Adversarial Transformer Language Models for Contextual Commonsense Inference

Pedro Colon-Hernandez [a], Henry Lieberman [b], Yida Xin [c], Claire Yin [b], Cynthia Breazeal [a] and Peter Chin [c]

[a] *Media Lab, Massachusetts Institute of Technology*
[b] *CSAIL, Massachusetts Institute of Technology*
[c] *Department of Computer Science Boston University*

Abstract. Contextualized or discourse aware commonsense inference [1] is the task of generating commonsense assertions (i.e., facts) from a given story, and a sentence from that story. (Here, we think of a story as a sequence of causally-related events and descriptions of situations.) This task is hard, even for modern contextual language models. Some problems with the task are: lack of controllability for topics of the inferred assertions; lack of commonsense knowledge during pre-training; and, possibly, hallucinated or false assertions. The task's goals are to make sure that (1) the generated assertions are plausible as commonsense; and (2) to assure that they are appropriate to the particular context of the story.

We utilize a transformer model as a base inference engine to infer commonsense assertions from a sentence within the context of a story. With our inference engine we address lack of controllability, lack of sufficient commonsense knowledge, and plausibility of assertions through three techniques. We control the inference by introducing a new technique we call "hinting". Hinting is a kind of language model prompting [2], that utilizes both hard prompts (specific words) and soft prompts (virtual learnable templates). This serves as a control signal to advise the language model "what to talk about". Next, we establish a methodology for performing joint inference with multiple commonsense knowledge bases. While in logic, joint inference is just a matter of a conjunction of assertions, joint inference of commonsense requires more care, because it is imprecise and the level of generality is more flexible. You want to be sure that the results "still make sense" for the context. To this end, we align the assertions in three knowledge graphs (ConceptNet [3], ATOMIC2020 [4], and GLUCOSE [5]) with a story and a target sentence, and replace their symbolic assertions with textual versions of them. This combination allows us to train a single model to perform joint inference with multiple knowledge graphs. We show experimental results for the three knowledge graphs on joint inference. Our final contribution is a GAN architecture that generates the contextualized commonsense inference from stories and scores the generated assertions as to their plausibility through a discriminator. The result is an integrated system for contextual commonsense inference in stories, that can controllably generate plausible commonsense assertions, and takes advantage of joint inference between multiple commonsense knowledge bases.

Keywords: Language Models, Adversarial, Commonsense, Joint Inference, Controllable Generation

1. Introduction

Contextualized or discourse aware commonsense inference [1] is a task in which we are given a text context (e.g., story) and a selected sentence from that context, and we have to infer a coherent and contextual commonsense assertion (i.e., fact) from the given context and target sentence. We extend this definition to additionally include story specific assertion inferences (i.e., templates that are instanced by elements from a story), or general assertion inferences (i.e., fact templates) as used in [5]. This framing of a text context and target sentence is important because the text could be a story, a procedure, etc. We define an *assertion* here as a tuple that represents a fact. This tuple contains at least a subject, a relation type, and an object (similar to subject-verb-object triples). We add a field

to this tuple, which is *specificity*. We define *specificity* as whether the assertion's content is about entities in the aligned story, or if it is a generalized version of an assertion. This can be seen as whether the assertion is a *general* template with variables, or a *specific* instance of this template. In the case of a story, contextual commonsense inference can help with story understanding (e.g., a contextual commonsense inference system could infer assertions in a story that indicate a revenge plot [6]), and in the case of a procedure, it could help with step explanations and step rephrasing by giving possibly unstated assertions. Such framing additionally allows us to utilize a trained contextual commonsense inference model downstream by going sentence-by-sentence, inferring assertions as the context changes. We give an example of the task in Figure 1.

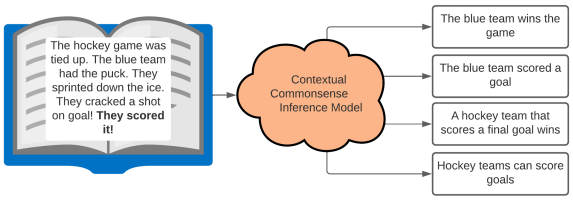

Fig. 1. Overview of the task of contextual commonsense inference. From the story on the left, and the **bolded** sentence, a model should infer assertions such as the ones on the right.

To clarify the task of contextual commonsense inference even further, below we give an example with a story, **a target sentence**, and some corresponding story specific and general commonsense assertion inferences. The story comes directly from the ROCStories corpus [7].

> **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

This task is hard for modern pre-trained contextual language models [1]. This may be because it may rely on information that a model may not have seen during pre-training, or the model has to figure out what topic to infer information about. The first issue is exacerbated because commonsense knowledge, present in everyone and target of a model trained in this task, tends to not be written explicitly in text [8–11]. In addition to these problems, the correctness of the information that is generated by the models is hard to evaluate and usually involves a costly human-in-the-loop setup.

Prior work, such as COMET [12], has tried to do sentence-level commonsense inference: generating a commonsense assertion, with at most a sentence as context. ParaCOMET [1] is an extension of COMET that was developed to work at a paragraph-level (i.e., what we describe as the contextual commonsense inference task). ParaCOMET utilizes a recurrent memory and is trained on a corpus of aligned stories and assertions. ParaCOMET builds this dataset to address the contextual commonsense inference task by aligning facts from a commonsense knowledge graph (i.e., ATOMIC [13]) with a story (i.e., sampled from ROCStories [14]) through a heuristic based on the ROUGE [15] metric. It goes a step further by utilizing the cross entropy of story tokens of a language model, conditioned on one of the matched facts, as a measure of coherence to keep only assertion matches that are coherent to the narrative. They additionally address the need for memory (i.e., for the model to remember prior events) by using and saving prior aligned assertions in a memory system. An example of an input and expected output from ParaCOMET 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[1], and that the relation we want to generate a tuple about is the "has the effect on a certain person(s)" respectively.

Another parallel work that has tackled contextual commonsense inference is GLUCOSE [5]. GLUCOSE annotates the ROCStories [14] corpus along ten dimensions of commonsense. The authors annotate every sentence with an assertion that is either present or implied in it for a given dimension. Additionally, they annotate each assertion with a general version of it, as we defined previously as general inferences, which includes variables and their descriptions. What this means is that any person(s) or object(s) in the assertion is(are) replaced with a token such as *Person_A*, etc. to represent a "general" version of the fact. 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 because it has to generate two sets of assertions a subject, relation, and an object tuples, where 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. With this corpus of annotated stories, the authors train a T5 [16] model to, given a dimension, target sentence, and story, generate both a story specific and general assertion . It is worth noting that in both works, the models have to do their inference from the story, a target sentence, and relation alone.

However, none of these works address controllability in the generation, which means that the models can generate assertions that may be irrelevant to the sentence, or may not be about a topic needed for a downstream application. Additionally, these models are only trained on one dataset at a time, which can hinder a model's capability to infer knowledge if it has not seen the knowledge elsewhere. Lastly, these models do not score the factuality or correctness of the assertions; at most they can generate a beam score, which indicates the likelihood of the generated phrase.

In this work, we attempt to address all of these shortcomings through various techniques. Firstly, we construct a dataset of contextualized assertions consisting of assertions from ConceptNet [3], ATOMIC 2020 [4], and GLUCOSE [5]. To construct this dataset, we align the ROCStories [14] with an assertion by generating sentence/paragraph embeddings for the stories and the assertions by using the sentence-transformers [17] library. We then use cosine distance to find the closest story for each assertion. With this closest story, we repeat the process once more, now with the sentences from the story, to find the closest sentence in the story to the assertion. This contextualization and alignment puts all the knowledge bases in the same contextual universe. Secondly, we augment this dataset of aligned assertions, stories, and target sentences, with "hints", as a method to communicate constraints when performing contextual commonsense inference. We automatically generate "hints" by selecting parts of a target assertion, along with a symbol identifying the parts. Lastly, we use this dataset to adversarially train two language models; one to infer assertions from the story and target sentence, and a second to validate or score the assertion given the story and the target sentence. This adversarial set of models can be utilized in downstream applications for tasks such as knowledge graph construction, and leverages multiple knowledge sources to infer and score commonsense inferences.

Altogether, our contributions in this work are:

- The utilization of a hinting mechanism to help condition and control a generative model for contextual commonsense inference.
- A simple method for contextualizing assertions to a given text with the purpose of performing joint inference.
- A method for adversarially training language models to infer and evaluate assertions from a story context.

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

This work is organized in the following manner. We begin with a Related Work section, and follow it into the Hinting and Joint Inference techniques. Once we have laid out the groundwork for these, we finish with the GAN approach and a Conclusion. Throughout the work, we utilize various vocabulary items and compile these into a Definitions appendix, along with links to a corresponding definition.

