Can Language Models Take A Hint? Prompting for Controllable Contextualized Commonsense Inference

Anonymous ACL submission

Abstract

Generating commonsense assertions within a given story context remains a difficult task for modern language models. Previous research has addressed this problem by aligning commonsense inferences with stories and training language generation models accordingly. One of the challenges is determining which topic or entity in the story should be the focus of an inferred assertion. Prior approaches lack the ability to control specific aspects of the generated assertions. In this work, we introduce "hinting," a data augmentation technique that enhances contextualized commonsense inference. "Hinting" employs a prefix prompting strategy using both hard and soft prompts to guide the inference process. To demonstrate its effectiveness, we apply "hinting" to two contextual commonsense inference datasets: ParaCOMET (Gabriel et al., 2021) and GLUCOSE (Mostafazadeh et al., 2020), evaluating its impact on both general and context-specific inference. Furthermore, we evaluate "hinting" by incorporating synonyms and antonyms into the hints. Our results show that "hinting" does not compromise the performance of contextual commonsense inference while offering improved controllability.

1 Introduction

The task of Contextual or Discourse-Aware Commonsense Inference, which consists of generating relevant and coherent commonsense assertions (i.e. facts) for a certain sentence in a story context, while easy for humans, remains challenging for machines (Gabriel et al., 2021). Within this task, we define an *assertion* as a tuple that contains a subject[1], a relation, and an object (e.g., *a dog, is a, animal*), similar to a subject-verb-object triple. An assertion in this task is a contextually *specific* fact or *generally* applicable templated fact, that can be inferred from a sentence within a given story context.

Automated systems (such as pre-trained transformer-based language models (Devlin et al., 2019; Radford et al., 2019)) struggle with generating contextual assertions since there is an implicit assumption that clues for making predictions can always be found explicitly in the text. (Da and Kasai, 2019; Davison et al., 2019; Liu and Singh, 2004; Zhang et al., 2021). This becomes problematic because a model for this task is essentially forced to use knowledge that it may not have seen during pre-training. Additionally, models are forced to guess what to predict about (e.g., what the subject of an assertion is), which may lead to decreased performance (e.g., the model generates an assertion about cats when it should have talked about dogs). Below we give an example of contextual commonsense inference with a story, *a target sentence*, and some corresponding story *specific* and *general* inferences. The story is picked directly from the ROCStories corpus. (Mostafazadeh et al., 2016)

> **Story:** The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! *They scored a final goal!*
>
> **Story Specific Commonsense Inference:** The red team, is capable of, winning the game
>
> **General Commonsense Inference:** Some people *scored a final goal*, causes, some people to be happy

In this example, we can see the aforementioned problems that models have to deal with. Although it is commonsense that a final goal will lead to a victory for the red team, it is not explicitly stated anywhere in the text. The pre-training of models may include text related to a sudden death goal from sources such as Wikipedia, but the model has to extrapolate that the final goal in this example is a type of sudden death goal and will concede victory to the red team. Similarly, although we may want assertions related to the red team, the model has

[1]We note that here we utilize the term "subject" as a part of the relation tuple, and it is not necessarily "subject" in a grammatical sense. In the case of ATOMIC, a subject could be a sentence describing an event that causes another event or a reaction, whereas in ConceptNet it could be a single concept.

to somehow know that it *needs* to infer assertions about this and **not** something else.

Previous attempts have tackled the problem of contextual commonsense inference by constructing datasets of stories aligned with assertions (i.e., an assertion is given for a sentence in a story), either through automated or human-annotated ways and building a model to, given the story and a target sentence, predict part or the whole assertion. A previous attempt to tackle this problem, ParaCOMET (Gabriel et al., 2021), trained a GPT-2 language model (Radford et al., 2019) to infer an object of a commonsense assertion tuple from the ATOMIC (Sap et al., 2019) knowledge base[2]. They formulate the task as follows. Given a story, a special sentence identifier token, and a specified relation type (i.e. dimension of commonsense), the ParaCOMET model must predict the object of a commonsense assertion.

Another work that tries to approach this is GLUCOSE (Mostafazadeh et al., 2020). Here, a dataset is constructed that consists of stories and human annotations for sentences in the stories. Human annotations provide *specific* and *general* commonsense assertions. The authors use this data set to train a T5 (Raffel et al., 2020) model to generate both types of assertions. The model takes an input sequence in the form of a story, a relation to predict, and a target sentence, and has to predict both the *general* and *specific* assertions that may be present in the target sentence with the given story context. In both works, the models are expected to infer from the story, a target sentence, and the relation alone. An example of these formulations can be seen in the Appendix.

Recently, there has been work on exploring *prompting* (Liu et al., 2021), which essentially involves finding ways of altering the input to a language model such that it matches templates that it has seen during pre-training. Prompting a model correctly can give stronger performance in tasks, help with controllability in the case of text generation, and is more parameter-efficient and data-efficient than fine-tuning (Li and Liang, 2021). One type of prompting is *prefix prompting* (Li and Liang, 2021; Lester et al., 2021). Prefix prompting consists of modifying a language model's input (i.e., prefix) by adding additional content. This can be explicit hard prompts (i.e., actual words such as "give a happy review") or soft prompts (i.e., embeddings that are input into a model and can be trained to converge on some virtual template or virtual prompt that can help the model).

