

Conversational Multi-Hop Reasoning with Neural Commonsense Knowledge and Symbolic Logic Rules

Forough Arabshahi*

Facebook
forough@fb.com

Jennifer Lee*

Facebook
jenniferlee98@fb.com

Antoine Bosselut

EPFL
antoine.bosselut@epfl.ch

Yejin Choi

University of Washington
yejin@cs.washington.edu

Tom Mitchell

Carnegie Mellon University
tom.mitchell@cmu.edu

Abstract

One of the challenges faced by conversational agents is their inability to identify unstated *presumptions* of their users’ commands, a task trivial for humans due to their common sense. In this paper, we propose a zero-shot commonsense reasoning system for conversational agents in an attempt to achieve this. Our reasoner uncovers unstated presumptions from user commands satisfying a general template of **if-(state), then-(action), because-(goal)**. Our reasoner uses a state-of-the-art transformer-based generative commonsense knowledge base (KB) as its source of background knowledge for reasoning. We propose a novel and iterative knowledge query mechanism to extract multi-hop reasoning chains from the neural KB which uses symbolic logic rules to significantly reduce the search space. Similar to any KBs gathered to date, our commonsense KB is prone to missing knowledge. Therefore, we propose to conversationally elicit the missing knowledge from human users with our novel dynamic question generation strategy, which generates and presents contextualized queries to human users. We evaluate the model with a user study with human users that achieves a 35% higher success rate compared to SOTA.

1 Introduction

Conversational agents are becoming prominent in our daily lives thanks to advances in speech recognition, natural language processing and machine learning. However, most conversational agents still lack commonsense reasoning, preventing them from engaging in rich conversations with humans.

Recently, Arabshahi et al. (2021) proposed a commonsense reasoning benchmark task for conversational agents that contains natural language commands given to an agent by humans. These commands follow a general template of:

* Equal contribution

Command: “If the weather is bad tomorrow then remind me to bring an umbrella because I want to stay warm.”

CLUE : Why does “bringing an umbrella” help you “stay warm”?

1. Because I need an umbrella
- 2. Because I stay dry**
3. Because it rains
4. Because the umbrella gets wet
5. Because rain gets on me
6. None of the above

Please choose number and provide an explanation

User: 2. if I bring an umbrella and use it then I will stay dry. If it stay dry then I will stay warm.

Figure 1: CLUE’s conversation with a human. The multiple choices **1** through **5** are commonsense knowledge obtained from COMET using our multi-hop reasoner, ranked by the reasoner’s confidence. Since the user chooses **Option 2** (Because I stay dry), CLUE selects this reasoning path as the final reasoning chain.

“**If (state holds), Then (perform action), Because (I want to achieve goal)**”. We refer to commands satisfying this template as if-then-because commands. As stated in Arabshahi et al. (2021), humans often under-specify conditions on the if-portion (*state*) and/or then-portion (*action*) of their commands. These under-specified conditions are referred to as *commonsense presumptions*. For example, consider the command, “If it’s going to rain in the afternoon (·) Then remind me to bring an umbrella (·) Because I want to remain dry”, where (·) indicates the position of the unstated commonsense presumptions. The presumptions for this command are (and I am outside) and (before I leave the house), respectively. The goal in this task is to infer such commonsense presumptions given if-then-because commands.

In this paper, we propose the Conversational Multi-hop Reasoner (CLUE) for this task which performs zero-shot reasoning. CLUE extracts a multi-hop reasoning chain that indicates how the