2. Related Work

2.1. Prompting

Recently, there has been a shift in Natural Language Processing from pre-training and fine-tuning a model, to pre-training, prompting, and predicting [2]. One reason for this shift is the creation of ever-larger language models, which have become computationally expensive to fine-tune. Prompting is a finding a way to convert a model's input sequence into another sequence that resembles what the model has seen during pre-training. Overall, most prompting research focuses 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) [18] or a soft prompt (i.e., adding trainable vectors that represent, but are not equivalent to, additional words) [2, 19, 20]. 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 (<*subj*>), relation type (<*relation*>), and an object (<*obj*>)), 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 as seen in our example in section 2.1. Hinting is similar to KnowPrompt [21], except that they use a masked language model and soft prompts for relationship extraction. AutoPrompt [18] 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 classify hinting as a hybrid prefix-prompting technique due to the inclusion of trainable symbols that are not part of the model's original vocabulary and can be viewed as soft-prompts given to the model, as well as the combination of these soft-prompts with actual hard prompts to generate a contextual inference. We refer to [2]'s definition of prompting as a three-step process. To begin, we define a function that converts the input to an intermediate template, more precisely in our case by including the hint between parenthesis at the end of the input. Second, the model must generate an answer Z at the end of its input (hence the prefix prompting), which is context-dependent commonsense inference. Finally, this Z is directly mapped to a Y, which corresponds to the target contextual assertion in the contextual commonsense inference task.

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 [22]. The authors supply control signals during pre-training of a language model. This approach is intended to provide a generally applicable 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 [23]. In contrast to GSum, our method of hinting 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 [24] who utilize what they call "control factors" as keywords or phrases that are supplied by a human-in-the-loop to guide a conversation. More similar to our work, but tailored for the task of interactive story generation and without trainable soft-embeddings, is the work by [25] 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 [26] 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. Story and Assertion Alignment

The closest works to ours, with respect to constructing a story aligned assertion dataset, are ParaCOMET and GLUCOSE [1, 5]. GLUCOSE uses human annotation to perform the alignment between stories and commonsense assertions. ParaCOMET takes an automated approach in which assertions are aligned either by giving the sentence to a COMET model as an input and producing a relevant inferred assertion, or by calculating the cross entropy of combining the story up until the target sentence with an assertion from a knowledge base. Our method differs from this in that we utilize cosine distance between semantic representations of the story and its sentences and an assertion from a knowledge base. Some possible differences that arise from this is that our method could match assertions that may not be explicit in a story to that story. Whereas ParaCOMET's approaches, which are based on cross-entropy for coherence, are likely to produce assertions that have parts that are explicit in text. Overall, our approach can match more abstract assertions to stories. Additionally, our method permits us to use the optimized FAISS library to scale up to billions of stories and assertions, and gives us the freedom to select how to embed the stories/sentences/assertions.

2.4. Commonsense: Grounding, Reasoning, and Knowledge

A related line of work has been in grounding commonsense statements for inference. However, this line of work is more aligned with natural language inference rather than assertions. One contribution in this area is HellaSwag [27] which constructs a question answering dataset whose plausible answers are intended to be confounders to language models. Our work differs from this line of work in that we intend to produce structured outputs.

Other work looks at reasoning with commonsense knowledge graphs. One work that utilizes the explicit graph structure to perform multi-hop reasoning is "Commonsense for Generative Multi-Hop Question Answering Tasks" [28]. The authors look to select grounded multi-hop relational commonsense information from ConceptNet via a pointwise mutual information and term-frequency based scoring function to fill in gaps of reasoning between context hops for a model they use. In contrast to this work, we are not explicitly looking at a graph structure for our inference, nor are looking at the task of question answering.

Some older work that looks at doing something similar to joint inference is blending [29]. This technique essentially consists of constructing and adding or blending together matrices of embeddings to find the commonalities between discrete knowledge sources and commonsense knowledge. This method, however, is hard to scale to large knowledge bases, and is not easily applied to the task of contextual knowledge inference. An even older project that looks into a certain kind of joint inference is Cyc [30]. Cyc uses the idea of "micro-theories"; there would be a small set of commonsense assertions that you could reason with, then combine them with the more general Cyc KB. However, this is not really joint inference in the sense that we use, but rather trying to address the problem of local vs. global inference.

We examine also the work by [31] which utilizes formal logic and restructuring of assertions to be able to combine and perform joint inference over knowledge bases, however the system as it is cannot be used for contextual commonsense inference because it requires explicit knowledge to be already present in a knowledge base to make inferences, whereas in our work through the underlying language models can produce inferences for unseen concepts. Additionally, we look at the work [32], which uses a combination of systems to extract high quality non-triple formatted facts from text. However, the system is based on grammatical structures in an input text, which means that implied facts may not be extracted from this if such a system were utilized for contextual commonsense inference.

Other works that have tried to consolidate commonsense knowledge are the following. [33] examines multiple sources of knowledge and unifies the relations in these under 13 dimensions of commonsense, however it still remains a challenge to unify the nodes in the different sources, and such a broad unification may make it challenging to generate inferences for detailed relations (i.e., a specific relation type).

Lastly, we mention some works on open knowledge bases that could be leveraged in future work for utilization in joint inference: TupleKB [34], Quasimodo [35], Ascent [36], GenericsKB [37].

2.5. Adversarial Language Models

Here we look into work that utilizes adversarial or pair training with language models. One such work is [38] in which the authors utilize a GPT-3 [39] model as a teacher in order to distill commonsense knowledge into a student model that is considerably smaller. This task is different from ours in that they do not explore contextual/discourse aware commonsense inference, instead they look at extracting the knowledge already found in a model.

Other work more similar to ours, albeit older, is [40]. In this work, the authors take a similar approach to our adversarial configuration, however they utilize the Wasserstein GAN objective [41] rather than the basic GAN formulation that we use. The authors additionally use the same approximation that we utilize in section 5.2. We note that they employ other strategies such as teacher helping, curriculum learning, and variable length that are worth looking into for future work. We note, however, that the authors tackle general language generation, rather than our task of contextual commonsense inference.

3. Hinting for Controllable Generation

In this section, we propose a technique called "Hinting" to address the lack of controllability in contextual commonsense inference.

3.1. What is hinting?

Recently, there has been work on exploring *prompting* strategies [2] for pre-trained, transformer-based language models [42, 43]. These are methods which alter the input to a language model such that it matches or approximates templates that it has seen during pre-training and can reuse or exploit this information. Prompting helps achieve higher performance in tasks with less training data, can help with controllability in the case of text generation, and is more parameter-efficient and data-efficient than fine-tuning, in some cases [19]. One type of prompting is *prefix prompting* [19, 20]. Prefix prompting consists of altering a language model's input (i.e. prefix) by adding additional words. These words can be explicit hard prompts such as actual phrases or words, or they can be soft prompts, 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.

Prompting holds potential for improving contextualized commonsense inference. We utilize the idea of a *hint*, a hybrid of hard and soft prompts. We define a *hint* as the part(s) of an assertion that a model has to predict, along with special identifiers for these parts, wrapped within parenthesis characters. A forthcoming companion paper focuses on the hinting mechanism. We include a brief description of it here, but we note that the focus of this work is the adversarial generation with the joint inference training. Because hinting is an essential component for controllability of this model, we include this overview.

Syntactically, a *hint* takes the form of: "([specificity], [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 in the target tuple. Hints are provided during training by sampling a binomial distribution ($p = 0.5$) 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 randomly sampling without replacement up to all but one of the elements in a target tuple. In Table 1 we can see various examples of how the hinting mechanism works for a model trained on GLUCOSE.

Here we can see more clearly that whenever we give a hint a model trained with hints (i.e., hinting) tends to produce generations that include the components given in the hint. We utilize hinting in training our models from here on out unless otherwise stated. The controllability that hinting enables can permit us to use models trained with it in downstream applications such as contextual knowledge graph generation.

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: (<|specific|><|subj|>the blue team does not score the final goal)**	**the blue team does not score the final goal** >Causes>the blue team feel(s) disappointment ** Some People_A do not score the final goal >Causes>Some People_A feel(s) disappointment
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|><|obj|>a child)**	they scored a final goal >Causes>**a child** feel(s) happy ** Some People_A scored a final goal >Causes>Someone_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 contextual commonsense inference model with hints. The *hint* is **bolded** and the parts of the *hint* are colored (subject, relation, object). Without a hint we can see that the model tries to infer directly on the content of the sentence, however with *hints*, the model tries to include an inference based on the target sentence with the contents of the *hint*.

3.1.1. Discourse-aware/contextual commonsense inference

Recall that commonsense inference is the task of generating a commonsense assertion. Discourse-aware/contextual commonsense inference is the task of, given a certain context, inferring commonsense assertions that are coherent within the narrative [1]. This task is particularly hard because commonsense knowledge may not be explicitly stated in text [9] 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 [10, 11, 44]. The amount of commonsense knowledge in these models can be improved by supplementing sparsely covered subject areas with structured knowledge sources such as ConceptNet [3, 11]. Knowing that these pre-trained language models may contain some commonsense information has led to the development of knowledge models such as COMET [12]. This line of research has been extended from the sentence-by-sentence level in COMET, to the paragraph-level in ParaCOMET [1]. Contemporaneously, GLUCOSE [5] builds a dataset of commonsense assertions that are contextualized to a set of stories, and generalized (e.g., *John is a human* is generalized to *Someone_A is a human*).

The general task of contextual commonsense inference can be formally described as follows. 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 relation type R. Given all this, we want to generate a tuple in the form of $(specificity, subject, R, object)$ that represents an assertion, present or implied, in S_t given the context S, and the relation type R.

3.1.2. 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), (<|subj|> the red team, <|rel|> capable of), (<|subj|> the red team, <|obj|> winning the game), (<|rel|> capable of, <|obj|> winning the game), (<|obj|> winning the game), (<|rel|> capable of)*

A hint for the given story, target sentence and target assertion, yields the following:

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

Putting everything altogether, 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 generic version of how the hinting mechanism works, and individual datasets (i.e., ParaCOMET and GLUCOSE) have slightly different variations of this.

