ATOMIC$^{20}_{20}$. F1 score is the main metric.

Our experiments are based on two large language models ChatGPT (gpt-3.5-turbo-0301) and GPT3.5 (text-davinci-003), as they are available, stable, and two of the most capable models at the time we were conducting our experiments. We set temperature $T = 0$ for all experiments. For KATE strategy, we use the best model reported in Liu et al. (2022).[3] For baselines with exemplars, all exemplars are selected from the training set provided in Fang et al. (2023) regardless of labels, and the number of exemplars used in each prompt is 5 by default.

We design three prompt templates which are used to convert knowledge triples (both tested instances and exemplars) into the free-text format, then input the text to LLMs. The result of each baseline will be averaged from the results of three different prompt designs. We leave details about prompt designs in Appendix B.2. Nonetheless, we provide one example of how we use a prompt template to convert a knowledge triple to the free-text format as follow:

Triple: (*PersonX* prepare for the competition, `xReact`, *PersonX* win)

Template: Answer whether the following statement is plausible. Answer with only Yes or No: <*head event*>, <*human-readable template of relation*> <*tail event*>.

$\rightarrow$ **Input Prompt**: Answer whether the following statement is plausible. Answer with only Yes or No: *PersonX* prepare for the competition, as a result, *PersonX* feels *PersonX* win.

[3]We use the checkpoint sentence-transformers/roberta-large-nli-stsb-mean-tokens via Huggingface Transformers.

4.3 Results and Analysis

The experimental results are shown in Table 2. We report accuracy w.r.t all instances and (binary) F1 score w.r.t. the positive class of all baselines and our method. Following Fang et al. (2023), we choose F1 as the main metric. In the columns corresponding to F1 score, we add numbers in scriptsize to indicate the performance gain of ConstraintChecker over prompting methods.

Overall, our method consistently improves[4] over all prompting methods and backbone LLMs for both benchmarks, by an average margin of 0.96%/1.11% and 1.64%/1.63% in F1 score with respect to backbone ChatGPT/GPT3.5 on CKBPv2 and SD-ATOMIC$^{20}_{20}$ respectively. Similar performance gain can be also observed in groups of non-CoT, CoT, zero-shot, few-shot baselines. Furthermore, we achieve the best result on the CKBPv2 benchmark with ConstraintChecker paired with Few-shot (Random), and also achieve the best result on the synthetic benchmark SD-ATOMIC$^{20}_{20}$ with ConstraintChecker paired with Few-shot-CoT. We note that on CKBPv2, our result is not directly comparable to results from previous works (Fang et al., 2023; Chan et al., 2023), as the scale of the evaluation set, the number of exemplars used in few-shot prompting methods, and the version of LLMs of our and previous works are different.

We further analyze[5] results on CKBPv2 in-depth

[4]Statistically significant under one-tailed t-test with confidence level 99%.

[5]Since CKBPv2 is human-curated, the analysis on the benchmark is more objective and reliable than that on our synthetic SD-ATOMIC$^{20}_{20}$.

to point out the source of improvement of ConstraintChecker and to compare the improvement brought by our method and by main-task prompting techniques.

Where does the improvement of ConstraintChecker come from? We take relations `xReact`, `oReact`, `xAttr` with the Typing constraint as an example to show the effect of ConstraintChecker on GPT3.5. Recall that when ConstraintChecker is applied, the final prediction will be the logical conjunction of predictions from the zero-shot baseline and our ConstraintChecker. Thus, triples that our method has an effect on are those with positive predictions from the zero-shot baseline and negative predictions from ConstraintChecker. As our method aims to correct False Positive (FP) predictions (and not to hurt True Positive predictions), we examine **among concerned triples**, how many cases ConstraintChecker:

1. correctly judge a triple as it violates the constraint, and because the triple's gold label has to be 0 (except incorrect human annotation), thus helps to turn the FP prediction of the baseline to True Negative,
2. incorrectly judge a triple as it violates the constraint (in fact it does not), however, because the triple's gold label is 0, the misjudgment accidentally helps to turn the FP prediction to True Negative,
3. incorrectly judge a triple as it violates the constraint (in fact it does not), and because the triple's gold label is 1, the misjudgment undesirably turns the True Positive prediction of the baseline to False Negative.

In fact, ConstraintChecker is designated for the first category. Therefore, the more the first category happens in comparison to the second and third categories, the more reliable the improvement of ConstraintChecker is.

We ask four external voluntary graduate NLP researchers who have at least one year of experience working on CSKBs to annotate the typing constraint status (i.e. "satisfied" or "not satisfied") of those considered triples. The Fleiss' Kappa score of this annotation is 0.2381, and the final label is the majority vote among four annotators. From relevant annotations and predictions, we calculate the percentage of cases falling into each mentioned category, and find that 93% of the concerned triples fall into the first category. Similarly, when considering other relations and other baselines, we observe the majority of the cases fall into the first category. That shows the valid source of improvement of ConstraintChecker.