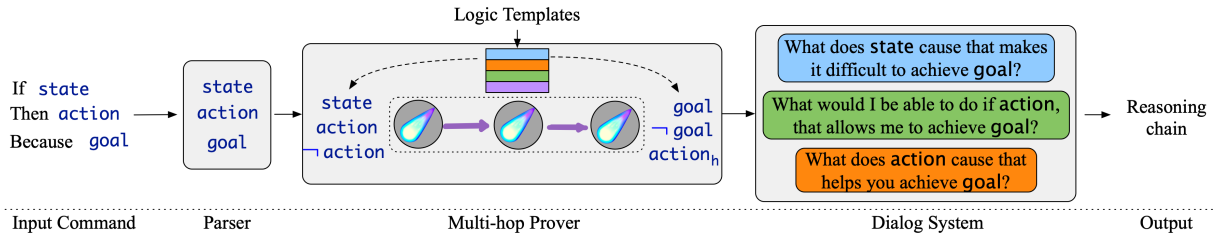


Figure 2: CLUE diagram. CLUE has three main components: The parser, the multi-hop reasoner (prover) and the dialog system. Given an if-then-because command, the parser extracts independent natural language clauses for the *state*, *action* and *goal*. The prover extracts multi-hop reasoning chains given the logic templates using our neural commonsense KB, COMET that indicates how the *action* achieves the *goal* when the *state* holds. The extracted reasoning chains go through the dialog system that generates template-dependent questions and converses with a human who either validates the returned proofs or contributes novel commonsense knowledge if the proofs are incorrect.

action leads to the *goal* when the *state* holds. For example, the simplified reasoning chain for the previous example is, “if I am reminded to bring an umbrella before I leave the house, then I have an umbrella”, “if I have an umbrella and it rains when I am outside, then I can use the umbrella to block the rain”, “if I block the rain, then I remain dry”. Additional commonsense knowledge provided by the reasoning chain is considered a commonsense presumption of the input command. In order to construct the multi-hop reasoning chain, we develop a novel reasoning system that uses a few symbolic logic templates to prune the exponential reasoning search space, resulting in significantly improved generated commonsense knowledge.

Our multi-hop reasoner uses a state-of-the-art (SOTA) transformer-based commonsense knowledge model called COMMONSENSE TRANSFORMERS (COMET) (Bosselut et al., 2019) as its source of background knowledge and is the first time reasoning task is tackled using a large scale KB. Knowledge models can be used in place of KBs, but they are more flexible in terms of information access, so we use this term interchangeably with KB.

Despite being a SOTA KB, COMET still misses requisite knowledge for reasoning about if-then-because commands. In fact, many if-then-because commands often fall under the long tail of tasks for which there is too little knowledge available in any commonsense KBs. For example, one of the commands in the dataset is: “If I get an email with subject the gas kiln fired, then send me a list of all the pots I put on the glaze shelf between the last firing and now, because I want to pick up the pots from the studio.” It is unlikely for any knowledge source to contain a fact that is contextually relevant

to this command. More importantly, it is also unlikely that a command requiring the same type of reasoning will occur again in the future, making it cost-ineffective to manually annotate.

To overcome this, we propose conversationally eliciting missing knowledge from human users. Conversation with users is readily available since CLUE is developed for conversational agents. We develop a novel question generation strategy that uses COMET to generate questions in the context of the input if-then-because commands, allowing CLUE to acquire contextual feedback from humans. We evaluate our reasoner by conducting a user study with humans.

Summary of Results and Contributions: The contributions of this paper are two-fold. First, we propose a novel conversational commonsense reasoning approach called Conversational mUlti-hop rEasoner (CLUE) that incorporates Commonsense Transformers (COMET), a large scale neural commonsense KB, as its main source of background knowledge. We propose a multi-hop knowledge query mechanism that extracts reasoning chains by iteratively querying COMET. This mechanism uses, for the first time, a few symbolic logic templates to prune the reasoning search space, significantly improving the quality of the generated commonsense knowledge. Second, we propose a conversational knowledge acquisition strategy that uses the knowledge extracted from COMET to dynamically ask contextual questions to humans whenever there is missing knowledge. We ran a user study and evaluated our proposed approach using real human users. Our results show that CLUE achieves a 35% higher success rate compared to a baseline, which uses a less sophisticated conversational inter-

face and a smaller background knowledge on the benchmark dataset. We also extensively evaluate the performance of the reasoner in an isolated non-conversational setting and empirically re-iterate the need for conversational interactions. CLUE’s components were not trained on the benchmark dataset. Therefore, our results assess the performance of CLUE’s zero-shot reasoning.