3.2. Experimental Setup

We run two sets of experiments to show the effectiveness of hinting. The first is utilizing the original ParaCOMET dataset and setup and adding hints. 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. $< |sent0| >, < |sent1| >, \ldots, < |sent(n-1)| >$). 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 along with the relation R, or an object symbol 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 our experiments on the ParaCOMET formulation with the GPT-2 model, we utilize the same cross-entropy loss as in [1]. We note that we utilize a sequence-to-sequence [45] formulation for the T5 and the BART models. 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. For simplicity, we construct the same "heuristic" dataset as ParaCOMET which utilizes a heuristic matching technique to align ATOMIC [13] triples to story sentences.

For our second set of experiments, we utilize the formulation utilized in GLUCOSE [5]. 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 $D \in 1, 2, \ldots 10$[2], 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 assertion, the model has to predict a generalized assertion (i.e., in this task we have to infer a general and context specific subject, object and a relation). For our hints we provide up to five out of these six things, 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 hints between parenthesis.

We run the ParaCOMET experiments for 10 epochs on the dataset's training data and evaluation data. We utilize a max source sequence length for the BART and 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 [46] optimizer with a learning rate of 2e-5, and a linear warm-up of 0.2 percent of the total iterations. 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. We utilize the scripts from [1] for data generation. We also utilize a batch size of 4 for training and we accumulate gradients for 4 steps for an effective batch size of 16. The results that we present are the average of the 10 runs over 4 seeds for hinted and non-hinted conditions.

We run GLUCOSE experiments for 5 epochs and 4 seeds on the original GLUCOSE data. Additionally, we utilize a linear warm-up of 3000 steps. We utilize the ADAM optimizer with a learning rate of 3e-4, a train batch size 4,

[2]The definition for each dimension number is given in the GLUCOSE work. Dimensions in GLUCOSE are (explicit or implicit) relations that help explain causality between the entities mentioned.

with gradient accumulation of 4 steps for an effective batch size of 16, 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 [47], ROUGE [15], and METEOR [48] using the datasets library [49] metrics system. We run our experiments in a machine with an AMD ThreadRipper 3970 Pro and 4 NVIDIA A6000s. Every epoch per model is approximately an hour.

Additionally, we run a small Mechanical Turk study similar to the one presented in the original ParaCOMET [1] in which a human judges a generated assertion and judges the plausibility of it on a 5-point Likert scale: obviously true (5), generally true (4), plausible (3), neutral or unclear (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 sample from each of the ParaCOMET and GLUCOSE test sets, 100 entries. Then based on the models for each dataset, we pick the epoch that had the highest automated scores and we proceed to randomly sample one of the trained hint and non-hinted models. We then select one sentence of the randomly sampled test entries and ask both models to generate an inference along a randomly sampled relation or dimension for that sentence.

3.3. Results and Effects of hinting

Model	BLEU		METEOR		ROUGE1		ROUGE2		ROUGE L		ROUGE L SUM	
	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint
ParaCOMET	**42.705***	41.960	**59.411***	59.045	**63.339***	61.454	**52.483***	50.513	**63.292***	61.395	**63.294***	61.399
Bart	**41.765***	41.639	**58.766***	58.639	**61.054**	61.013	**49.970**	49.889	**61.004**	60.964	**61.010**	60.969
T5	41.070	**41.102**	**58.004**	58.000	59.535	**59.631**	48.695	**48.823**	59.488	**59.588**	59.494	**59.597**

Table 2

Averages of 4 different seeds over 10 epochs for *hinted* (Hint) and non-hinted (No Hint) runs of the ParaCOMET dataset from [1]. The largest scores are **bolded** and significantly different scores 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, and do not decrease performance significantly. For significance we use the t-Test: Paired Two Sample for Means.

BLEU		Meteor		Rouge 1		Rouge 2		ROUGE L		Rouge LSUM	
No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint	No Hint	Hint
58.542	**59.099***	66.829	**66.917**	66.387	**66.681***	47.850	**48.141**	62.542	**62.874***	62.528	**62.868***

Table 3

Averages of 4 different seeds over 5 epochs for *hinted* (Hint) and non-hinted (No Hint) runs of the GLUCOSE contextual inference task dataset. This is the same dataset as the work in [5] The largest scores are **bolded** and significantly different scores have an asterisk (*) next to them. Once more, we see that hinting provides a small, increase in performance, all the while permitting controllability. t-Test: Paired Two Sample for Means

Model	Non-Hinted	Hinted
ParaCOMET	3.71	**3.76**
Bart	**3.72**	3.48
T5	**3.73**	3.68
T5-GLUCOSE	**4.10**	4.06

Model	Non-Hinted	Hinted
ParaCOMET	81%	**84%**
Bart	**83%**	74%
T5	**81%**	**81%**
T5-GLUCOSE	**92%**	90%

Table 4

Results of human evaluation of ParaCOMET and GLUCOSE datasets. The largest scores are **bolded** and significantly different scores have an asterisk (*) next to them. We sampled 100 test points for each model from their test datasets and had the hinted and non-hinted models infer assertions. Humans judged these assertions on a 5 point Likert scale where above 3 was plausible similar to [1]. On the left we can see the average values of the human judgments and on the right we can see the percentage of plausible inferences (rated >= 3). We can see that hinting provides comparable performance.

3.3.1. Experiment 1: ParaCOMET with hints

The aggregated (averaged) 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 *only* adds the object of the assertion 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 than that of the other models, 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.

3.3.2. 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 does tend to improve the performance of the contextual commonsense inference task. This suggests that hinting is indeed beneficial for the task, especially when faced with the harder task of generating both a general and specific 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, but further experimentation would be needed to verify this.

3.3.3. Experiment 3: Human Judgements

The results for a small Mechanical Turk study for human evaluation of model inferences can be seen in Table 4. Overall, we can see here that hinted systems are judged as less plausible. Interestingly, after inspecting the results where there was a large difference (more than two points between the systems), we see that there are some cases in which the same or very similar responses got completely different scores. 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”).

3.3.4. Final Remarks on Hinting

From the results of our experiments, we can see that hinting tends to increase the performance of contextualized commonsense inference at least with regard to automated metrics and does not significantly degrade or improve human judgements. Without any significant cost, by utilizing hinting, we gain controllability in the generation. By supplying these *hints*, we are teaching the model to 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.

4. Joint Inference of Commonsense Assertions

Given that we have presented hinting, which is a method to help with the controllability of contextual commonsense inference generation, we now look at how we can combine multiple knowledge bases for this task along with hinting for contextual commonsense inference.

4.1. What is joint commonsense inference?

In this work, we define *Joint commonsense inference* as inferring commonsense knowledge assertions by leveraging knowledge from multiple knowledge bases. To illustrate this, we give the following example, story:

> *John is a regular person who has a dog.* ***John, every day, goes out to walk his dog.*** *John met a friend when walking his dog. They exchanged stories about their dogs.*

From this story, we want to infer the general version of the commonsense assertion of "John is capable of walking his dog", derived from **the second sentence**. This general version can look similar to "Someone_A who has an animal (that is a dog) enables Someone_A to walk the animal (that is a dog)". To generalize this, we must know that: *John is a person's name*, which we can find from a semantic tagger. A much more commonsensical fact needed to infer the assertion is that: *a dog is an animal*, which is a fact found in ConceptNet. Lastly, to infer the assertion, we need to know that: *A person having a dog has the effect that a person goes to walk their dog*, which is a fact that we could find from ATOMIC 2020. Therefore, to infer the general assertion that we presented, we must join information from *at least* two knowledge bases to infer our general assertion. This process of joining the information from multiple sources is what we call joint commonsense inference. Joint commonsense inference is useful because it could lead to implicitly applying or combining knowledge and/or analogies that might be present in the different knowledge sources, which may lead to better results when performing contextual commonsense inference.

4.2. Joint Inference Approach Overview

To perform joint inference in the task of contextual commonsense inference, we propose the following approach:

1. *For each knowledge base that we have, we convert each of the assertion found in them into a tuple format of* ***{subject, relation, object, specificity}***[3]*. We note that each part of the tuple must be text*[4] *(i.e., if a relation is symbolically "IsA", the text version would be "is a").*
2. *We align each knowledge base tuple with a story (e.g., the ROCStories corpus) and target sentence from the story. The target sentence is the sentence which is most likely to be used to infer the tuple. We perform the alignment by vectorizing the tuples and stories and utilizing nearest neighbors with the cosine distance as a metric. We give details of this alignment in section 4.4.*
3. *We combine into one list and shuffle the aligned knowledge base tuples from multiple knowledge sources.*
4. *We replace the naming scheme of variables that may be present in general assertions with the naming scheme from GLUCOSE (e.g., PersonX is replaced with Person_A).*
5. *We train a contextual commonsense inference model on this dataset, whose inferences are joint inferences.*

By following this procedure, we will end up with a dataset of story aligned assertions. In this dataset, all of the assertions are grounded in the same set of stories. With this we can train models that can perform joint contextual commonsense inference. Now we will go into some details of this process.