Comparison of ConstraintChecker and other prompt engineering techniques We also compare the average effectiveness of our method with other types of prompt engineering, including 1. the use of exemplar, 2. exemplar selection, and 3. chain-of-thought. We estimate the effectiveness (i.e. net average gain) of each prompt engineering type as the average difference of F1 score between two groups of corresponding baselines with or without the appliance of such a type. While exemplar selection and chain-of-thought do not bring any certain benefit, the usage of exemplars brings a remarkable improvement. In fact, exemplars help to hugely increase the recall of zero-shot baselines with GPT3.5 backbone (from 36.06% to 66.55% on average). However, it is deemed to the strictness of GPT3.5 in judging if a knowledge triple is commonsense, as its zero-shot baselines have a much lower recall comparing to other baselines. Meanwhile, the improvement of ConstraintChecker is consistent over all baselines as it helps to correct False-Positive predictions. Indeed, it improves the precision of baselines and does not significantly hurt the recall. Also, thanks to the simple prompt design, our method is more efficient than the use of exemplars and CoT (Table 8).

4.4 Ablation Study

We further conduct several additional experiments with ChatGPT on CKBPv2 to show the importance of our preset rules, constraint-checking prompt choices, and ConstraintChecker's role as a separate module from the main-task component.

We report the result of the zero-shot baseline with each constraint and each prompt design applied in Table 3. We focus on F1 score of test triples of 5 relations `xReact`, `oReact`, `xAttr`, `xIntent`, `xNeed` in which ConstraintChecker has effects on according to the final preset rules, as well as of 4 extra relations `xWant`, `xEffect`, `HinderedBy`, `Causes`. We show F1 score of these relations because `xWant` and `xEffect` account for a large portion of the test set, while `HinderedBy` and `Cause` were set to be checked with temporal constraint before the pilot study.

Effect of Preset Rules In previous analyses, we show where and how ConstraintChecker improves

Constraint (Prompt Design)	xReact	oReact	xAttr	xIntent	xNeed	xWant	xEffect	HinderedBy	Causes
N/A (Zero-shot baseline)	52.75	41.76	49.89	37.79	46.11	51.76	52.61	34.67	44.18
Typing (selected P.D.)	54.38	41.76	51.66	-	-	-	-	-	-
Typing (alternative P.D.)	59.37	28.33	52.77	-	-	-	-	-	-
Temporal (selected P.D.)	45.24	54.27	49.89	44.88	48.48	30.41	42.97	27.76	31.19
Temporal (alternative P.D.)	-	-	-	26.13	17.27	-	-	-	-
Ambiguity (P.D. 1)	31.15	41.27	27.01	13.65	2.9	41.24	24.4	3.33	14.85
Ambiguity (P.D. 2)	38.67	42.18	34.48	30.73	5.34	48.52	50.53	3.33	31.29
Constraint-L2M (w/o exemplars)	45.35	43.16	43.43	23.92	24.26	-	-	-	-
Constraint-L2M (w/ 3 exemplars)	52.16	38.72	31.5	13.49	42.9	-	-	-	-

Table 3: Ablation study on each constraint and prompt choice. P.D. in the setting names abbreviates "prompt design". Results in this table are F1 scores w.r.t. triples of each relation. The notation "-" indicates no change in comparing to the result of the zero-shot baseline, because we do not consider those constraint-relation pairs in the preset rules (either pre- or post-pilot-study) w.r.t. the setting.

the results of other main-task prompting methods. However, it does not mean both typing and temporal constraints are necessary. As observed in two rows, Typing (selected prompt design, P.D. for short) and Temporal (selected P.D.), of Table 3, each constraint boosts the performance on each relation that they affect on according to the post-pilot-study preset rules. That demonstrates the importance of each selected constraint.

Similarly, we study the result regarding constraint-relation pairs which are never in or removed from the preset rules after our pilot study. As shown in Table 3, for Temporal (selected P.D.) constraint, F1 score on `HinderedBy` and `Causes` are lower than the counterparts in the zero-shot baseline. Also, for other relations which are never in the preset rules of the temporal constraint, such as `xWant` and `xEffect`, the constraint often hurts the performance. Apart from that, the Ambiguity constraint (in both prompt designs which are shown in Appendix B.2) also hurt the performance of all relations. We argue that ambiguity is a subjective concept, thus within the simple design philosophy of ConstraintChecker, we may not find the best way to use the constraint. Overall, the result regarding unconsidered constraint-relation pairs is consistent with our observation in the pilot study.

Effect of the Prompt Design We also ablate prompt designs of typing and temporal constraints to study the effect of constraint question design on the performance on triples of each relation. In Table 3, Typing (alternative P.D.) and Temporal (alternative P.D.) indicate the result w.r.t the alternative prompt design for typing and temporal constraints respectively. In fact, the alternative prompt designs (shown in Appendix B.2), are formulated in a more direct way that asks if the constraint is satisfied/violated, while our selected prompt design asks more general multiple-choice questions. It can be seen that the alternative for typing constraint gives higher scores for `xReact` and `xAttr` but much lower scores for `oReact`. Also, the alternative prompt of temporal is hugely worse than the selected prompt design. We argue that the reason could be the advantage of having more context and references when asking general multiple-choice questions than when focusing on a specific case. That demonstrates the sensitivity of our method on constraint prompt design.