We introduce the idea of a *hint*, a hybrid of hard and soft prompts. We define a *hint* as an additional input to a model in the form of a part(s) of an assertion that a model has to predict, along with special identifier tokens for these parts, wrapped within parenthesis characters. Syntactically, a *hint* would take the form of: "*([subject symbol, subject], [relation symbol, relation], [object symbol, object])*" where the actual content of the *hint*, between the parenthesis, would be a permutation of all but one of the elements of the target tuple during training. In the case of supplying hints to GLUCOSE, we include a special *specific* or *general* token which determines whether the part of the hint belongs to a story *specific* assertion or a *general* assertion. For clarity, an example of a *hint* for the hockey example using the GLUCOSE formulation would be "*(<|specific|><|subj|>The red team scores, <|general|><|obj|>People_A win the game)*". Altogether, the model input would be:

> **Model Input:** 1: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! **They scored a final goal!** *(<|specific|><|subj|>**The red team scores**, <|general|><|obj|>**People_A win the game**)*
>
> **Model Target/Output:** **The red team scores**, Causes/Enables, The red team wins the game ** People_A score, Causes/Enables, **People_A win the game**

Hints are provided during training by sampling a binomial distribution (with $p = 0.5$) for each element in a minibatch, which determines whether to give a *hint* or not. The actual content of the *hint* would then be generated by random sampling without replacement of up to all but one of the elements in a target tuple. Once more, this only happens during training, later on one can supply whatever is desired as a hint to guide the generation of the commonsense inference.

We hypothesize that this scheme of *hinting* strikes a balance between the model recalling information from its pre-training, with information that it may not have seen that may only be present in the target tuple. Additionally, by providing and fine-tuning a model on the combination of hard and soft prompts, a generative language model can be guided to "talk" about a certain subject, object, or relation, thus enabling finer control of the models in downstream applications. We note that the approach was designed to be simple to implement and to give control when generating text. In the following sections, we give some background and

[2]ATOMIC is composed of causal assertions, where a certain subject event, causes a certain object event through a given relation.

follow this with a set of experiments to show the effects of *hinting* for the ParaCOMET and GLUCOSE datasets, and finally, analyze the results, and present future directions for this work. Concretely, our contributions are:

- A hybrid prefix prompting technique called *hinting* that provides a partial assertion to augment data for contextual commonsense inference, and
- Demonstrating that *hinting* improves the performance for contextual commonsense inference as measured by automated metrics and is comparable in human-based metrics.

2 Related Work

2.1 Prompting

Recently, there has been a shift in paradigm in Natural Language Processing from pre-training and fine-tuning a model, to pre-training, prompting, and predicting (Liu et al., 2021). One primary reason for this shift is the creation of ever-larger language models, which have become computationally expensive to fine-tune. Prompting can be described as converting a pre-trained language model input sequence into another sequence that resembles what the language model has seen during pre-training. Overall, most prompting research is focused on formulating the task as a *cloze* (fill-in-the-blanks) task. However, we consider the task of language generation, an open-ended formulation.

Recall that prefix prompting modifies the input to a language model, by adding either a hard prompt (additional words to the input sequence)(Shin et al., 2020) or a soft prompt (i.e., adding trainable vectors that represent, but are not equivalent to, additional words) (Li and Liang, 2021; Lester et al., 2021; Liu et al., 2021).

Unlike classic prefix prompting, *hinting* uses both hard and soft prompts. The soft prompts are in the form of symbols that represent the different parts of the assertion (i.e., subject, relation type, and object), and the hard prompts are in the form of the actual parts of the assertion that are selected to be appended as part of the *hint*. Our work is similar to KnowPrompt (Chen et al., 2021a), except that they use a masked language model and soft prompts for relationship extraction. AutoPrompt (Shin et al., 2020) is also similar, but finds a set of "trigger" words that give the best performance on a *cloze*-related task, whereas we provide specific structured input for the model to guide text generation. We additionally note that although there are prompt-based relation extraction models (Chen et al., 2021b), we are performing a different task which is contextual commonsense inference. Another recent contribution, P-Tuning (Liu et al., 2023), shares similarities with our approach by combining trainable continuous prompt embeddings with discrete prompts. Both hinting and P-tuning share the overarching objective of enhancing prompt learnability. PTR, or prompt-tuning with rules (Han et al., 2021), shares similarities with hinting as it involves encoding prior knowledge about tasks and classes. Similarly to PTR, hinting also introduces additional information to provide the model with contextual understanding of the relationships between words.

2.2 Controllable Generation

Controllable generation can be described as ways to control a language model's text generation given some kind of guidance. One work that tries to implement controllable generation is CTRL (Keskar et al., 2019). The authors supply control signals during pre-training of a general language model. A body of work in controllable generation has focused on how it can be used for summarization. Representative work that uses techniques similar to ours is GSum (Dou et al., 2021).

In contrast to GSum, our method is model independent, allows for the source document to interact with the guidance signal, and contains soft prompts in the form of trainable embeddings that represent the parts of a tuple. The GSum system gives interesting insight into the fact that highlighted sentences, and the provision of triples, does in fact help with the factual correctness of abstractive summarization. We make the distinction that *hinting* falls more under prompting for the reason that we utilize additionally the trainable soft embeddings rather than purely additional hard tokens and that our task of contextual commonsense generation is not explored in the controllable generation works, whose main focus is on controlling unstructured text generation. Some works that are in this area are also (Peng et al., 2018) who utilize what they call "control factors" as keywords or phrases that are supplied by a human-in-the-loop to guide a conversation. Another work, Diffusion-LM (Li et al., 2022), developed a language model that utilizes a sequence of Gaussian noise vectors that gradu-

ally transform into words, creating an organized structure to guide what the model generates. More similar to our work, but tailored for the task of interactive story generation and without trainable soft-embeddings, is the work by (Brahman et al., 2020) which uses automatically extracted keywords to generate a story. Future work we could possibly utilize the automatic keyword extraction to supply parts of a hint, rather than our approach of complete parts of an assertion, and expand this to utilize synonyms of keywords. Lastly, there is the work by (See et al., 2019) which looks at controllable text generation for the purpose of conversation and utilizes an embedding give quantitative control signals as part of conditional training.