2 Background and Notation

In this section, we briefly re-introduce the benchmark task, the logic templates, and Arabshahi et al.’s reasoning engine CORGI (COMmonsense ReasoninG By Instruction), along with an overview of COMET (Bosselut et al., 2019).

2.1 If-Then-Because Commands

The benchmark dataset (Arabshahi et al., 2021) contains natural language commands given to conversational agents. These commands follow the template “**If-(state), Then-(action), Because-(goal)**”. The if-clause is referred to as the *state*, the then-clause as the *action* and the because-clause as the *goal*.

Logic Templates: The data is partitioned into 4 color-coded reasoning logic templates that indicate how the commanded *action* leads to the *goal* when the *state* holds. To be self-contained, we have included the table of logic templates from Arabshahi et al. in Figure 3. In the interest of space, we give one example of the blue logic template, $(\neg(\text{goal}) \text{ :- } \text{state}) \wedge (\text{goal} \text{ :- } \text{action}(\text{state}))$, where \neg indicates negation, and \wedge indicates logical AND. Under this template, the *state* implies the negation of the *goal* AND the *action* implies the *goal* when the *state* holds. An example if-then-because command that satisfies this template is, “If it snows tonight then wake me up early because I want to get to work on time”. Here, the *state* of snowing (a lot) at night results in the negation of the *goal* and the user will not be able to get to work on time. AND if the *action* of waking the user up earlier is performed when the *state* of snowing (a lot) at night holds, the user will get to work on time. The other three follow different logic templates, but are of the same nature. We refer the reader to Arabshahi et al. for more details.

$(\neg(\text{goal}) \text{ :- } \text{state}) \wedge (\text{goal} \text{ :- } \text{action}(\text{state}))$	If it snows tonight then wake me up early because I want to arrive to work on time
$((\text{goal}) \text{ :- } \text{action}(\text{state})) \wedge (\neg(\text{goal}) \text{ :- } \neg(\text{action}(\text{state})))$	If I am walking to a meeting then remind me who else is there because I want to be prepared for the meeting
$((\text{goal}) \text{ :- } \text{action}_h) \wedge (\text{action}_h \text{ :- } \text{action}(\text{state}))$	If we are approaching Fall then remind me to buy flower bulbs because I want to make sure I have a pretty Spring garden.
$(\text{goal} \text{ :- } \text{state}) \wedge (\neg(\text{goal}) \text{ :- } \text{action}(\text{state}))$	If I am at the grocery store but I have a trip coming up in the next week then remind me not to buy perishables because they will go bad while I am away

Figure 3: Logic templates and an example if-then-because command for each one. \wedge , \neg , and :- denote logical AND, negation, and implication, respectively. action_h indicates a hidden action. For example, the hidden action for the third command (green template) is “planting flower bulbs”.

2.2 CORGI

COMmonsense ReasoninG By Instruction (CORGI; Arabshahi et al. (2021)) is the SOTA commonsense reasoning engine designed for inferring commonsense presumptions of if-then-because commands. CORGI has a neuro-symbolic reasoning module (theorem prover) that generates multi-hop reasoning trees given if-then-because commands. The neuro-symbolic reasoning module is a soft logic programming system that reasons using backward chaining (a backtracking algorithm). CORGI does not use the logic templates to do reasoning. One of the main limitations of CORGI is that it uses a small hand-crafted KB that is not diverse enough to reason about all the if-then-because commands in the benchmark dataset. CORGI was tested only on 10 commands in the released benchmark (Arabshahi et al., 2021). Moreover, CORGI’s KB is programmed in a syntax similar to Prolog (Colmerauer, 1990), but large-scale commonsense Prolog-like knowledge bases are not readily available. Therefore, in this paper we propose to use a large-scale SOTA neural commonsense KB which both extends that limited source of background knowledge and also enables the use of a new *type* of knowledge source.