Effect of ConstraintChecker as an Independent Component ConstraintChecker is used in a plug-and-play manner, where we can get predictions independently from the main-task component. In this part, we study an alternative single-prompt design choice that models ConstraintChecker as an end-to-end CoT-like pipeline to directly add to the main-task prompt. This serves as additional experiments to demonstrate the advantage of our plug-and-play design as opposed to the single-prompt counterpart.

Inspired by Least-to-Most (L2M) (Zhou et al., 2023) which first decomposes a complex problem into a list of easier subproblems, and then sequentially solving these subproblems in different passes to LLM to reach the final answer, in this ablation, we treat constraints as easier subproblems and the main question as the hardest question which is asked ultimately. The CoT will immediately stop and conclude that a triple is not commonsense if the triple does not satisfy the constraint. We name the alternative method as Constraint-L2M for simplicity. The prompt design of this baseline is shown in Appendix B.2.

On the one hand, results in Table 3 show that even with exemplars, Constraint-L2M hurts the performance on all considered relations, in contrast to ConstraintChecker. Taking a closer look into the output of Constraint-L2M (w/o exemplars) w.r.t backbone GPT3.5 and the typing constraint, we observe that 14% of the logic step (to check if the determined tail event's type matches with the desired type, virtually equivalent to string matching) is incorrect. Meanwhile, other combinations, though having a small error rate in the logic step, unexpectedly perform worse than the zero-shot counterparts in the main-task step. We argue that the phenomenon possibly results from the influence of patterns (Turpin et al., 2023) in constraint-checking steps' output. That shows the advantages offered by our method's separateness from main-task prompting methods, as fusing the constraint checking step to the main-task prompt can even increase the error rate due to failures in the symbolic reasoning step or its interference on the main-task step.

On the other hand, in term of efficiency, Constraint-L2M even used without exemplars poses a much higher cost than our method (Table 8). Thus, the plug-and-play design of our method is preferable over the single-prompt design in both aspects - effectiveness and cost.

5 Conclusion

In this paper, we proposed ConstraintChecker, a constraint-wise plugin, to help LLMs and prompting methods to cope with the problem of explicit relational constraint in CSKB reasoning. Experimental results show that ConstraintChecker consistently improves over any main-task prompting technique by a significant margin, achieving SoTA performance on the benchmark.

Acknowledgement

The authors of this paper were supported by the NSFC Fund (U20B2053) from the NSFC of China, the RIF (R6020-19 and R6021-20) and the GRF (16211520 and 16205322) from RGC of Hong Kong, and the UGC Research Matching Grants (RMGS20EG01-D, RMGS20CR11, RMGS20CR12, RMGS20EG19, RMGS20EG21, RMGS23CR05, RMGS23EG08). We also thank Samuel Cahyawijaya and Bryan Wilie from HKUST for their helpful comments.

Limitations

This paper works on a task-specific method as well as evaluates it on two CSKB reasoning benchmarks. Further study of the proposed method on other reasoning tasks is expected to examine its broader generalizability. Also, the process of building the first module of our method is complex and requires a certain understanding of the task and the benchmarks. How to systematically adapt this module to other tasks (e.g. automatically inducing constraints from questions), such as CSKB reasoning in the generative setting or commonsense question answering, remains to be studied. Last but not least, as the number of prompt designs, especially the set of human-readable templates, is limited, it is not 100% guaranteed that our method will be effective for other designs of prompt of this CSKB reasoning or other reasoning tasks in general. More experiments are needed to make our claim more certain.

Ethical Considerations

This work aims to improve the performance of LLMs on a commonsense reasoning task, which - in the case of this work - involves the use of ChatGPT (gpt-3.5-turbo-0301) and GPT3.5 (text-davinci-003). Therefore, the same risks from LLMs research are also applicable to this work (Bender et al., 2021).

In term of computational cost, our work does not involve any training or finetuning of (large) language models. From our rough calculation which is partially shown in Table 8, the expense to conduct all experiments (including both preliminary and main experiments) in this project is around US$200.

Last but not least, this paper works on CKBPv2 and ATOMIC$^{20}_{20}$, open-source benchmarks for the research community to study the CSKB reasoning problem under an MIT and CC-BY license respectively. This work shares the same ethical issues as the work of CKBPv2 and ATOMIC$^{20}_{20}$. Data is anonymized, thus our work does not propagate any privacy problems about any specific entities. Also, we carried out human expert annotation for analysis purposes. Since the amount of work is small, we and the annotators agree to consider it as a voluntary service.

References

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. 2021. Deep learning for AI. *Commun. ACM*, 64(7):58–65.