4.3. Specificity in assertions

Recall that we define *specificity* as whether an assertion's content is about specific entities in the aligned story, or if it is a generalized version of an assertion. This can be seen as whether the assertion is a *general* template with variables, or a *specific* instance of this template. To make the difference between *specific* and *general* assertions clearer, we give the following example. Using the same story as before:

> *John is a regular person who has a dog.* ***John, every day, goes out to walk his dog.*** *John met a friend when walking his dog. They exchanged stories about their dogs.*

As before, we focus on the second sentence: **John, every day, goes out to walk his dog.**. From here, we can infer the *specific* assertion: "John is capable of walking his dog". The assertion is *specific* because it fills out a broadly applicable template, which we will present next, that speaks about John and his dog from the story. From the sentence, we can also infer the *general* version of the assertion: "Someone_A who has Something_A (that is a dog) enables Someone_A to walk the Something_A (that is a dog)". This latter assertion is *general* because it speaks in

[3] This follows a similar pattern to subject-verb-object triples, but has the added field of specificity which is whether the assertion is contextual to a story, or a generally applicable assertion

[4] This ultimately helps us express the assertion in a textual way (i.e., (a dog, IsA, animal) when converted to the tuple (a dog, is a, animal, specific) and passed to a string representation function can be expressed as "Specifically, a dog is a animal".

a template format (i.e., broader terms) the same fact. A *general* assertion is not the story-dependent instance of the template, but the broader, story-independent template. These *general* assertions contain variables in them.

In this work we utilize ConceptNet, ATOMIC 2020, and GLUCOSE as our knowledge bases, and propose to combine them to perform joint inference. In table 5 we give the different available specificities for these knowledge bases. From this, we can see that ConceptNet does not have *general* specificity assertions. Although this may

Knowledge Base	General	Specific
ConceptNet	✗*	✓
ATOMIC 2020	✓	✗*
GLUCOSE	✓	✓

Table 5

Here we can see the available specificities in ConceptNet, ATOMIC 2020, and GLUCOSE. We mark with ✗ the specificities that are not available by default, and add * to those that can be generated.

sound counterintuitive, ConceptNet gives *specific*, untemplated, instances of assertions, in contrast to ATOMIC and GLUCOSE, which describe *general* versions of assertions. ATOMIC 2020 has the opposite problem, it gives *general* versions of rules, (e.g., PersonX participates in some event, has some effect on PersonX or Y around them), and does not give, within our contextual commonsense inference framework, the *specific* instance of the templates (e.g., filling out PersonX, PersonY, etc.). To remedy this lack of specificity within two of our sources, we mention ways to generate examples of the missing specificity, and implement the solution for ATOMIC 2020.

4.3.1. Generating Missing Specificity

ConceptNet To generate *general* assertions for ConceptNet, we could possibly run a classifier that would determine whether a given set of tokens is a, person, place, object, among others. With this information we could fill out, as an example, the template that GLUCOSE broadly utilize which is: {Category}({Description}), relation, {Possibly Other Category} ({Possibly Other Description}). From ConceptNet, we could find the relation: "a dog, IsA, animal". A *general* version of this assertion can be "Something_A (that is a dog), IsA, Something_B (that is a animal)". Although we describe this process, we do not implement it in our work.

ATOMIC 2020 To generate *specific* assertions for ATOMIC 2020 we can do the following. We can first identify variables (PersonX, PersonY, etc.) that are in the assertion. We can then replace these variables with a mask token from a language model that was trained with the masked language modeling objective [43, 50], and use the language model to fill in the Mask token similar to a cloze (i.e., fill-in-the-blanks) task. To give the model sufficient context, we insert the assertion to the right of the nearest aligned sentence (we describe the process to get this in the next section). In the case of Person*N* variable, this usually leads to the variable being replaced with a character from the story. In addition to this, ATOMIC 2020 contains blanks demarcated by underscore characters (i.e., ____), which we can once more replace with a mask token and have the model fill it out with the given context. We use this process in our work, filling in the blanks with a ROBERTA [50] large model.

4.4. Aligning Assertions with Stories

To align assertions with stories, we do of the following procedure. On a high-level, we vectorize the stories, and we vectorize the assertions and we then utilize the cosine distance to find the nearest story for each assertion. We then go into more detail and repeat the same procedure (i.e., vectorization and similarity search) for each sentence in the previously found nearest story. Ultimately, we are left with the story and sentence that is most relevant or similar to the assertion. On a low-level, we utilize the sentence transformers package along with the "paraphrase-mpnet-base-v2" model from the repository, to generate a representative vector for every story and for every assertion from each of our knowledge bases. We then utilize the FAISS package [51] to perform a fast approximate cosine similarity search to find, for each assertion, what is the nearest story. Once we have this nearest story, we again utilize the sentence transformers model to vectorize every sentence in that story along with the FAISS package for the cosine similarity search, to find the nearest sentence to the assertion. This process can be visualized and figures 2 and 3.

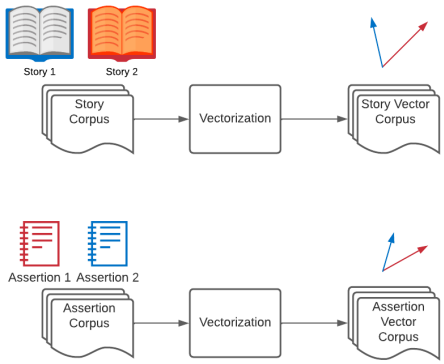

Fig. 2. Step 1: The story and assertion corpus are vectorized. In our work we utilize the sentence-transformers package [17] to achieve this.

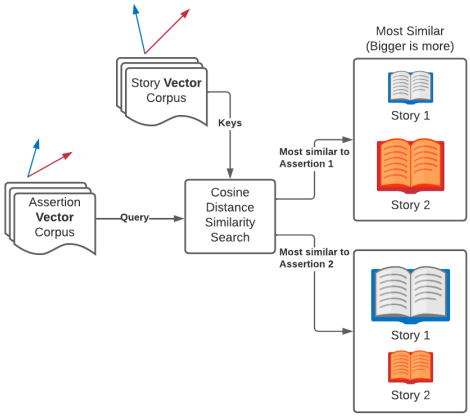

Fig. 3. Step 2: The resulting assertion vectors are utilized as queries, and the resulting story vectors are used as keys for a memory-like lookup. In this work we use the FAISS package for this. The output of the memory-like lookup is the nearest story for each vector. This process is repeated for the sentences in the nearest story, to align the assertion with a sentence.

4.5. *Experimental Setup*

To evaluate the effects of joint inference by combining multiple knowledge bases in the task of contextual commonsense inference we do the following. We generate a story aligned assertion dataset for each knowledge base individually (i.e., for ConceptNet, for ATOMIC 2020, and for GLUCOSE) as described in the previous sections. Once we have generated a dataset for each, we proceed to perform combinations of the datasets: ConceptNet-ATOMIC 2020, ConceptNet-GLUCOSE, GLUCOSE-ATOMIC, ConceptNet-ATOMIC-GLUCOSE. For the individual and the combined datasets we perform three sets of automated tests. One that includes hinting the specificity, subject, and relation during evaluation, one that includes hinting the subject during evaluation, and the other without these. The rationale behind these setups is that we want to evaluate what the model infers without any guidance, and see what it infers with varying levels of guidance with multiple knowledge sources. To train our models, we use a batch size of 50, on 4x NVIDIA A6000, a learning rate of 1e-5 for an ADAM [46] optimizer, and 3 epochs over the data.

We note that the data for ConceptNet we utilize is the dataset given by [52], specifically the data in the "train_600k.txt" which are approximately 600,000 examples of assertions from ConceptNet, and as a test set we utilize the "test.txt" that they provide. For ATOMIC 2020 we utilize the training and testing data files provided by the authors [4]. Lastly, for GLUCOSE we use the training and evaluation files also provided by the authors in the corresponding repository.

Additionally, we look into running a small Mechanical Turk evaluation of generated test assertions, because we suspect that automated metrics may hurt the model's evaluation when not using hinting. We sample 100 entries from the testing files of each knowledge base (ATOMIC 2020, ConceptNet, and GLUCOSE), and run these through a set of models trained firstly with only one of the test knowledge bases (i.e. a model trained only ConceptNet, a model trained in ATOMIC 2020, and a model trained in GLUCOSE) and secondly a model trained with the combination of knowledge bases and evaluated with and without hinting. We take the generated inferences and ask 2 raters from Amazon Mechanical Turk to determine whether the assertion is acceptable, whether it is acceptable with the context that it was aligned with, and whether the gold standard assertion was acceptably aligned with the context. We mark as acceptable the answers that both human annotators agree as valid and the others as invalid.