CORGI is equipped with a conversational interaction strategy to acquire knowledge from humans when that knowledge is missing from the KB. However, CORGI uses a static question generation strategy that often confuses the end-users. First, the parser often omits verb conjugation or subjects, resulting in grammatically incorrect questions. Second, the static question generation strategy does not always generate contextual queries. Therefore, in this paper we propose a dynamic question genera-

tion strategy that generates more relevant questions resulting in a higher quality knowledge extraction from humans. We also include a more robust parser to ensure the questions are grammatically correct.

2.3 COMET

COMMONSENSE Transformer (COMET; Bosselut et al.) is a generative transformer-based commonsense KB that learns to generate rich and diverse commonsense descriptions. COMET constructs commonsense KBs by using existing tuples as a seed set of knowledge on which to train. In essence, a pre-trained language model learns to adapt its learned representations to knowledge generation, producing novel high-quality tuples.

Unlike other KBs, COMET represents knowledge implicitly in its neural network parameters and expresses knowledge through generated free-form open-text descriptions. This makes it well-suited for the studied reasoning task, as the if-then-because commands are also expressed in free-form text descriptions. Therefore, we can directly query COMET for commonsense pre-conditions and post-effects of the `state`, `action` and `goal`. For example, “pouring coffee” is a commonsense pre-condition for “drinking coffee” (Fig 4a).

Our system uses a COMET model trained on two knowledge graphs, ATOMIC (Sap et al., 2019) and ConceptNet (Speer et al., 2016). Each of these knowledge graphs consists of a collection of tuples, $\{s, r, o\}$, where s and o are the *subject* and *object* phrase of the tuple, respectively and $r \in [r^0, r^1, \dots, r^{\ell-1}]$ is the *relation* between s and o , and ℓ is the number of relations in the KB. COMET is trained to generate o given s and r . E.g., Figure 4 shows examples of the generated objects given an input subject, “I drink coffee,” and several relations from the ATOMIC-trained (Fig. 4a) and ConceptNet-trained (Fig. 4b) COMET models.

3 CLUE: Conversational Multi-Hop Reasoner

CLUE (Conversational Multi-hop Reasoner) is a commonsense reasoning engine that inputs if-then-because commands and outputs a reasoning chain (proof) containing if-then logical statements that indicates how the `action` achieves the user’s `goal` when the `state` holds. CLUE is built on top of CORGI and is triggered when CORGI fails. It consists of three components: (1) the parser, (2) the multi-hop prover (reasoner), and (3) the dialog sys-

tem (Figure 2). The Parser takes in the command and extracts the `state`, `action` and `goal` from it. At the second step, the prover attempts to find reasoning chains that connect the `action` to the `goal` when the `state` holds. The extracted reasoning chains go through the dialog system that generates and presents contextualized questions to a human user who either validates a returned reasoning chain or contributes novel knowledge to CLUE if none of the reasoning chains are correct.

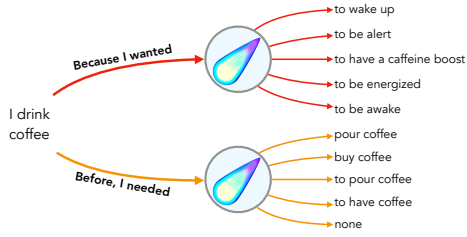
All three components interact with CLUE’s two sources of background knowledge: (I) A small handcrafted knowledge base containing logic facts and rules, \mathcal{K} , programmed in a logic programming language and (II) A large scale neural knowledge base, COMET, containing free-form text. During the user-interaction sessions, the system’s background knowledge base \mathcal{K} grows in size as new knowledge is added to it. This new knowledge is either novel information added by the user during conversational interactions, or knowledge queried from COMET and confirmed by the user. This new knowledge will be added to \mathcal{K} for future similar reasoning tasks. In what follows, we explain the three components of CLUE in detail.