Ning Bian, Xianpei Han, Le Sun, Hongyu Lin, Yaojie Lu, and Ben He. 2023. Chatgpt is a knowledgeable but inexperienced solver: An investigation of commonsense problem in large language models. *arXiv preprint arXiv:2303.16421*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Chunkit Chan, Jiayang Cheng, Weiqi Wang, Yuxin Jiang, Tianqing Fang, Xin Liu, and Yangqiu Song. 2023. Chatgpt evaluation on sentence level relations: A focus on temporal, causal, and discourse relations. *ArXiv*, abs/2304.14827.

Ernest Davis. 2023. Benchmarks for automated commonsense reasoning: A survey. *ArXiv*, abs/2302.04752.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 1173–1178.

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models.

Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving knowledge graph embedding using simple constraints. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 110–121, Melbourne, Australia. Association for Computational Linguistics.

Tianqing Fang, Quyet V. Do, Sehyun Choi, Weiqi Wang, and Yangqiu Song. 2023. Ckbp v2: An expert-annotated evaluation set for commonsense knowledge base population. *ArXiv*, abs/2304.10392.

Tianqing Fang, Weiqi Wang, Sehyun Choi, Shibo Hao, Hongming Zhang, Yangqiu Song, and Bin He. 2021a. Benchmarking commonsense knowledge base population with an effective evaluation dataset. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 8949–8964. Association for Computational Linguistics.

Tianqing Fang, Hongming Zhang, Weiqi Wang, Yangqiu Song, and Bin He. 2021b. DISCOS: bridging the gap between discourse knowledge and commonsense knowledge. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 2648–2659. ACM / IW3C2.

Wenyue Hua and Yongfeng Zhang. 2022. System 1 + system 2 = better world: Neural-symbolic chain of logic reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 601–612, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6384–6392. AAAI Press.

Filip Ilievski, Jay Pujara, and Hanzhi Zhang. 2021. Story generation with commonsense knowledge graphs and axioms. In *Workshop on Commonsense Reasoning and Knowledge Bases*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.

Denis Krompaß, Stephan Baier, and Volker Tresp. 2015. Type-constrained representation learning in knowledge graphs. In *The Semantic Web - ISWC 2015*, pages 640–655, Cham. Springer International Publishing.

Yinyu Lan, Shizhu He, Kang Liu, and Jun Zhao. 2023. Knowledge reasoning via jointly modeling knowledge graphs and soft rules.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making large language models better reasoners with step-aware verifier.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2925–2933. AAAI Press.

Nasrin Mostafazadeh, Aditya Kalyanpur, Lori Moon, David W. Buchanan, Lauren Berkowitz, Or Biran, and Jennifer Chu-Carroll. 2020. GLUCOSE: generalized and contextualized story explanations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4569–4586. Association for Computational Linguistics.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3027–3035. AAAI Press.

KaShun Shum, Shizhe Diao, and Tong Zhang. 2023. Automatic prompt augmentation and selection with chain-of-thought from labeled data.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451. AAAI Press.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *International Joint Conference on Artificial Intelligence*.

Weiqi Wang, Tianqing Fang, Baixuan Xu, Chun Yi Louis Bo, Yangqiu Song, and Lei Chen. 2023a. CAT: A contextualized conceptualization and instantiation framework for commonsense reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13111–13140, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Changlong Yu, Hongming Zhang, Yangqiu Song, and Wilfred Ng. 2022. Cocolm: Complex commonsense enhanced language model with discourse relations. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1175–1187.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Iteratively learning embeddings and rules for knowledge graph reasoning. In *WWW*, pages 2366–2377. ACM.

Yichi Zhang, Zhuo Chen, Wen Zhang, and Huajun Chen. 2023. Making large language models perform better in knowledge graph completion. *arXiv preprint arXiv:2310.06671*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Pei Zhou, Karthik Gopalakrishnan, Behnam Hedayatnia, Seokhwan Kim, Jay Pujara, Xiang Ren, Yang Liu, and Dilek Hakkani-Tur. 2021. Commonsense-focused dialogues for response generation: An empirical study. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 121–132.

A Experiments

In this section, we provide details of how to form test benchmarks, the pilot study, additional analyses of results, and baselines.

A.1 Benchmarks

CKBPv2 originally consists of approximately 1k development instances and 4k test instances. To reduce the computational cost while keeping the same data distribution, we use stratified sampling to down-scale the test split of the benchmark by a factor of 4, hence forming a test set of 979 instances. The down-sampled test set includes 208 instances with label 1 (which means they are commonsense or "positive"), thus, the ratio of the number of commonsense/not commonsense instances remains approximately 1/4. In fact, results of the human-performance baseline and supervised-learning baseline in Table 4 suggest that the down-scaled test set is representative of the whole test set.