2.3 Discourse-aware/Contextual commonsense inference

Commonsense inference is the task of generating a commonsense assertion. Discourse-aware/contextual commonsense inference is the task of, given a certain narrative or discourse, inferring commonsense assertions that are coherent within the narrative(Gabriel et al., 2021). This task is particularly hard because commonsense knowledge may not be explicitly stated in text (Liu and Singh, 2004) and the model needs to keep track of entities and their states either explicitly or implicitly. Research into the knowledge that pre-trained language models learn has yielded good results in that they do contain various types of factual knowledge, as well as some commonsense knowledge(Da and Kasai, 2019; Petroni et al., 2019; Davison et al., 2019). The amount of commonsense knowledge in these models can be improved by supplementing sparsely covered subject areas with structured knowledge sources such as ConceptNet (Speer et al., 2017; Davison et al., 2019).

Knowing that these pre-trained language models may contain some commonsense information has led to the development of knowledge models such as COMET(Bosselut et al., 2019). This line of research has been extended from the sentence-by-sentence level in COMET, to the paragraph-level in ParaCOMET (Gabriel et al., 2021). Contemporaneously, GLUCOSE Mostafazadeh et al. (2020) builds a dataset of commonsense assertions that are contextualized to a set of stories, and generalized. More recently, the idea of knowledge models (Da et al., 2021) or models that can be leveraged to generate commonsense assertions has been gaining track. One recent approach has been kogito (Ismayilzada and Bosselut, 2022). Kogito is a toolkit for commonsense inference, which permits training and access of similar to COMET, along with providing tools for selecting a subject and a relation. Kogito utilizes the same formulation as the ParaCOMET work, in which a subject and a relation are provided. However, kogito does not tackle the more complicated general commonsense inference as in GLUCOSE. With our work, we could provide kogito with a framework to be able to train models that can perform this type of controllable, generalized inference and improve the overall training.

3 Modeling

3.1 Task

We now detail the task of Contextual Commonsense Inference. We are given a story S composed of n sentences, $S = \{S_1, S_2, \ldots, S_n\}$, a target sentence from that story, S_t, where $S_t \in S$, and a dimension/relation type R. Given all this, we want to generate a tuple in the form of $(subject, R, object)$ that represents an assertion, present or implied, in S_t given the context S, and the relation type R.

We run tests with two variations of this task, one is the ParaCOMET variation and the other the GLUCOSE variation. In the ParaCOMET experiments, we represent S_t with a special token. Additionally, we only generate the $object$ of the tuple. In our GLUCOSE experiments, we represent S_t by marking it with $*$ on the left and right of the actual target sentence. Additionally, we generate at most two *subject, R, object* tuples: one that is the context-specific tuple, and the other is the general tuple, separated by two asterisks (**). An example of how the inputs look for both datasets can be seen earlier in the Introduction.

3.2 Hinting

The mechanism we present in this work, called *hinting*, is a kind of mixed/hybrid prompting for generative language models. Prompting is essentially supplying additional text (i.e. prompts) to a language model to aid/guide it in a specific task. In our case, we opt to give a “hint", as to what the assertion that we want to predict contains, at the end of our input text. We chose placing the hint at the end of the input for simplicity in dataset processing, but it can be placed anywhere and we leave it as future work to explore the effects of placing hints possibly next to the target sentence or at the beginning of the input. Hinting can be seen as

Model Input	Model Output
7: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal!. * They scored a final goal!. *	They scored a final goal >Causes>They feel(s) happy ** Some People_A scored a final goal >Causes>Some People_A feel(s) happy
7: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal!. * They scored a final goal!. * **hint: (<|specific|><|subj|>the red team scores the final goal)**	**the red team scores the final goal** >Causes>the red team feel(s) happy ** Some People_A (who are a team) score the final goal >Causes>Some People_A feel(s) happy
7: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal!. * They scored a final goal!. * **hint: (<|general|><|subj|> Something_A (that is a point))**	They scored a final goal >Causes>They feel(s) happy ** **Something_A (that is a point))** is scored >Causes>Some People_A feel(s) happy

Table 1: Example of inputs and outputs for the GLUCOSE trained model with hints. The *hint* is **bolded**). Without a hint we can see that the model does a good faith attempt with the given context and with a "hint" it is guided.

a hybrid of prompting the generative model with hard prompts composed of parts of what should be predicted along with soft prompts of symbols that represent those parts. These symbols are for the *subject*, *relation (R)*, and *object* respectively. These soft prompts utilize untrained embeddings for the task. We structure hinting this way such that, after training, whenever a *hint* is given, the model can be guided to generate assertions about the *hint's* content based on the target sentence and context.

To balance the model's reliance on the context, its knowledge, and the *hint*, we determine whether to supply the *hint* by sampling a binomial distribution ($p = 0.5$). Thus, we can control the frequency of when to supply a *hint*. Additionally, the content of the *hint* is determined by random sampling of permutations of components, up to a maximum of all but one component. Since our task is to predict the tuple, we do not want to make the model overly reliant on *hints* for the answer.

3.3 An example of Hinting

A simple example of *hinting* is the following:

> **Story:** *The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! They scored a final goal!*
>
> **Target sentence:** *They scored a final goal!*
>
> **Target assertion:** *(subject: the red team, relation: are capable of, object: winning the game.)*

A *hint* can be any permutation of the target assertion, except the complete assertion, along with some symbol that indicates which part it is:

> **Possible Hints:**
> *(<|subj|> the red team),*
> *(<|obj|> winning the game),*
> *(<|rel|> capable of),*
> *(<|subj|> the red team, <|rel|> capable of),*
> *(<|subj|> the red team, <|obj|> winning the game),*
> *(<|rel|> capable of, <|obj|> winning the game)*

Sampling randomly from one of those permutations, a *hint* for the given story, target sentence and target assertion, yields the following:

> **Hint:** *(<|subj|> the red team, <|rel|> capable of)*

Putting everything together, the input for the model would be:

> **Story with Hint:** *The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! They scored a final goal!* ***(<|subj|> the red team, <|rel|> capable of)***.