4.6. Effects of joint inference

The results for our automated experiments can be found in Table 6 and from our human experiments in 7. From our experiments in this area, we notice the following. Firstly, when used with hinting, joint inference *does not* seem to improve the performance of synthetic tests. What this may mean is that hinting manages to utilize the format (e.g., relation types) of the knowledge base that it is tapping into for information. Additionally, some of the knowledge sources that we are using have a little overlap (GLUCOSE and ConceptNet had approximately 0.34% of overlap [5], and ATOMIC 2020 has approximately 9.4% of overlap with ConceptNet [4]), which means that once hinting is utilized to give control signals to the models, this lack of overlap may attribute to why the metrics do not decrease drastically. Secondly, without hinting, in the automated tests that we run, performance seems to degrade when we add knowledge bases. Upon further inspection of the results, the reason for seems to be that the model thinks that an assertion in the format of another knowledge base (e.g., generalized assertion from GLUCOSE on the ConceptNet

Training Set(s)	Test Set	Hint	BLEU	METEOR	ROUGE
ATOMIC 2020, ConceptNet, GLUCOSE	ATOMIC 2020	No	51.114	52.224	52.681
ATOMIC 2020, ConceptNet	ATOMIC 2020	No	51.164	52.151	52.713
ATOMIC 2020	ATOMIC 2020	No	51.139	52.699	52.904
ATOMIC 2020, ConceptNet, GLUCOSE	ATOMIC 2020	Subject	79.89	80.956	82.927
ATOMIC 2020, ConceptNet	ATOMIC 2020	Subject	80.046	81.144	83.087
ATOMIC 2020	ATOMIC 2020	Subject	80.203	81.221	83.079
ATOMIC 2020, ConceptNet, GLUCOSE	ATOMIC 2020	Subject, Specificity, Relation	87.031	88.172	89.606
ATOMIC 2020, ConceptNet	ATOMIC 2020	Subject, Specificity, Relation	87.091	88.242	89.645
ATOMIC 2020	ATOMIC 2020	Subject, Specificity, Relation	87.095	88.226	89.621
ATOMIC 2020, ConceptNet, GLUCOSE	ConceptNet	No	51.892	58.803	60.302
ATOMIC 2020, ConceptNet	ConceptNet	No	56.136	62.114	63.532
ConceptNet, GLUCOSE	ConceptNet	No	59.285	63.685	65.65
ConceptNet	ConceptNet	No	60.63	64.653	66.46
ATOMIC 2020, ConceptNet, GLUCOSE	ConceptNet	Subject	76.404	79.484	78.413
ATOMIC 2020, ConceptNet	ConceptNet	Subject	76.7	79.614	78.617
ConceptNet, GLUCOSE	ConceptNet	Subject	76.014	78.664	77.841
ConceptNet	ConceptNet	Subject	76.635	79.108	78.314
ATOMIC 2020, ConceptNet, GLUCOSE	ConceptNet	Subject, Specificity, Relation	92.695	94.253	94.171
ATOMIC 2020, ConceptNet	ConceptNet	Subject, Specificity, Relation	92.892	94.286	94.378
ConceptNet, GLUCOSE	ConceptNet	Subject, Specificity, Relation	92.729	94.071	94.109
ConceptNet	ConceptNet	Subject, Specificity, Relation	92.77	94.159	94.232
ATOMIC 2020, ConceptNet, GLUCOSE	GLUCOSE	No	36.23	41.338	48.629
ConceptNet, GLUCOSE	GLUCOSE	No	37.823	41.997	49.577
GLUCOSE	GLUCOSE	No	42.51	47.186	53.856
ATOMIC 2020, ConceptNet, GLUCOSE	GLUCOSE	Subject	80.879	82.014	85.664
ConceptNet, GLUCOSE	GLUCOSE	Subject	81.349	82.775	85.926
GLUCOSE	GLUCOSE	Subject	80.928	82.076	85.681
ATOMIC 2020, ConceptNet, GLUCOSE	GLUCOSE	Subject, Specificity, Relation	85.721	87.433	90.04
ConceptNet, GLUCOSE	GLUCOSE	Subject, Specificity, Relation	85.72	87.518	90.034
GLUCOSE	GLUCOSE	Subject, Specificity, Relation	85.65	87.473	89.967

Table 6

Here we present the results of our joint inference tests. We color code sets of rows as testing run on ATOMIC 2020, GLUCOSE, and ConceptNet. The Training Set(s) column contains the knowledge bases that were used to train the model. The Test set column contains which knowledge base test set was used to evaluate the models. The Hint column represents the Hints that were given to the model during testing. Overall, we can see that with hinting on the test set (i.e., hinting the subject or the subject and the relation type), the addition of knowledge bases for inference does not improve nor degrade substantially the performance. To view this in the table, we can compare the rows in which the Hint column is either "Subject" or "Subject, Relation, Specificity". Additionally, we can see that without hinting on the test set (i.e., rows that Hint is "No"), the addition of knowledge bases for inference tends to decrease the performance.

Model	Acceptable	Contextually Acceptable	Alignment Acceptable	Acceptable	Contextually Acceptable	Alignment Acceptable	Acceptable	Contextually Acceptable	Alignment Acceptable
ATOMIC 2020 - No Hint	0.70	0.66	0.6	-	-	-		-	-
ConceptNet - No Hint	-	-	-	0.77	**0.71**	**0.72**	-	-	-
GLUCOSE - No Hint	-	-	-	-	-	-	**0.81**	**0.68**	**0.8**
ATOMIC 2020 - ConceptNet-GLUCOSE - No Hint	**0.76**	**0.68**	**0.63**	**0.83**	0.7	0.65	0.79	0.67	0.59
ATOMIC 2020 - ConceptNet-GLUCOSE - Hint	0.71	0.53	0.57	0.77	0.64	0.68	0.77	0.64	0.68

Table 7

Results for human annotation of 100 randomly sampled assertions from ATOMIC 2020, GLUCOSE, and ConceptNet test sets and the inferred commonsense from these. We have three sets of three columns Acceptable, Contextually Acceptable, and Alignment Acceptable. Each set of columns is color-coded to represent a knowledge base. Firstly, the Acceptable column is the ratio of whether humans thought that inferred assertions, without context, were acceptably commonsense or not. The Contextually Acceptable, column represents the ratio of whether humans thought that inferred assertions given the context, were acceptable or not. Lastly, the Alignment Acceptable column is whether humans thought that the gold standard (from a knowledge base) assertion was correctly matched to the context. We can see that without hinting, the joint inference model (i.e ATOMIC 2020 - ConceptNet - GLUCOSE - No Hint) improves the acceptability, both with and without context, of assertions predicted in the ATOMIC 2020 test set. We can also see that performance does not degrade much in whether it produces assertions that are contextually acceptable throughout the test sets. We can see that with hinting, however, the performance is decreased and becomes closer to what the individually trained models can achieve. This suggests that with hinting, the model tries to channel the knowledge base that we are targetting, and aligns to what we see in the automated metrics.

test set, relation type from ATOMIC in the GLUCOSE test set) may be more relevant than the test assertion types that we are evaluating. Upon inspecting some inspections of test results, this seems to be the case. We give one such example now, where a model that was trained on all the knowledge bases (ATOMIC 2020, ConceptNet, GLUCOSE) has to predict assertions for a test set from ConceptNet:

> **Model Input: Someone I went to school with had a funny laugh.** He would chuckle. Then he would make a whew sound. He had not realized it before. He could not stop his crazy laugh.
>
> **Expected Output:** <general> <subject> person <relation> is/are capable of <object> laugh at joke
>
> **Model Output:** <general> <subject> Someone I went to school with had a funny laugh. <relation> makes others react <object> entertained
>
> **Model Output with Subject, Relation, Specificity Hint (<general> <subject> person <relation> is/are capable of):** <general> <subject> person <relation> desires <object> laugh at joke

The relation "makes others react" is not from ConceptNet, but from ATOMIC 2020. If we hint the model the subject, relation type, and specificity, it produces a result similar to what we are expecting, where it only defers to using the "desires" relation for the contextual inference. Which, in this case, would not be an incorrect inference.

Following this, we look at the results for our Mechanical Turk study, can be found in table 7. We can see that without hinting, the joint inference model (i.e. ATOMIC 2020 - ConceptNet - GLUCOSE - No Hint) improves the acceptability, both with and without context, of assertions predicted in the ATOMIC 2020 test set. We can also see that performance does not degrade much in whether it produces assertions that are contextually acceptable throughout the test sets. We can see that with hinting, however, the performance decreases and becomes closer to what the individually trained models can achieve. This suggests that with hinting, the model tries to channel the knowledge base that we are targeting, and this aligns to what we see in the automated metrics. This is reinforced by the test example given previously, and similar examples can be found for the different test sets.

Now, taking these results together, when we use hinting and join our multiple knowledge sources to perform this joint inference, we are able to within one model, essentially fit all the knowledge bases, that we are evaluating, at hardly any loss in plausibility/acceptability, or at the cost of automated metrics. This has implications for down-stream applications because they no longer require multiple models. With one model and hinting we can do what three separate models would do.

Lastly, we also note that on average our alignment technique has 60% approval rate for ATOMIC 2020, 68.3% for ConceptNet, and 69% for GLUCOSE, which gives us on average 65% approval for our alignment strategy of using sentence-transformers with the FAISS similarity search.

5. Adversarial Language Models

5.1. Adversarially training language models

In this work, our main contribution is providing and demonstrating the usefulness of a method for adversarially training language models for the task of contextual commonsense inference. In the broader literature of generative adversarial networks (GANs) [53], the adversarial training of models, tends to lead to better results than training each model individually, possibly because of the gradients flowing from the discriminator informing the generator on how to improve. Additionally, the discriminative model can give a measure (usually in a 0-1 range) of how good generations are. In this work, we propose using the generative language model (i.e., a generator) that we train for contextual commonsense inference and combine it with a discriminative model (i.e., discriminator) whose inputs are the same story and target sentence along with the generator's inference. The discriminator can give a measure of how good the quality of the generated inference is for the given context. This general architecture can be seen in Figure 4.[5]

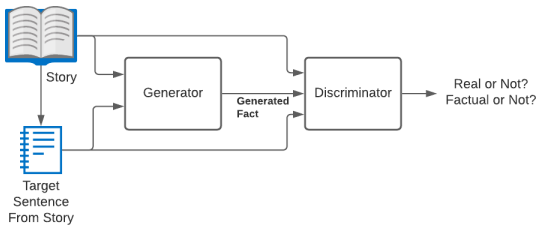

Fig. 4. Overview of the proposed GAN architecture. A story and a target sentence are fed into the generator, which infers a contextual commonsense fact. This fact, along with the story and target sentence, are passed into a discriminator to determine whether it is from a generator or not and whether it is factual or not.