Meanwhile, SD-ATOMIC$_{20}^{20}$ is curated from the test set of ATOMIC$_{20}^{20}$ as follow. First, we randomly select 1000 distinct head events, and then for each head event, we select one triple (i.e. instance). Since our final preset rules concern 5 relations `xReact, oReact, xAttr, xIntent, xNeed`, the selected triples are also of these 5 relations only. Finally, we apply a data-manipulation method on the collection of 1000 triples to sample negative instances for SD-ATOMIC$_{20}^{20}$, in which we randomly select (1) 250 triples and change the relation, and (2) 250 triples (h,r,t) and change the tail event so that obtained triples do not exist in the ATOMIC$_{20}^{20}$'s test set. By assumption, these 500 instances are not commonsense and then assigned the label 0, while the rest 500 instances which remain intact are commonsense and assigned the label 1.

A.2 Pilot Study

We sampled 102 instances from the dev split of CKBPv2 in a relation-wise stratified manner to form a small dataset for the pilot study. The prompt design used in this pilot study consists of zero-shot template design 3 (Table 13), constraint template design 1 for Typing and Temporal and template design 2 for Ambiguity (Table 12). The result is shown in Table 5, with a similar organization as Table 3.

We observed no effect of the Ambiguity constraint, thus we dropped that constraint. Furthermore, as we observed no effect of the Temporal

Method	Acc.	Pre.	Rec.	F1
Random ($p = 0.5$)	50.00	20.00	50.00	28.57
Human *(full set)*	-	-	-	91.50
Prior best *(full set)*	-	-	-	46.63
Human	96.38	94.37	88.22	91.17
Prior best	60.54	32.08	76.76	45.25

Table 4: Random, Human, and previous best Supervised Learning baselines' performance as a lower bound, upper bound, and a competitive baseline to compare with LLM prompting methods. Here, Acc., Pre., Rec. respectively mean Accuracy, Precision, and Recall. The random baseline follows the Bernoulli distribution with probability p is 0.5. Results of baselines with suffix *(full set)* are results on the whole test set of CKBPv2 (Fang et al., 2023). For the last two rows, we use the available human annotation of CKBPv2 and rebuild the best baseline based on its description in Fang et al. (2023) to compute these statistics. Results in this table suggest that the down-scaled test set is representative of the whole test set.

constraint on samples of relations `HinderedBy` and `Causes`, we decided to remove these constraint-relation pairs. Nonetheless, while there is no effect of the Typing constraint on samples of relations `oReact` and `xAttr`, we still kept these constraint-relation pairs because the readable templates of the two relations do not adequately reflect their Typing constraint.

A.3 Analysis

As observed in Table 2, two groups of non-CoT and CoT baselines have the results w.r.t ChatGPT and GPT3.5 showing different patterns. While non-CoT baselines with backbone ChatGPT do not benefit much from or even suffer from a performance decrease due to exemplars, the non-CoT baselines with GPT3.5 and all CoT baselines hugely benefit from in-context exemplars. In CKBPv2, baselines with in-context exemplars are 6% to 10% better than their corresponding zero-shot counterparts.

For CKBPv2, we try two exemplar optimization methods - Active-CoT (Diao et al., 2023) and Automate-CoT (Shum et al., 2023) which respectively use uncertainty-based active learning and rational chains optimization for exemplar selection (Table 6). We notice that exemplar optimization becomes more important for CoT baselines, as the optimization gives a significant gain on ChatGPT and a large improvement on GPT3.5. Also, CoT baselines generally achieve higher score than non-CoT baselines, which is often observed in other benchmarks.

We further explore the dependence of overall baseline performance on our 3 seed prompt designs. The average precision, recall, F1 score over all baselines w.r.t to each prompt design is reported in Table 7. There is a gap between the third seed prompt design and other two seed prompt designs, however, the gap is not significantly large. Therefore, we conclude that there is no significant dependence of baseline performance on seed prompt designs.

Last but not least, we examine to what extent LLMs fail to handle the explicit constraints. We focus on a specific context, which considers the prediction of Few-shot-CoT baseline (with ChatGPT backbone and the third prompt design) and the `xReact` relation. As the Few-shot-CoT baseline works on the main-task question of whether a triple is commonsense, its prediction is not equivalent to the prediction of whether the triple satisfies the constraint. Only its "Yes" prediction implies a "Yes" prediction of constraint satisfaction. Thus, we estimate the failure rate of the Few-shot-CoT baseline based on triples with positive predictions. Among those triples, 43% of the triples do not satisfy the typing constraint, but the baseline implicitly predicts them as satisfied. That supports our claim that LLMs and advanced prompting techniques become less effective in handling explicit constraints in CSKB reasoning.

A.4 Cost Estimation

In Table 8, we estimate the total number of words processed for each instance in each baseline as well as the overhead cost of using additional prompting engineering techniques. We ignore the cost of exemplar optimization, as the process is done at most once per baseline and independent of the size of the test set. As such, here we treat Few-shot (KATE(-s)) the same as Few-shot (Random), and treat Active-CoT and Automate-CoT the same as Few-shot-CoT. Also, since ConstraintChecker's constraint design only involves the head and tail events of test triples, which are irrelevant to seed prompt designs for the main-task component, we only need to run ConstraintChecker once then apply for all baselines and for all seed prompt designs. Overall, it shows the efficiency of ConstraintChecker over other types of prompt engineering.