We note that this is a general version of how the hinting mechanism works. The dataset specific hints that we utilize are described in the Appendix.

3.4 Hinting with Synonyms and Antonyms

To explore other ways of hinting, we devised another technique by swapping parts of the hint with synonyms and antonyms. We used the synsets and antsets from the WordNet (Fellbaum, 2010) knowledge base to find synonyms and antonyms respectively. In the case that there is no viable synonym or antonym we keep the original word. To introduce synonyms and antonyms, the process is as follows. When a hint is generated, we copy it and we decide whether to supply it with synonyms and/or antonyms by swapping the words in the hint prompt with their synonyms or antonyms by sampling from a binomial distribution (we explored various values for p, with 0.5 being the most effective) to control the frequency of hints with synonyms/antonyms. The reasoning for this is that if we only supply synonyms/antonyms, we lose the control that hinting provides, so we introduce synonyms and antonyms at a certain rate to have the model still be controllable, while seeing "new" information through the synonyms/antonyms. After the swapping process, we insert a soft prompt, <|*syn*|> or <|*ant*|>, at the end of the substituted prompt to signal that the model is using a synonym or an antonym respectively. We give an example of how these hints look in the Appendix.

3.5 Models

For our first set of experiments, we utilize the ParaCOMET (Gabriel et al., 2021) dataset and the framework with the same GPT-2 model as ParaCOMET, along with a T5 (Raffel et al., 2020)

model to observe the effects of *hinting* in a sequence-to-sequence formulation of the dataset. We use the off-the-shelf (Huggingface (Wolf et al., 2019)) pre-trained "base" version of these models for efficiency. For our second set of experiments with the GLUCOSE dataset, we also used the T5 model, as was done in GLUCOSE.

4 Experimental Setup

To show the effectiveness of *hinting* we use the following setups. First we utilize the original ParaCOMET dataset and setup and adding *hints with/without synonyms and antonyms* . The ParaCOMET setup consists of given a story S composed of n sentences, $S = \{S_1, S_2, \ldots, S_n\}$, a relation type R, and a target sentence token (i.e. <|*sent0*|>, <|*sent1*|>, ..., <|*sent(n-1)*|>). In the ParaCOMET dataset, we must predict the *object* of a triple, utilizing implicitly the sentence as a subject and explicitly the supplied sentence symbol and relation R symbol.

Within this framework, after the relation R, we add our *hint* between parenthesis (i.e. "([*hint*])"). In this framing, our *hint* can be composed of: a *subject* symbol (<|subj|>) along with the target sentence to serve as a *subject*, a relation symbol (<|rel|>) along with the *relation* R, or an *object* symbol (<|obj|>) along with the *object* of the triple. Using the hockey example a possible *hint* in this set of experiments would be: "(<|rel|> <|xEffect|>,<|obj|> they win the game)". In the case that we add a synonym and/or antonym, we do the appropriate replacement and add the <|*syn*|> or <|*ant*|> tags. It is possible, although we leave for future work, to include both the hint *and* the synonym/antonym hint as part of the prompt.

In our GPT-2 experiments, we utilize the same cross-entropy loss as in (Gabriel et al., 2021). We note that we also utilize a sequence-to-sequence (Sutskever et al., 2014) formulation for the T5 model. This in contrast to the GPT-2-based system requires encoding a source sequence (i.e., story, target sentence, and relation symbol), and decoding it into a target sequence (i.e., the *object* of an assertion). For the T5 model, we add the prefix "source:" before the story S, and the prefix "hint:" for placing our *hints*. In addition, whenever there is a synonym or antonym added, it is added as another prefix ("synonym:[synonym]" or "antonym:[antonym]"). For simplicity, we construct the same "heuristic" dataset as ParaCOMET which utilizes a heuristic matching technique to align ATOMIC (Sap et al., 2019) triples to story sentences.

Secondly, we utilize the formulation utilized in GLUCOSE (Mostafazadeh et al., 2020). The formulation utilizes the T5 model in a sequence-to-sequence formulation once more. In this formulation, the source text is composed of a prefix of a dimension to predict $D \in 1, 2, \ldots 10$[3], followed by the story S with the marked target sentence. The target sentence, S_t, is marked with "$*$" before and after the sentence. An example input is: "1: The first sentence. **The target sentence.* * The third sentence.". This task is slightly different from the ParaCOMET one, in that in addition to predicting a context *specific* triple, the model has to predict a *generalized* triple.

In the GLUCOSE task we have to infer both *general* and context *specific subject*, *object* and *relation* elements. For our *hints* we provide up to five out of these six elements while training, along with a symbol that represents whether it is the *subject*, *object* or a *relation*, and another symbol that represents whether it is part of the *general* or *specific* assertion. We add our *hint* after the story S, utilizing the prefix "hint:" and supplying the *hint* between parenthesis. Given the hockey story, an example of a *hint* for GLUCOSE can be: "***(<|general|> <|obj|> People_A win a Something_A)***". Hyperparameter configuration details can be seen in the Appendix.

Thirdly, for testing the controllability of the model, we train 5 models on the GLUCOSE data: one without hints, one with hints, one with hints and synonyms, one with hints and antonyms, and one with hints, synonyms, and antonyms. To test the control, we make a synthetic test in which we utilize the GLUCOSE testing data, and supply the model with hints for the specific subject and/or relation and/or the general subject and/or relation. The test measures the overlap between the elements provided in the hint and the elements present in the output. We note that this is a synthetic benchmark which demonstrates that the model is capable of incorporating the hint into its output accordingly.