3.1 Parser

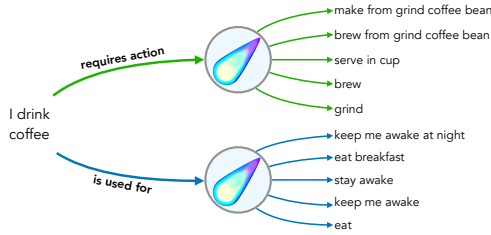
Our parser extracts the `state`, `action`, and `goal` as independent clauses from the input if-then-because command. We use Spacy’s (Honnibal and Montani, 2017) NLP tools such as POS tagging and coreference resolution to make the clauses self-contained and contextual.

3.2 Multi-Hop Prover

Inspired by how Prolog generates proofs containing chains of logical rules and facts using a KB, our prover (reasoner) generates chains of natural language commonsense facts and rules to reason about an input command by iteratively querying COMET (details on the analogue between Prolog and our prover are in the appendix). We denote a COMET query by $\text{COMET}(r, s)$ where r is a relation and s is a subject or an input natural language clause. COMET uses beam search at decoding time and $\text{COMET}(r, s)$ outputs a list of candidate objects $O = [o^0, o^1, \dots, o^{b-1}]$ ordered by confidence, where b indicates COMET’s beam size. We categorize COMET *relations* into two classes, namely pre-conditions and post-effects. For example, the relations “Because I wanted” and “is used for” in Fig 4 are post-effects, whereas the relations “Before



(a) ATOMIC beam results for two relations *Because I wanted* and *Before, I needed*



(b) ConceptNet beam results for two relations *requires action* and *is used for*

Figure 4: $\text{COMET}_r(s)$ generations for $s = I \text{ drink coffee}$ and two different relations r . Our neural KB was trained on ATOMIC and ConceptNet knowledge graphs. The generations are listed on the right of the block.

I needed” and “requires action” are pre-conditions.

Let us now formally define a reasoning chain or proof extracted from our neural knowledge base. A proof consists of a chain of knowledge tuples $\{s_i, r_i, o_i\}$, where $i \in [1, N]$ and N indicates the number of hops in the proof chain such that o_{i-1} is *semantically close* to s_i . (We discuss the notion of semantic closeness in the next subsection.) The search space for finding a proof chain from COMET grows exponentially with N if implemented naively. In the next subsections we explain how we prune the search space using the logic templates released with the benchmark dataset.

The goal of our prover is to scale up CORGI’s small hand-crafted knowledge base, which is programmed in a Prolog-like language. Since a large-scale commonsense KB in Prolog is not readily available, CLUE proposes an alternative prover that enables using SOTA large-scale commonsense KBs. Moreover, CLUE’s prover is consistent with CORGI’s neuro-symbolic logic theorem prover (refer to the appendix for details), allowing us to seamlessly extend CORGI’s background knowledge without requiring a large scale commonsense KB programmed in Prolog. Lastly, CLUE performs reasoning in a zero-shot manner since the pre-trained COMET is not trained on the bench-

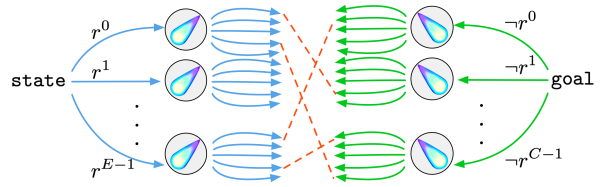


Figure 5: Bidirectional two-hop proof for the blue template $(\neg(\text{goal}) :- \text{state})$. The dotted orange lines indicate semantic closeness. r^i ’s are KB relations, C is the number of pre-condition relations and E is the number of post-effect relations in the KB. The \neg symbol refers to relations that represent negation; for example, *NotCapableOf* or *NotIsA*.

mark dataset used for evaluation in this paper.