Constraint	All	xReact	oReact	xAttr	xIntent	xNeed	xWant	xEffect	HinderedBy	Causes
N/A (Baseline)	61.29	66.67	100.00	75.00	50.00	50.00	66.67	66.67	100.00	0.00
Typing	72.73	62.30	100.00	75.00	-	-	-	-	-	-
Temporal	64.41	-	-	-	100.00	57.14	-	-	100.00	0.00
Ambiguity	29.27	25.00	0.00	57.14	66.67	0.00	57.14	0.00	0.00	0.00

Table 5: Result in the pilot study. The result is presented in a similar organization as Table 3

Method	ChatGPT		GPT3.5	
	Acc	F1	Acc	F1
Active-CoT	74.67	49.29	72.28	50.39
Automate-CoT	76.0	50.82	76.68	50.31

Table 6: Results of extra exemplar-optimization baselines on CKBPv2.

Prompt design	1	2	3
Precision	39.85	39.76	39.63
Recall	59.95	59.31	60.37
F1 score	46.38	46.32	47.15

Table 7: Average Precision, Recall, and F1 score over all baselines of each seed prompt design. There is no significant dependence of baseline performance on seed prompt designs.

A.5 Why is ConstraintChecker not extended to intervene False Negatives?

Intervening on False Negatives is equivalent to using constraints satisfaction to "convince" the main-task component that a triple is correct. However, constraint satisfaction is not adequate to justify if the triple is correct, as we also need to consider its overall semantic.

A real example from the development split of CKBPv2 is (*PersonX* go to sleep on hollow, `xReact`, *PersonX* be tired). Clearly, "*PersonX* be tired" expresses a mental state, which means the triple satisfies the typing constraint corresponding to `xReact` that the tail event has to express a mental state. However, the phrase "sleep on hollow" is ambiguous, and even if we ignore the words "on hollow", it's unlikely that "*PersonX* be tired" is a result of "*PersonX* go to sleep". That means the triple is not common sense.

B Supplementary Materials

B.1 Taxonomy of CSKB relations

The taxonomy of CSKB relations are aggregrated from prior works (Sap et al., 2019; Hwang et al., 2021) and demonstrated in Table 9.

Method	Words
Representative baselines	
Zero-shot	28
Few-shot	120
Zero-shot-CoT	68
Few-shot-CoT	321
Type of prompt engineering	
Using exemplars	172
Using CoT	120
ConstraintChecker	72
Constraint-L2M (w/o exemplar)	254

Table 8: Per-instance estimated costs for baselines and additional costs for each type of prompt engineering, including "Using exemplars", "Using CoT", and ConstraintChecker. "Words" indicates the average number of words in the input prompt and generated which are both charged by OpenAI, which are proportional to the actual costs. For types of prompt engineering other than ConstraintChecker, we take the average gap between two groups of baselines w.r.t the type. Here, we generously assume that all constraints of ConstraintChecker are run for every instance, instead of only instances of concerned relations as following the preset rules.

B.2 Prompt design

For each triple (head event, relation, tail event), we convert the triple into a free-text format (so-called *assertion*) using human-readable templates. Along with the original set of templates in Hwang et al. (2021), we also design and experiment with another set of templates to study the correlation between human-readable template design and the result. Likewise, we take the direct question-answering prompt (so-called *main question*) design from Fang et al. (2023) and self-curate another one. The two sets of human-readable templates and two main question designs are shown in the following tables.

An input prompt to LLMs consists of two main parts, the main question and the assertion. We select three combinations of human-readable templates of relations and main question designs as *seed prompt designs*, from which each baseline will adapt to get its three prompt designs (if necessary). The result of each baseline will be averaged

Type	Relations
ATOMIC (Sap et al., 2019)	

Event
Mental state
Persona
	`xNeed, xEffect, xWant, oEffect, oWant`
	`xIntent, xReact, oReact`
	`xAttr`

ATOMIC$^{20}_{20}$ (Hwang et al., 2021)	

Physical-Entity
Event-Centered
Social-Interaction
	`ObjectUse, AtLocation, MadeUpOf, HasProperty, CapableOf, Desires, Not Desires`
	`IsAfter, HasSubEvent, IsBefore, HinderedBy, Causes, xReason, isFilledBy`
	`xNeed, xAttr, xEffect, xReact, xWant, xIntent, oEffect, oReact, oWant`

Table 9: Taxonomy of CSKB relations, cited from previous works (Sap et al., 2019; Hwang et al., 2021). In the context of this work, we are only interested in 15 relations included in CKBPv2.

from the results of three different prompt designs.