Lastly, we ran a tiny Mechanical Turk study similar to the one presented in the original ParaCOMET (Gabriel et al., 2021) in which a human judges the plausibility of the generated assertion based on the context on a 5-point Likert scale: obviously true (5), generally true (4), plausible (3), neutral or un-

[3]The definition for these numbers is in (Mostafazadeh et al., 2020)

Model	BLEU1	BLEU2	BLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM
GPT-2	90.507	83.464	42.406	68.858	61.985	51.051	61.951	61.952
GPT-2 Hint	91.410*	85.934*	43.519*	69.558*	63.501*	52.763*	63.481*	63.477*
GPT-2 Hint+Synonyms	**91.782***	**86.610***	43.611*	69.592*	63.559*	52.825*	63.546*	63.539*
GPT-2 Hint+Antonyms	91.622*	86.491*	**43.652***	**69.605***	**63.636***	**52.915***	**63.619***	**63.619***
GPT-2 Hint+Syn.+Ant.	91.497*	85.850*	43.495*	69.428*	63.311*	52.579*	63.292*	63.290*
T5	94.133	80.462	38.585	65.815	56.095	44.636	56.078	56.079
T5 Hint	93.851*	81.284	39.622	66.409*	57.623	46.441	57.591	57.593
T5 Hint+Synonyms	94.562	**83.953**	**40.656***	**67.381***	**58.385***	**47.565***	**58.348***	**58.350***
T5 Hint+Antonyms	**95.482***	82.286*	38.735	65.917	56.223	44.748	56.191	56.190
T5 Hint+Syn.+Ant.	94.543	81.193	39.153	66.465	56.805	45.588	56.765	56.766

Table 2: Averages of 4 different seeds over 5 epochs for the ParaCOMET dataset from (Gabriel et al., 2021) for different hinting configurations. The largest scores are **bolded** and scores that are significantly different from the non-hinted model have an asterisk (*) next to them. We can see from the results that *hinted* systems tend to achieve higher performance even if slightly and in some cases significantly. We also see that hinting training with synonyms tends to be more beneficial than with antonyms, but both improve the performance of hinting.

Model	BLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM
T5	75.998	80.683	80.282	67.349	78.624	78.622
T5 Hint	76.266	**80.971**	**80.543**	**67.725**	**78.894**	**78.908**
T5 Hint+Synonyms	75.848	80.879	80.511	67.605	78.844	78.843
T5 Hint+Antonyms	**76.282***	80.840*	80.448	67.720	78.846*	78.858*
T5 Hint+Syn.+Ant.	76.138	80.798	80.365	67.607	78.746	78.756

Table 3: Averages of 4 different seeds over 5 epochs for the GLUCOSE dataset from (Mostafazadeh et al., 2020) for different hinting configurations. The largest scores are **bolded** and scores that are significantly different ($p<0.05$ in T test) from the non-hinted model have an asterisk (*) next to them. Once more we see that hinted models tend to perform better or comparable to not hinting. Here we see that the benefits of adding synonyms and antonyms to training are not as pronounced as in Table 2.

clear (2), and doesn't make sense (1). We present the results in the same manner where Table 4 displays the percent of inferences judged as plausible or true (3-5), and the average rating per inference. Participants were given $0.1 to complete the task. We give an image of the HIT in the Appendix. We sample from the GLUCOSE test set, 75 entries randomly. Then, for all the trained models, we generate assertions for these entries and hand them off to 3 human raters.

5 Results and Analysis

5.1 Experiment 1: ParaCOMET with *hints*

The aggregated results for this set of experiments can be found in Table 2. We can see here that on average, *hinting* does tend to improve the score even if slightly. It seems that providing a *hint* is beneficial and not detrimental for contextual commonsense inference. Given the way that this task is framed, a possibility that could explain the relative similarity of the performances, is that *hinting* in this formulation **only** adds the *object* of the triple as additional possible data that the model may see during training; the subject and the relation can be repeated with *hinting*. We note that the performance of the T5 model was less, and we believe that it may be lack of hyperparameter tuning, as it was seen that the model was sensitive to the learning rate and had to use a higher than usual learning rate. In these tests, we also note that hinting with synonyms tends to consistently improve the performance even further than just plain hinting, indicating that the model benefits from making associations of related concepts. We see that hinting with antonyms also tends to be beneficial, but the benefit is not as consistent as with synonyms. Interestingly, we see that combining both synonym and antonym training does not bring the best of both worlds, but more closely an average between the performance of hinting with either.

5.2 Experiment 2: GLUCOSE with *hints*

The aggregated results for this set of experiments can be found in Table 3. Once more we notice that *hinting* (with and without synonyms and/or antonyms) does tend to improve the performance of the contextual commonsense inference task. This suggests that *hinting* is indeed beneficial for the task of contextualized commonsense inference, especially when faced with the harder task of generating both a general and context dependent assertion. We believe that this improvement is because *hinting* gives the model the clues it may need to decide on what to focus or attend to, to generate useful inferences.

5.3 Experiment 3: Controllability

In Table 4 we see the results of our synthetic controllability test. We see that the model without hinting tends to predict about relevant things (indicated by the BLEU scores above 50), however when hints are injected, the model always predicts about what the hint was suggesting (indicated by the nearly perfect scores). We also see that supplying synonyms and antonyms does not decrease the controllability of the models.