To be able to achieve this architecture, we need to be able to connect a generative language model to a regular language model with some additional final layers that produce a score. In our work, we utilize a transformer-based encoder-decoder generative model. Specifically, we use the BART model [55]for conditional generation, provided by the Huggingface Transformers library [56]. For our discriminator, we utilize a regular BART for sequence classification model also from Huggingface Transformers. However, it is not as simple as conditionally generating, and passing into the discriminator the generated assertions. The generative process utilized is a recurrent next token generation. Recall that to pick a next token with this method, a non-differentiable *argmax* operation is used. This impedes the gradients from being calculated in backpropagation. The issue becomes even more complex, in that the generation process can utilize beam search to find even better generations, and each beam at the end of each generation step selects a next best token also with an *argmax*. To address this discontinuity, we utilize an approximation of the *argmax* (i.e., a soft *argmax*) described in the next section and similar to the work described in Section 2.5, and perform a dot operation on the scores from this soft-*argmax* with the embeddings from the embedding layer to get an approximate and differentiable input embedding for the discriminator. Finally, we pick two of the same types of base model (e.g., BART), in order for both the generator and discriminator to share a vocabulary. The reason for sharing the vocabulary is addressed in the next section, however this may not be necessary, and we give an alternative way of being able to "splice" together different models for this task in section 5.4.

[5]One interesting aspect of this formulation, is that it becomes a kind of conditional GAN [54], which could reinforce the control signals given in hints.

We give some formal notations to describe the GAN framework. Let G be a learnable function (implemented as a Generative Neural model) that can take an input from a domain X and convert it to an output in another domain Y, namely $G : X \rightarrow Y$. That output Y is evaluated by a learnable function D that scores the output Y. A generative adversarial network (GAN) is an interplay between G and D, in which G tries to minimize the difference between what it generates, and D tries to maximize its discrimination of fake generations [53]. We note that we are not the first to attempt utilizing (GAN) systems for text generation as seen in Section 2.5, but we are the first to apply this system for the task at hand.

5.2. *Addressing the Discontinuity in Generation*

Recall that during recurrent conditional language generation, a next token, N, is selected by finding the *argmax* of a $softmax$ of all the vocabulary, after a language model is given the generated phrase up until step $N-1$. Also recall that an embedding layer is a neural network component that given an index i , returns a row vector, from a vocabulary matrix, that corresponds to i. This lookup operation can also be achieved by performing a dot product of a one hot vector that represents the index i and the vocabulary matrix. This essentially scales every row in the matrix by the corresponding vector component and sums all the vectors. In the case of a one-hot vector, it scales all but one vector to zero, therefore leaving only the desired i at the end of the summation.

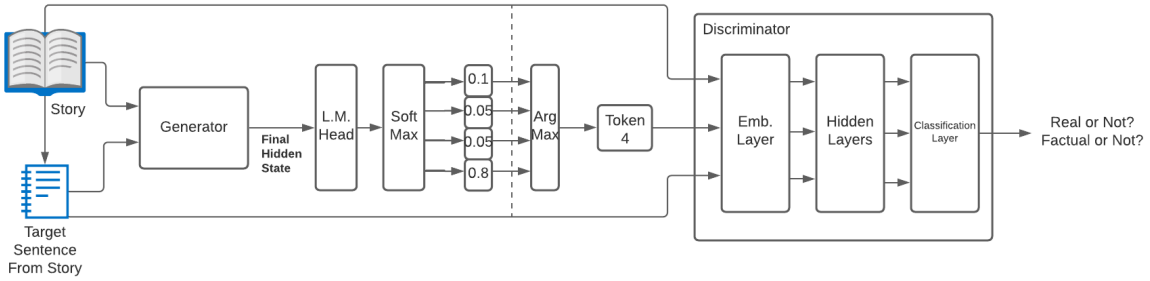

Fig. 5. We visualize an example that shows the discontinuity when combining a generative language model with a discriminative language model. The dashed line represents where the gradients are discontinued because of the non-differentiable *argmax* operation.

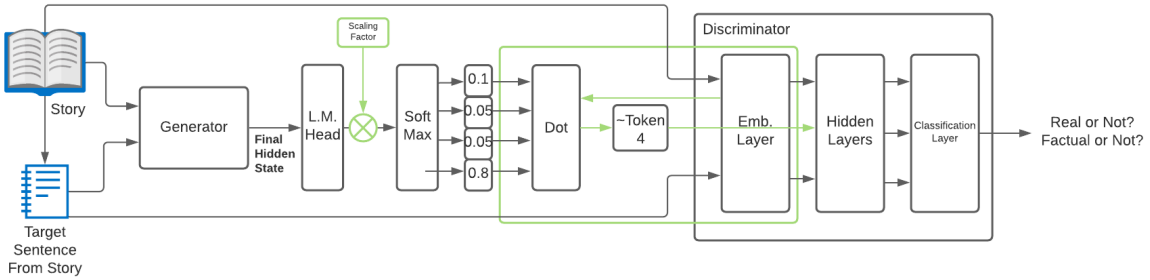

Fig. 6. We visualize an example that shows how we address the discontinuity by replacing the non-differentiable *argmax* with a dot product between the softmax and the embedding layer matrix. Additionally we highlight where the scaling factor is inserted to make the approximation more accurate. We mark our approach in green.

Now, to maintain the gradients, we need to connect the output of our generative model, which is the softmax, to the embedding layer of our discriminator model so that it can be input, scored, and backpropagated correctly. To do this, we simply replace the aforementioned one-hot vector that represents our index, with the softmax that the generative model produces at a given generation step, and perform a dot product of this softmax with the embedding matrix. This method is approximate, given that there may be noise from other non-zero elements in the softmax, and the top element is not an exact 1. To somewhat remedy this approximation, we can multiply the input of the softmax

by a certain factor to essentially give a more approximate one-hot vector. However, this factor cannot be very large, because it may cause instability during backpropagation. In our work, we use a scaling factor of 1, as this seems to be accurate enough. We repeat this approximation for every generation step, and are left with a list of input embeddings for the discriminator that represent the output of the generator with usable gradients. Since we are using the same vocabulary, we can verify how accurate the output is, by training a K-Nearest Neighbors system, and finding the K=1 neighbor of the output of the softmax and embedding matrix dot product against the embedding matrix. We can use this test to determine an appropriate factor for scaling the softmax. Altogether, this approximation permits us to train our two language models adversarially by having gradients flow from the discriminator to the generator. We note that there is work that utilizes a similar simplification to permit gradient flow in an adversarial system in Section 2.5.

5.3. Addressing Different Generation Types

The aforementioned approximation for the discontinuity, as we described it, can be utilized for greedy selection of the next-token (i.e., we always pick the maximal one from the final softmax). We can also apply this technique to beam-search generation, and at every point in constructing the top scored beam, we utilize the softmax of the maximal scoring beam, essentially simplifying the problem back down to a greedy generation-like formulation. In this work however, we do not explore top-k generation, top-p generation, nor sampling during generation. Top-k and top-p generation can be seen as masking out with zeros, tokens that do not meet a certain criteria. Sampling is more complicated. To use our approximation with sampling, we would need to model the sampling function at every generation step with something like a recurrent neural network. We leave this line of research for future work.

5.4. Splicing different models

We come back to address the issue of having to utilize models that have the same vocabulary. The reason for this is that the soft argmax operation matches in matrix multiplication dimensions between models. We now give an alternative, although unexplored, option. Given that a generative model will produce a softmax vector of vocabulary size V, and we have another model that has a different vocabulary size of M, we can train decoding layers that can convert the output tokens of the generative model, into corresponding tokens from the discriminative model. However, this conversion layer would need to be trained beforehand, and may need to be frozen during the adversarial training, otherwise it would be a disconnect between the two models, and the input given to the discriminator may be corrupted. To train this conversion layer beforehand, one could use as a ground truth, the results that a tokenizer from model B, with the vocabulary size of M, would use as the targets in a cross entropy loss, and the results that a tokenizer from model A, with the vocabulary size of V, uses as the input to the layer.

5.5. Factuality in the Discriminator

Given that we can now adversarially train our models, we explore enhancing the discriminator with some way to determine factuality. We take a simple approach that in addition to the normal discriminator training objective (i.e., the discriminator is given a batch of generated text and a batch of real text and evaluated whether it inferred this correctly), we add a confounder loss. Our additional confounder loss is based on the confounder loss by [52], in that we shuffle around the subjects and objects and expect our model to determine that when shuffled objects are false. Since our generated outputs are structured (i.e., we have symbol tokens that delimit the different parts of assertions), we can do this shuffling easily. Although shuffling may incur in some false negatives (we may have a shuffled configuration that is factually correct), since we supply the story and target sentence, we expect the discriminator to be able to discern this correctly. We believe that we could also apply the max-margin loss utilized to great effectiveness by other language GAN literature [57, 58], although we leave this for future work.