Relation	Template
xWant	as a result, PersonX wants to
oWant	as a result, PersonY or others want to
xEffect	as a result, PersonX will
oEffect	as a result, PersonY or others will
xReact	as a result, PersonX feels
oReact	as a result, PersonY or others feel
xAttr	PersonX is seen as
xIntent	because PersonX wanted
xNeed	but before, PersonX needed
Causes	causes
xReason	because
isBefore	happens before
isAfter	happens after
HinderedBy	can be hindered by
HasSubEvent	includes the event or action

Table 10: Readable templates from Hwang et al. (2021) (denoted as H-template), concerning 15 relations which comprise CKBPv2. When these templates are used in the main-task component, the head and tail events will be respectively prepended and appended to the templates.

Relation	Template
xWant	<*h*>, thus, <*t*>
oWant	<*h*>, thus, <*t*>
xEffect	<*h*>, thus as an result, <*t*>
oEffect	<*h*>, thus as an result, <*t*>
xReact	<*h*>, thus as a result on PersonX's emotion, <*t*>
oReact	<*h*>, thus as a result on PersonY's emotion, <*t*>
xAttr	<*h*>, thus it can be seen about PersonX's attribute that <*t*>
xIntent	<*h*>, thus it can be seen about PersonX's intention that <*t*>
xNeed	The event <*h*> will not happen unless <*t*>
Causes	Because <*h*>, <*t*>
xReason	<*h*>, because <*t*>
isBefore	After <*h*>, <*t*>
isAfter	Before <*h*>, <*t*>
HinderedBy	The event <*h*> will not happen, if <*t*>
HasSubEvent	The event <*h*> includes the event/action that <*t*>

Table 11: Self-curated readable templates (denoted as S-template) for 15 relations in CKBPv2. <*h*> and <*t*> denote the head and tail event respectively.

Constraint	Prompt Designs

Typing	**Design 1 (selected):**
	Which aspect (among three options 1. event/activity, 2. persona, 3. mental state) of the subject does the clause <*t*> express? Answer the choice only.
	
	Design 2 (alternative):
	Determine if clause <*t*> expresses an event or activity of the subject. Answer "Yes" or "No" only.

Temporal	**Design 1 (selected):** Which one of the following two statements is more plausible: 0. <*t*> before <*h*>, 1. <*t*> after <*h*>. Answer 0 or 1 only. **Design 2 (alternative):** Judge if the event <*t*> likely occurs after the event <*h*>. Answer "Yes" or "No" only.
Ambiguity	**Design 1:** Which one of the following two statements make more sense: 0. Two clauses <*h*> and <*t*> all have clear meaning. 1. One of two following clauses <*h*> and <*t*> has ambiguous meaning. Answer 0 or 1 only. **Design 2:** Judge if the meaning of the clauses <*h*> and <*t*> are all clear. Answer 'Yes' or 'No' only.

Table 12: Constraint prompt designs for typing, temporal, and ambiguity constraints. <*h*> and <*t*> denote the head and tail event respectively.

Baselines	Prompt Designs
Zero-shot	**Design 1:** Answer whether the following statement is plausible. Answer with only Yes or No: <free-text H-template format of the test triple> **Design 2:** Answer whether the following statement is plausible. Answer with only Yes or No: <free-text S-template format of the test triple> **Design 3:** Judge the following statement if it's likely to occur, only answer True or False: <free-text S-template format of the test triple>
Few-shot	**Design 1:** Answer whether the following statement is plausible. Answer with only Yes or No: Statement: If PersonX push PersonY back, as a result, PersonY or others will, PeopleX step back from PersonX Answer: Yes Statement: If PersonX regain PersonY 's composure, can be hindered by, PersonY be disown personx Answer: Yes Statement: If PersonX be nowhere, can be hindered by, PersonX friend will not keep PersonY Answer: Yes Statement: If PersonX chase PersonZ away, as a result, PersonX will, PersonY lose friend Answer: Yes Statement: If PersonX wave PersonY away, as a result, PersonX will, PersonY roll PersonZ eye Answer: Yes Statement: <free-text H-template format of the test triple> Answer: **Design 2:** Answer whether the following statement is plausible. Answer with only Yes or No: Statement: PersonX stay away from PersonY, thus as an result, PersonX call out to PersonX Answer: No Statement: PersonX help the PersonY, thus as an result, PersonX be rebuff by PersonY Answer: Yes Statement: PersonX turn down that, thus it can be seen about PersonX's attribute that PersonX get PersonY into trouble Answer: No Statement: PersonX be real, thus as an result, PersonY argue PersonZ about it Answer: Yes Statement: PersonX challenge PersonZ 's friend, thus, PersonY want PersonY not let Answer: Yes Statement: <free-text S-template format of the test triple> Answer: **Design 3:** Judge the following statement if it's likely to occur, only answer True or False: Statement: PersonX get PersonX thing together, thus it can be seen about PersonX's attribute that PersonX be helpful Answer: True Statement: PersonX invite PersonY to lunch, thus, PersonY want PersonX be a leader Answer: False Statement: PersonX catch, thus as a result on PersonY's emotion, PersonY feel PersonY be fluster Answer: True Statement: PersonX break PersonX glass, thus as a result on PersonX's emotion, PersonX feel PersonX be ashamed Answer: True Statement: The event PersonX need to set plan will not happen unless PersonX know about it Answer: False Statement: <free-text S-template format of the test triple> Answer:
Zero-shot-CoT	**Design 1:** Answer whether the statement <free-text H-template format of the test triple> is plausible. Let's think step by step, then conclude by answering True or False. **Design 2:** Answer whether the statement <free-text S-template format of the test triple> is plausible. Let's think step by step, then conclude by answering True or False.