Model	General Subject	General Subj.+Rel.	Specific Subject	Specific Subj.+Rel.	Specific Subj.+Rel. & General Subj.	Specific Subj.+Rel. & General Subj.+Rel.
T5	51.047	56.256	58.386	67.287	59.345	61.555
T5 Hint	99.928	**99.974**	99.981	**100.000**	99.963	99.977
T5 Hint+Synonyms	99.892	99.931	**100.000**	**100.000**	99.689	99.987
T5 Hint+Antonyms	99.939	99.839	**100.000**	**100.000**	**99.988**	99.987
T5 Hint+Syn.+Ant.	**99.988**	99.854	99.973	**100.000**	99.978	**99.990**

Table 4: Results of control tests to see if elements provided in hint show up in the output. All results are in BLEU. We see that models that are given hints during the training retain controllability in evaluation. We see that if a certain element of a relation is given in the hint, the model does a good effort, even with synonym and antonym training, of utilizing that in its output.

Model	Avg. Rating	Plausibility (>=3)
T5	3.77	98.66%
T5 Hint	**3.90**	96.05%
T5 Hint+Synonym	3.85	94.73%
T5 Hint+Antonym	3.76	97.36%
T5 Hint+Syn.+Ant.	3.86	**100.00**%

Table 5: Results of human evaluation of GLUCOSE dataset model. The largest scores are **bolded**. We sampled 75 test points for each model from the test dataset and had them infer assertions. Humans judged these assertions on a 5 point Likert scale where above 3 was plausible similar to (Gabriel et al., 2021). We had 3 raters on each inference and utilized the median of the rating. We can see that hinting provides similar plausibility and tends to be rated higher than not hinting at all.

5.4 Experiment 4: Human Judgements

The results for a tiny Mechanical Turk study for human evaluation of model inferences can be seen in Table 5. Overall we can see here that hinted systems are judged as slightly higher in plausibility. We also see upon looking some of the inferences that the hinted model tends to be more general and provide shorter responses than the non-hinted model (e.g., hinted inference: “satisfied" vs. non-hinted inference: “happy and satisfied").

6 Discussion

6.1 Why *hint*?

From the results of our experiments, we can see that *hinting* tends to increase the performance of contextualized commonsense inference at least with regards to automated metrics and does not significantly degrade or improve human judgements. This brings the question of: Why *hint* at all? The primary reason is for controllability in the generation. By supplying these *hints*, we are teaching the model pay attention and generate inferences about a certain subject, relation, or object. This in turn, after training, can be leveraged by a user or downstream application to guide the model to generate assertions from parts that are manually supplied. Although this is not very clear within the ParaCOMET formulation, it becomes clearer in the GLUCOSE formulation of the problem. We give an illustrative example of the usefulness of *hinting* in Table 1. We can see that by giving a model the *hint*, the model could be capable of inferring about information that may not be present in the story. We note that this behavior is useful in downstream tasks such as story understanding and contextual knowledge graph generation in which we may need a model to have a specific subject or object . Lastly, *hinting* was designed to be simple to implement, and is model independent.

6.2 Is *hinting* optimal?

This work was a proof of concept for this technique. We acknowledge there is a large body of research on the area of prompting. The way the *hinting* mechanism was designed however, leaves much space to explore alternate mechanisms such as AutoPrompt(Shin et al., 2020), including additional soft prompts such as those in Li and Liang (2021), or even replacing the contents of the hint with synonyms or related words. Because of the naivety of the approach, we do not think it is an optimal approach, and there is a large body of research that points to manual templating of prompts being less effective than learned prompts (Liu et al., 2021). However, from our tests, our approach does not degrade performance, and only improves it.

7 Conclusion

In this work we presented *hinting*, a simple hybrid prompting mechanism that consists of appending parts of a target tuple into an input sequence for the task of contextual commonsense inference. We showed that *hinting* tends to improve performance in automated metrics and provides comparable performance with human-based judgements. With this, we open the doors for exploring prompting within the realm of contextual commonsense inference.

The hinting system design acknowledges areas for improvement, particularly in developing smarter strategies for selecting when and what to hint, and enhancing the hint with additional soft prompts. Future work and exploration is further described in the Appendix F.

8 Ethics Statement

In this work, we propose a mechanism called "hinting" to create a controllable contextual commonsense inference model. Our goal is to improve the usability of contextual commonsense inference models in downstream applications. However, it's important to note that our mechanism may be subject to limitations due to biases existing in the knowledge bases used (e.g., ATOMIC, and GLUCOSE).

While our model is controllable to some extent, it could potentially generate harmful or incorrect assertions as a result of the conditioned biases. Additionally, since our model generates text, it might produce erroneous statements. We did not analyze the degree of these biases or incorrect inferences in this study; however, we utilize well-vetted knowledge bases which should minimize any significant negative impacts on performance.

9 Limitations

We note that our work has some limitations. One of these is the length of the stories that were given for the task of contextual commonsense inference. Many of these stories are around 5 sentences long. This may in turn harm the generalization of the effectiveness of this technique to longer stories. We also note that this technique was designed for language models that are not extremely large (>7B parameters) which have recently shown their effectiveness in a wide variety of tasks, however, since these are prompts, the hinting technique can be utilized as part of example prompts for these extremely large models. Lastly, one limitation of this method is that the inferences that are produced have no way of being evaluated on their relevance and truthfulness. This is to be addressed in future work with a classifier for these assertions.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, 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 (ACL)*.

Faeze Brahman, Alexandru Petrusca, and Snigdha Chaturvedi. 2020. Cue me in: Content-inducing approaches to interactive story generation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 588–597, Suzhou, China. Association for Computational Linguistics.

Xiang Chen, Xin Xie, Ningyu Zhang, Jiahuan Yan, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021a. Adaprompt: Adaptive prompt-based finetuning for relation extraction. *CoRR*, abs/2104.07650.

Xiang Chen, Xin Xie, Ningyu Zhang, Jiahuan Yan, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021b. Adaprompt: Adaptive prompt-based finetuning for relation extraction. *arXiv preprint arXiv:2104.07650*.