5.6. Experimental Setup

For our GAN experiments, we had the following setup. We built a joint inference dataset using the procedure given in section 4, and we augmented it with hinting. Hinting was done in the same manner as in Section 3.1,

by sampling a binomial distribution ($p = 0.5$) and if the sample is true we provide a hint by randomly sampling parts of the target assertion. This dataset is composed of 1,479,811 training examples, and 30,183 testing examples. We fed this data to a model with the adversarial setup described in the previous sections. The generator model utilized was the BART-base for conditional generation, and the discriminator model was a BART-base model for sequence classification. For the sake of time, we run our model on only 100,000 examples, with a batch size of 32, on 3xNVIDIA A6000 for 3 epochs. In addition to this, to see the effects of the adversarial formulation and of the confounder loss, we train a model without the adversarial approach that we propose (a separated Generator and Discriminator with the confounder loss), and an adversarial model without the confounder loss to be able to gauge the effects of it. We train 4 random seeds for each of these 3 conditions. Additionally, to test the performance of the discriminator, we used the alignment technique from Section 4 to align the ConceptNet test set of [52] to the ROCStories corpus. We then passed the story, target sentence and the test assertion into the discriminator to determine whether it was true or not. We used a threshold of 0.5 to determine whether an assertion was marked as true (1) or false (0).

5.7. *Effects of adversarial training*

Model	ROUGE1	ROUGE2	ROUGEL	ROUGELSUM	BLEU	METEOR	Accuracy of Discriminator
+ADVERSARIAL+CONFOUNDER	43.656	**10.544**	40.380	40.379	31.335	61.683	0.690
+ADVERSARIAL-CONFOUNDER	**43.747**	10.559	**40.530**	**40.531**	31.279	61.623	0.481
-ADVERSARIAL+CONFOUNDER	43.715	**10.680**	40.292	40.292	**31.470**	**61.776**	**0.794**

Table 8

We present the results of the adversarial test with ablations. We can see that the Adversarial models tend to have improved recall (ROUGE) scores, but lower precision (BLEU/METEOR) scores and lower accuracy on classifying a ConceptNet test set of assertions. The adversarial model with everything (+ADVERSARIAL+CONFOUNDER) strikes a balance of the benefits of precision and recall that the non-adversarial, non-confounder loss model give respectively.

After running automated tests, we see some mixed results between the three conditions (Adversarial+Confounder, -Adversarial+Confounder, Adversarial-Confounder). These results are in Table 8. We can see that the Adversarial models tend to have improved recall (ROUGE) scores, but lower precision (BLEU/METEOR) scores and lower accuracy on classifying a ConceptNet test set of assertions. The adversarial model with everything (+ADVERSARIAL+CONFOUNDER) strikes a balance of the benefits of precision and accuracy, and recall that the non-adversarial, non-confounder loss model give respectively. Some possible causes for these mixed results may be that our approach may be too naive, and possibly an improved GAN formulation such as the Wasserstein GAN [41] used in [40] may help our results, our approximation to connect the generator and discriminator may be too naive and may need a more complex approach such as utilizing a recurrent neural network during the generation steps to encode them then decode them into the discriminator.

6. Contributions & Takeaways

In this work we have presented three things: a method for controlling contextual commonsense inference called hinting, a method for combining multiple knowledge graphs for joint contextual commonsense inference, and an adversarial and non-adversarial model trained with these techniques. Taken altogether, we can obtain one model that is capable of inferring on a topic given by a hint, the inference can be performed on any subject, object, or a relation, and specificity from source knowledge bases, and the model's discriminator is capable of scoring assertions. These contributions serve as a baseline to explore the area of contextual commonsense inference, and leave much room to explore avenues on hinting, joint inference, and adversarial training of transformer-based language models.

References

[1] S. Gabriel, C. Bhagavatula, V. Shwartz, R. Le Bras, M. Forbes and Y. Choi, Paragraph-level Commonsense Transformers with Recurrent Memory, *Proceedings of the AAAI Conference on Artificial Intelligence* **35**(14) (2021), 12857–12865. https://ojs.aaai.org/index.php/AAAI/article/view/17521.

[2] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi and G. Neubig, Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, *arXiv preprint arXiv:2107.13586* (2021).

[3] R. Speer, J. Chin and C. Havasi, Conceptnet 5.5: An open multilingual graph of general knowledge, in: *Thirty-first AAAI conference on artificial intelligence*, 2017.

[4] J.D. Hwang, C. Bhagavatula, R.L. Bras, J. Da, K. Sakaguchi, A. Bosselut and Y. Choi, (Comet-) Atomic 2020: On Symbolic and Neural Commonsense Knowledge Graphs., in: *AAAI*, 2020, pp. 6384–6392.

[5] N. Mostafazadeh, A. Kalyanpur, L. Moon, D. Buchanan, L. Berkowitz, O. Biran and J. Chu-Carroll, GLUCOSE: GeneraLized and COntextualized Story Explanations, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Online, 2020, pp. 4569–4586. doi:10.18653/v1/2020.emnlp-main.370. https://aclanthology.org/2020.emnlp-main.370.

[6] B.M. Williams, H. Lieberman and P. Winston, A commonsense approach to story understanding, in: *Thirteenth International Symposium on Commonsense Reasoning (Commonsense-2017)*, 2017. http://www.media.mit.edu/~lieber/Publications/Understanding-Stories-Commonsense.pdf.

[7] N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli and J. Allen, 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*, 2016, pp. 839–849.

[8] Z. Zhang, X. Geng, T. Qin, Y. Wu and D. Jiang, Knowledge-aware procedural text understanding with multi-stage training, in: *Proceedings of the Web Conference 2021*, 2021, pp. 3512–3523.

[9] H. Liu and P. Singh, ConceptNet—a practical commonsense reasoning tool-kit, *BT technology journal* **22**(4) (2004), 211–226.

[10] J. Da and J. Kasai, 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*, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 1–12. doi:10.18653/v1/D19-6001. https://aclanthology.org/D19-6001.

[11] J. Davison, J. Feldman and A.M. Rush, 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)*, 2019, pp. 1173–1178.

[12] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Çelikyilmaz and Y. Choi, COMET: Commonsense Transformers for Automatic Knowledge Graph Construction, in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[13] M. Sap, R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N.A. Smith and Y. Choi, Atomic: An atlas of machine commonsense for if-then reasoning, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 3027–3035.

[14] N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli and J. Allen, 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*, Association for Computational Linguistics, San Diego, California, 2016, pp. 839–849. doi:10.18653/v1/N16-1098. https://aclanthology.org/N16-1098.

[15] C.-Y. Lin, ROUGE: A Package for Automatic Evaluation of Summaries, in: *Text Summarization Branches Out*, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 74–81. https://www.aclweb.org/anthology/W04-1013.

[16] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P.J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, *Journal of Machine Learning Research* **21**(140) (2020), 1–67. http://jmlr.org/papers/v21/20-074.html.

[17] N. Reimers and I. Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, in: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2019. https://arxiv.org/abs/1908.10084.

[18] T. Shin, Y. Razeghi, R.L. Logan IV, E. Wallace and S. Singh, Autoprompt: Eliciting knowledge from language models with automatically generated prompts, *arXiv preprint arXiv:2010.15980* (2020).

[19] X.L. Li and P. Liang, Prefix-tuning: Optimizing continuous prompts for generation, *arXiv preprint arXiv:2101.00190* (2021).

[20] B. Lester, R. Al-Rfou and N. Constant, The power of scale for parameter-efficient prompt tuning, *arXiv preprint arXiv:2104.08691* (2021).

[21] X. Chen, X. Xie, N. Zhang, J. Yan, S. Deng, C. Tan, F. Huang, L. Si and H. Chen, AdaPrompt: Adaptive Prompt-based Finetuning for Relation Extraction, *CoRR* **abs/2104.07650** (2021). https://arxiv.org/abs/2104.07650.

[22] N.S. Keskar, B. McCann, L.R. Varshney, C. Xiong and R. Socher, Ctrl: A conditional transformer language model for controllable generation, *arXiv preprint arXiv:1909.05858* (2019).

[23] Z.-Y. Dou, P. Liu, H. Hayashi, Z. Jiang and G. Neubig, 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*, Association for Computational Linguistics, Online, 2021, pp. 4830–4842. doi:10.18653/v1/2021.naacl-main.384. https://aclanthology.org/2021.naacl-main.384.

[24] N. Peng, M. Ghazvininejad, J. May and K. Knight, Towards controllable story generation, in: *Proceedings of the First Workshop on Storytelling*, 2018, pp. 43–49.

[25] F. Brahman, A. Petrusca and S. Chaturvedi, 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*, Association for Computational Linguistics, Suzhou, China, 2020, pp. 588–597. https://aclanthology.org/2020.aacl-main.59.

[26] A. See, S. Roller, D. Kiela and J. Weston, 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)*, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 1702–1723. doi:10.18653/v1/N19-1170. https://aclanthology.org/N19-1170.

[27] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi and Y. Choi, HellaSwag: Can a Machine Really Finish Your Sentence?, in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[28] Y.W. Lisa Bauer* and M. Bansal, Commonsense for Generative Multi-Hop Question Answering Tasks, in: *Proceedings of the Empirical Methods in Natural Language Processing*, 2018.