Table 13: Prompt designs of baselines. H-template and S-template denote two sets of readable templates, from Hwang et al. (2021) and self-curated. For baselines which select exemplars randomly or based on the test instance, the exemplars will change according to each test instance, rather than remaining the same as what are shown here.

Baselines	Prompt Designs
Zero-shot-CoT	**Design 3:** Judge the statement <free-text S-template format of the test triple> if it's likely to occur. Let's think step by step, then conclude by answering True or False.
Few-shot-CoT	**Design 1:** Q: Answer whether the following statement is plausible: PersonX find PersonY happens before PersonX wake up on ground. A: Let's think step by step. If PersonX just wake up on the ground, that means PersonX was sleeping before that. Thus, PersonX cannot be conscious to find another person. Thus, the statement is not likely to occur. Q: Answer whether the following statement is plausible: PeopleX deserve happiness, as a result, PersonX feels, PersonX reach out to PeopleX. A: Let's think step by step. The events 'PeopleX deserve happiness' and 'PersonX reach out to PeopleX' are likely irrevelant. Also, the clause 'PersonX reach out to PeopleX' does not describe what PersonX feels. Thus, the statement is not likely to occur. Q: Answer whether the following statement is plausible: PersonX have a sheet, but before, PersonX needed PersonX meet PersonY requirement. A: Let's think step by step. If PersonX doesn't meet PersonY requirement, PersonX likely doesn't get the reward from PersonY. However, in this case, it's not clear whether 'a sheet' refer to PersonY's reward or not. Thus, the statement is not likely to occur. Q: Answer whether the following statement is plausible: PersonX occupy PersonY position, as a result, PersonX wants to PersonY want to aid in position. A: Let's think step by step. When PersonX occupy PersonY position, it means PersonY already worked at this position and has experience to do the job. Therefore, it's likely that PersonX want PersonY to aid PerosonX when PersonX is in that job position. Thus, the statement is likely to occur. Q: Answer whether the following statement is plausible: PersonX see that, as a result, PersonX will PersonX want a pet. A: Let's think step by step. In this context, we can refer the word 'that' as some activity where people play with their pet. Therefore, it stimulates PersonX's desire to have a pet. Thus, the statement is likely to occur. Q: Answer whether the following statement is plausible: <free-text H-template format of the test triple>. A:
Constraint-L2M	**Design 1:** For each statement below, please answer several questions to reach the final conclusion if the statement is commonsense. Whenever your answer of a question is No, please conclude that the statement is not commonsense. Otherwise, please conclude that the statement is commonsense. Statement: If PersonX prevent PersonY, as a result, PersonX feels, PersonX never reach out to anyone Q: Which aspect (among three options 1. event/activity, 2. persona, 3. mental state) of the subject does the clause 'PersonX never reach out to anyone' express. Answer the choice only. A: 1. event/activity Q: Is the above answer different from option 1. event/activity? A: No. Thus, the statement is not commonsense Statement: If PersonX go to sleep on hollow, as a result, PersonX feels, PersonX be tired Q: Which aspect (among three options 1. event/activity, 2. persona, 3. mental state) of the subject does the clause 'PersonX feel PersonX be tired' express. Answer the choice only. A: 3. mental state Q: Is the above answer different from option 1. event/activity? A: Yes Q: Is the statement "If PersonX go to sleep on hollow, as a result, PersonX feels, PersonX be tired" plausible? A: No. Thus, the statement is not commonsense Statement: If PersonX eat the sub, as a result, PersonX feels, PersonX be full Q: Which aspect (among three options 1. event/activity, 2. persona, 3. mental state) of the subject does the clause 'PersonX feel PersonX be full' express. Answer the choice only. A: 3. mental state Q: Is the above answer different from option 1. event/activity? A: Yes Q: Is the statement "If PersonX eat the sub, as a result, PersonX feels, PersonX be full" plausible? A: Yes. Thus, the statement is commonsense

Table 14: (Cont.) Prompt designs of baselines. H-template and S-template denote two sets of readable templates, from Hwang et al. (2021) and self-curated. For baselines which select exemplars randomly or based on the test instance, the exemplars will change according to each test instance, rather than remaining the same as what are shown here. For concision, we only show **Design 1** for Few-shot-CoT and Constraint-L2M.