Jeff Da, Ronan Le Bras, Ximing Lu, Yejin Choi, and Antoine Bosselut. 2021. Analyzing commonsense emergence in few-shot knowledge models. *arXiv preprint arXiv:2101.00297*.

Jeff Da and Jungo Kasai. 2019. Cracking the contextual commonsense code: Understanding commonsense reasoning aptitude of deep contextual representations. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 1–12, Hong Kong, China. Association for Computational Linguistics.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. GSum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4830–4842, Online. Association for Computational Linguistics.

Christiane Fellbaum. 2010. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.

Saadia Gabriel, Chandra Bhagavatula, Vered Shwartz, Ronan Le Bras, Maxwell Forbes, and Yejin Choi. 2021. Paragraph-level commonsense transformers with recurrent memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12857–12865.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. Ptr: Prompt tuning with rules for text classification.

Mete Ismayilzada and Antoine Bosselut. 2022. kogito: A commonsense knowledge inference toolkit. *arXiv preprint arXiv:2211.08451*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Quentin Lhoest, Albert Villanova del Moral, Patrick von Platen, Thomas Wolf, Yacine Jernite, Abhishek Thakur, Lewis Tunstall, Suraj Patil, Mariama Drame, Julien Chaumond, Julien Plu, Joe Davison, Simon Brandeis, Victor Sanh, Teven Le Scao, Kevin Canwen Xu, Nicolas Patry, Steven Liu, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Nathan Raw, Sylvain Lesage, Anton Lozhkov, Matthew Carrigan, Théo Matussière, Leandro von Werra, Lysandre Debut, Stas Bekman, and Clément Delangue. 2021. huggingface/datasets: 1.13.2.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pages 4328–4343. Curran Associates, Inc.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.

Nasrin Mostafazadeh, Aditya Kalyanpur, Lori Moon, David 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)*, pages 4569–4586, Online. Association for Computational Linguistics.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1702–1723, Minneapolis, Minnesota. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed 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*, volume 35, pages 24824–24837. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhihan Zhang, Xiubo Geng, Tao Qin, Yunfang Wu, and Daxin Jiang. 2021. Knowledge-aware procedural text understanding with multi-stage training. In *Proceedings of the Web Conference 2021*, WWW '21, page 3512–3523, New York, NY, USA. Association for Computing Machinery.

A Mechanical Turk Survey

In Figure 1 and 2 we can see the survey that was handed to MTurk participants.

B Synonym/Antonym Configurations

The synonym/antonym prompts are added in the same way as the hinting prompts. We experimented with the p value of binomial distribution to control the proportion of synonym/antonym prompts we give during training to look for the most effective proportion. The p value we have used is 0.5. An example would be:

> **Synonym + Hinting:** *The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! They scored a final goal!* ***Target sentence:*** *They scored a final goal!*
> ***Hint + synonym prompt:*** *(<|subj|> the red team, <|rel|> capable of) (<|subj|> the red squad, <|rel|> capable of,<|syn|>).*

C ParaCOMET Commonsense Inference Example

An example of an input and expected output from the ParaCOMET formulation can be seen below:

> **Model Input:** The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! *They scored a final goal!* <|sent5|> <|xEffect|>
> **Model Target/Output:** win the game

In this example, since the model is predicting ATOMIC objects, the output is a single phrase (i.e., win the game). Additionally, the symbols <|*sent5*|> and <|*xEffect*|> mean that the target sentence is sentence number five[4], and that the relation we want to generate a tuple about is the "has the effect on a certain person(s)" respectively.

D GLUCOSE Commonsense Inference Example

An example of the GLUCOSE formulation's inputs and expected outputs is given below:

> **Model Input:** 1: The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! **They scored a final goal!**
> **Model Target/Output:** The red team scores, Causes/Enables, they win the game ** People_A score, Causes/Enables, People_A win a game

This formulation of contextual commonsense inference is harder than the ParaCOMET one in that it has to generate two sets of a subject, relation, and an object tuples, in which one is the story specific one and the other is the general version of the assertion. These are seen above separated by the ** respectively. In this example additionally, we can see the symbol *1:* which tells the model to predict along a dimension of commonsense described by GLUCOSE (i.e., 1: Event that directly causes or enables X), and the sentence enclosed by asterisks (*) which signifies it is the target sentence.

D.1 An example of Hinting with Synonyms and Antonyms

An example of the synonym/antonym enhanced hinting is the following. Given the following data:

> **Story:** *The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! They scored a final goal!*
>
> **Target sentence:** *They scored a final goal!*
>
> **Target assertion:** *(subject: the red team, relation: are capable of, object: winning the game.)* .

[4]We note that in the original ParaCOMET work, the sentences were 0-start indexed. We utilize 1-start indexing for clearer understanding.

A possible hint given to the model and its corresponding synonym and antonym equivalent is:

> **Hint:** *(<|subj|> the red team, <|rel|> capable of)*
>
> **Synonym Hint:** *(<|subj|> the red* ***squad****, <|rel|> capable of,* ***<|syn|>****)* .
>
> **Antonym Hint:** *(<|subj|> the red* ***individual****, <|rel|> capable of,* ***<|ant|>****)* .

Put together with the rest of the story, the input for the model (with a synonym hint) would be:

> **Story with Synonym Hint:** *The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! They scored a final goal!* ***(<|subj|> the red squad, <|rel|> capable of, <|syn|>)***.

We note that although we provide the example for a synonym, antonyms follow the same process. Additionally, since we sample whether to supply a synonym or antonym, or both, it is possible to give one prompt with all three: hint, synonym hint, and antonym hint.