[29] C. Havasi, R. Speer, J. Pustejovsky and H. Lieberman, Digital Intuition: Applying Common Sense Using Dimensionality Reduction, *IEEE Intelligent Systems* **24**(4) (2009), 24–35. doi:10.1109/MIS.2009.72.

[30] D.B. Lenat, R.V. Guha, K. Pittman, D. Pratt and M. Shepherd, Cyc: Toward Programs with Common Sense, *Commun. ACM* **33**(8) (1990), 30–49–. doi:10.1145/79173.79176.

[31] Y. Chalier, S. Razniewski and G. Weikum, Joint reasoning for multi-faceted commonsense knowledge, *arXiv preprint arXiv:2001.04170* (2020).

[32] T.-P. Nguyen, S. Razniewski and G. Weikum, Advanced Semantics for Commonsense Knowledge Extraction, in: *Proceedings of the Web Conference 2021*, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 2636–2647–. ISBN 9781450383127. doi:10.1145/3442381.3449827.

[33] F. Ilievski, A. Oltramari, K. Ma, B. Zhang, D.L. McGuinness and P. Szekely, Dimensions of commonsense knowledge, *Knowledge-Based Systems* **229** (2021), 107347. doi:https://doi.org/10.1016/j.knosys.2021.107347. https://www.sciencedirect.com/science/article/pii/S0950705121006092.

[34] B.D. Mishra, N. Tandon and P. Clark, Domain-targeted, high precision knowledge extraction, *Transactions of the Association for Computational Linguistics* **5** (2017), 233–246.

[35] J. Romero, S. Razniewski, K. Pal, J. Z. Pan, A. Sakhadeo and G. Weikum, Commonsense Properties from Query Logs and Question Answering Forums, in: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1411–1420–. ISBN 9781450369763. doi:10.1145/3357384.3357955.

[36] T.-P. Nguyen, S. Razniewski and G. Weikum, Advanced semantics for commonsense knowledge extraction, in: *Proceedings of the Web Conference 2021*, 2021, pp. 2636–2647.

[37] S. Bhakthavatsalam, C. Anastasiades and P. Clark, Genericskb: A knowledge base of generic statements, *arXiv preprint arXiv:2005.00660* (2020).

[38] P. West, C. Bhagavatula, J. Hessel, J.D. Hwang, L. Jiang, R.L. Bras, X. Lu, S. Welleck and Y. Choi, Symbolic Knowledge Distillation: from General Language Models to Commonsense Models, *arXiv preprint arXiv:2110.07178* (2021).

[39] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, Language Models are Few-Shot Learners, in: *Advances in Neural Information Processing Systems*, Vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan and H. Lin, eds, Curran Associates, Inc., 2020, pp. 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[40] O. Press, A. Bar, B. Bogin, J. Berant and L. Wolf, Language Generation with Recurrent Generative Adversarial Networks without Pre-training, *arXiv preprint arXiv:1706.01399* (2017).

[41] M. Arjovsky, S. Chintala and L. Bottou, Wasserstein Generative Adversarial Networks, in: *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y.W. Teh, eds, Proceedings of Machine Learning Research, Vol. 70, PMLR, 2017, pp. 214–223. https://proceedings.mlr.press/v70/arjovsky17a.html.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser and I. Polosukhin, Attention is All You Need, in: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 6000–6010–. ISBN 9781510860964.

[43] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, 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)*, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423. https://aclanthology.org/N19-1423.

[44] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu and A. Miller, 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)*, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 2463–2473. doi:10.18653/v1/D19-1250. https://aclanthology.org/D19-1250.

[45] I. Sutskever, O. Vinyals and Q.V. Le, Sequence to sequence learning with neural networks, in: *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[46] D.P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, *CoRR* **abs/1412.6980** (2015).

[47] M. Post, A Call for Clarity in Reporting BLEU Scores, in: *Proceedings of the Third Conference on Machine Translation: Research Papers*, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 186–191. doi:10.18653/v1/W18-6319. https://aclanthology.org/W18-6319.

[48] S. Banerjee and A. Lavie, 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*, Association for Computational Linguistics, Ann Arbor, Michigan, 2005, pp. 65–72. https://www.aclweb.org/anthology/W05-0909.

[49] Q. Lhoest, A.V. del Moral, P. von Platen, T. Wolf, Y. Jernite, A. Thakur, L. Tunstall, S. Patil, M. Drame, J. Chaumond, J. Plu, J. Davison, S. Brandeis, V. Sanh, T.L. Scao, K.C. Xu, N. Patry, S. Liu, A. McMillan-Major, P. Schmid, S. Gugger, N. Raw, S. Lesage, A. Lozhkov, M. Carrigan, T. Matussière, L. von Werra, L. Debut, S. Bekman and C. Delangue, huggingface/datasets: 1.13.2, Zenodo, 2021. doi:10.5281/zenodo.5570305.

[50] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, *arXiv preprint arXiv:1907.11692* (2019).

[51] J. Johnson, M. Douze and H. Jégou, Billion-scale similarity search with GPUs, *arXiv preprint arXiv:1702.08734* (2017).

[52] X. Li, A. Taheri, L. Tu and K. Gimpel, Commonsense knowledge base completion, in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1445–1455.

[53] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, Generative Adversarial Nets, in: *Advances in Neural Information Processing Systems*, Vol. 27, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence and K.Q. Weinberger, eds, Curran Associates, Inc., 2014. https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[54] M. Mirza and S. Osindero, Conditional generative adversarial nets, *arXiv preprint arXiv:1411.1784* (2014).

[55] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer, BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, 2020, pp. 7871–7880. doi:10.18653/v1/2020.acl-main.703. https://aclanthology.org/2020.acl-main.703.

[56] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz et al., Huggingface's transformers: State-of-the-art natural language processing, *arXiv preprint arXiv:1910.03771* (2019).

[57] E.M. Ponti, I. Vulić, G. Glavaš, N. Mrkšić and A. Korhonen, Adversarial Propagation and Zero-Shot Cross-Lingual Transfer of Word Vector Specialization, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 282–293. doi:10.18653/v1/D18-1026. https://aclanthology.org/D18-1026.

[58] P. Colon-Hernandez, Y. Xin, H. Lieberman, C. Havasi, C. Breazeal and P. Chin, RetroGAN: A Cyclic Post-Specialization System for Improving Out-of-Knowledge and Rare Word Representations, in: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Association for Computational Linguistics, Online, 2021, pp. 2086–2095. doi:10.18653/v1/2021.findings-acl.183. https://aclanthology.org/2021.findings-acl.183.

[59] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J.E.L. Gayo, S. Kirrane, S. Neumaier, A. Polleres et al., Knowledge graphs, *arXiv preprint arXiv:2003.02320* (2020).

[60] R. Speer and C. Havasi, Representing General Relational Knowledge in ConceptNet 5., in: *LREC*, 2012, pp. 3679–3686.

Appendix A. Definitions

Contextual/Discourse-Aware Commonsense Inference Task in which, given a textual context (e.g., story) and a selected sentence from that context, a model must infer a coherent and contextual commonsense assertion.

Assertion Used interchangeably with a fact, it is a tuple that consists of a subject, relation, object, and generality and represents a fact.

Sentence-Level Commonsense Inference Generation of a commonsense assertion utilizing, at most, a sentence as context.

Story specific commonsense assertion inference Commonsense assertion templates that are instanced by elements from a story.

General commonsense assertion inference Commonsense assertion templates that are not instanced, but are derived from a story.

Specificity Whether an assertion's content is about entities in the aligned story (i.e., given context), or if it is a generalized version of an assertion.

Joint Commonsense Inference To infer commonsense knowledge assertions by leveraging knowledge from multiple knowledge bases.

Hint The part(s) of an assertion that a model has to predict, along with special identifiers for these parts, wrapped within parenthesis characters, that are passed to a model to infer a commonsense assertion.

Hinting Proposed technique that trains a contextual commonsense inference model that can be guided on its inference using hints.

Knowledge Graph A knowledge graph (used somewhat interchangeably with knowledge base, although they are different concepts) is defined as "a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities" [59]. Formally, a knowledge graph is a set of tuples that represents nodes and edges between these nodes. Let us define a set of vertices (which we will refer to as concepts) as V, a set of edges as E (which we will refer to as assertions as per Speer and Havasi [60]), and a set of labels L (which we will refer to as relations). A knowledge graph is a tuple $G := (V, E, L)$. We use the formal definitions found in Appendix B of [59]. The set of edges (E) or assertions is composed of tuples $E \subseteq V \times L \times V$ which are seen as a subject (a concept), a relation (a label), and object (another concept) respectively (e.g., $(subject, relation, object)$). These edges in some cases can have weights to represent the strength of the assertion, and in this work they additionally have a parameter of *specificity* (whether the assertion's content is about entities in the aligned story, or if it is a generalized version of an assertion). Broadly speaking, knowledge graphs (KGs) are a collection of tuples that represent things that should be true within the knowledge of the world that we are representing.

Generative Adversarial Network (GAN) Adversarial system in which a Generator model produces inferences that are scored by a Discriminator system. The Discriminator provides feedback to the Generator on how to improve at the same time that it tries to improve itself on Discriminating Generated and Real data. A more detailed explanation can be found in [53].