E Experimental Hyperparameter Configurations

We ran the ParaCOMET experiments for 5 epochs on the dataset's training data and evaluate on the dataset's evaluation data. We utilize a max source sequence length for the T5 models of 256, and a max target length of 128. For the GPT-2 models we utilize a max sequence length of 384. Additionally, we use the ADAM (Kingma and Ba, 2015) optimizer with a learning rate of 2e-5. For the T5 models we utilize a learning rate of 1e-4 because early experiments showed that the model would not converge with lesser learning rates. In both models we use a batch size of 4. We utilize the scripts from (Gabriel et al., 2021) for data generation. The results that we present are the average of the 5 runs over 4 seeds for the conditions.

We ran the GLUCOSE experiments for 5 epochs and 4 seeds on the original GLUCOSE data. We utilize the ADAM optimizer with a learning rate of 3e-4, a batch size of 8, and a max source length of 256 and max target length of 128. In our results, we present the average of the 4 seeds across the 5 epochs. In both experiments we report the scores given by SacreBLEU (Post, 2018), ROUGE (Lin, 2004), and METEOR (Banerjee and Lavie, 2005) using the datasets library (Lhoest et al., 2021) metrics system. We run our experiments in a machine with an AMD ThreadRipper 3970 Pro and with NVIDIA A6000s. Every epoch per model is approximately an hour.

F Future Work

When designing the *hinting* system certain aspects were formulated to leave space for improvements. One such area is finding a smarter way of selecting when to *hint*, and finding a smarter way of selecting what to *hint*. Additionally, more soft prompts could be added to the *hint* such that they would learn a better virtual template.

Another area to explore is providing deeper ablation studies to determine what parts of the *hint* are more effective and when. This work is more a proof-of-concept that *hinting*, or more broadly prompting, is useful towards the task of contextual commonsense inference.

Exploring further, another approach, chain-of-thought prompting (Wei et al., 2022), is commonly utilized in very large language models—a practice that does not extend to the smaller models within our scope. Considering this, we could explore an analogous approach, such as chain-of-hinting, an adaptation that may enable us to incorporate multi-hop graph reasoning for contextual commonsense inference.

Furthermore, given that models trained with *hinting* for contextual commonsense inference can be guided by the information supplied in *hints*, such models can be utilized in a variety of downstream applications such as story understanding and contextual knowledge graph generation.

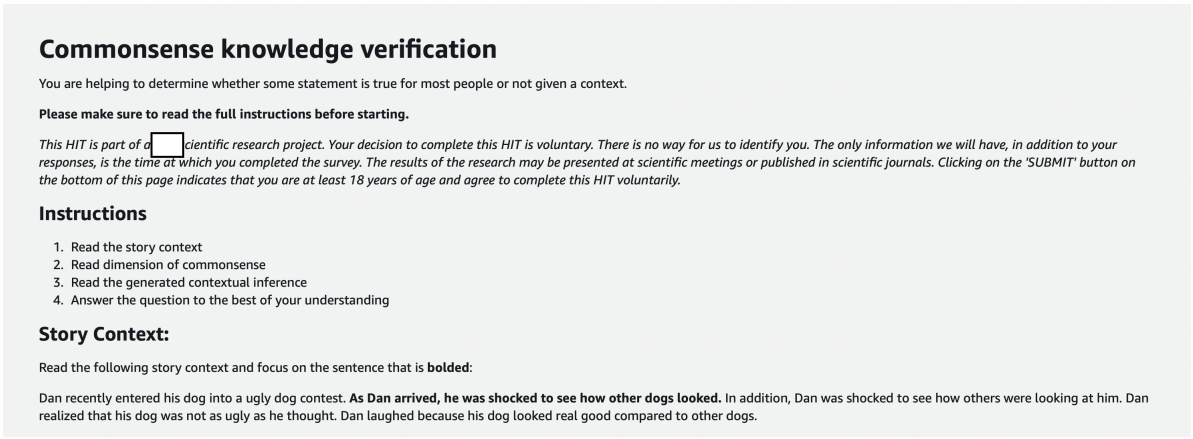

Figure 1: Screenshot of the Mechanical Turk Task

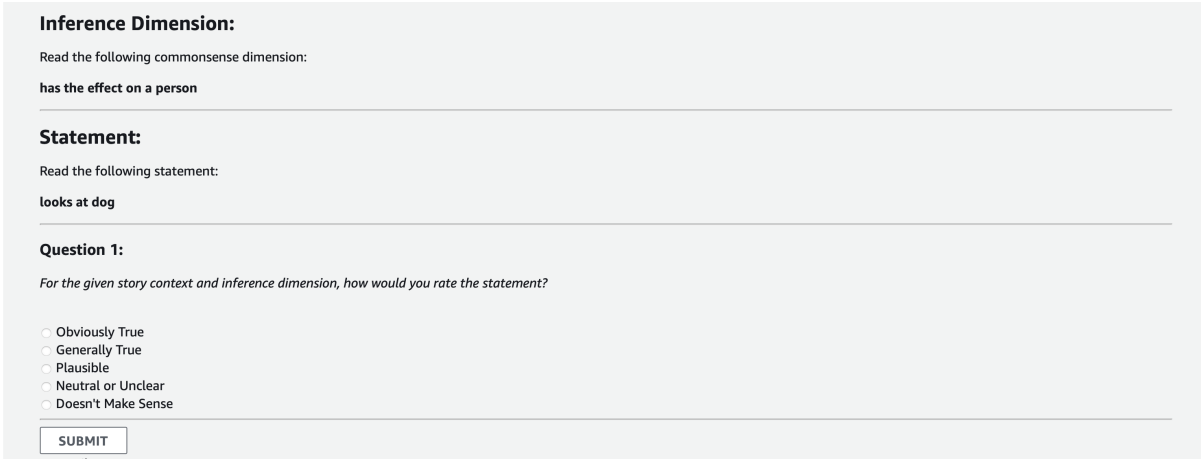

Figure 2: Screenshot of the Mechanical Turk Task pt.2