Semantic Closeness: In order to measure semantic closeness of the *object* and *subject* phrases, we embed the sequence of tokens that make up the object of the previous tuple (o_{i-1}) and the sequence of tokens that make up the subject of the next tuple (s_i) in the proof trace. Semantic closeness, used for ranking the returned proofs, is defined as a vector cosine similarity of larger than a threshold, τ .

We investigate several embedding methods to find one that best suites our multi-hop prover. We used GloVe embeddings (Pennington et al., 2014), BERT pre-trained embeddings (Devlin et al., 2019) and fine-tuned embeddings in COMET, which we call *commonsense embeddings*. For GloVe and BERT embeddings, we compute the phrase embedding by averaging the embeddings of the tokens. For commonsense embeddings, we use the phrase embeddings returned by COMET. In the results section, we compare the outcome of these choices.

Pruning the Proof Search Space In order to form a reasoning chain (proof) for an if-then-because command using COMET, we leverage the logic templates discussed in Sec. 2. All the templates consist of a conjunction of two logical implications. For each implication (Head :- Body), the Head and Body are given in the if-then-because command. For example, the goal and state in the first implication of the purple template (goal :- state) are extracted from the input command. Therefore, in order to prove the first implication, we need to find a chain of reasoning that leads from the state to the goal in a series of N hops. In order to do that, we either perform unidirectional or bidirectional beam search as explained below.

Unidirectional Beam Search: Here, for a given number of hops N , we first construct all beam results $O_1 = \text{COMET}(r_e, s_1)$ where s_1 is the natural language clause that corresponds to the Body of the implication in the logic template, and for all r_e that are post-effect COMET relations. We continue to query $\text{COMET}(r_e, o_n^j)$ recursively in a breadth-first manner for $\forall o_n^j \in O_n$ where $n \in [1, \dots, N]$ is the hop index. In each hop, we only continue the query for the top K results ranked by the semantic closeness of the returned beam result o_n^j and the implication’s Head, where K is the search’s beam size. At the N^{th} hop, the prover returns proof chains for which o_N is semantically close to the implication’s Head, which corresponds to the goal.

Bidirectional Beam Search: Here, we construct all possible beam results of $\text{COMET}(r_e, s_1)$, where s_1 is the natural language clause that corresponds to the implication’s Body, for all r_e that are post-effect relations as well as $\text{COMET}(r_c, o_N)$, where o_N is the natural language clause corresponding to the Head of the implication in the logic template, for all r_c that are pre-condition relations. The proof succeeds when two intermediate beam results are semantically close in the beam search path from either direction (Fig 5).

3.3 Dialog System

The reasoning chains obtained by the prover are passed to the Dialog System, which has two main goals. The first is to confirm the prover obtained valid reasoning chains by asking the user if they think the automatically recovered proofs are “correct” from a commonsense stand point (Fig 1). The necessity of this was also confirmed in [Bosselut et al. \(2019\)](#)’s human evaluation studies. The second is to overcome the problem of missing knowledge when the user rejects all the proofs returned by the prover. We introduce the dialog generators responsible for fulfilling each of the above goals in what follows.

Humans as Knowledge Evaluators The dialog generator confirms the returned COMET proofs with humans before adding it as background knowledge to \mathcal{K} . In order to do this, it chooses the top 5 proofs with the highest similarity scores and presents them as candidates to the human user to choose from. Our study shows that these multiple choices not only help confirm COMET results but they also provide guidance to users as to what information the system is looking for.

As shown in Figure 1, five explanations for the question are returned by CLUE. The user chooses one and provides an explanation as to why he/she chose that option. The explanation is in open-domain text, formatted as a series of if-then statements. This is because if-then formatted explanations can be easily parsed using our parser. In this step, the question asked from users is contingent on which logic template the if-then-because command follows. The dialog system’s flowchart as well as an example dialog are in the appendix.

Humans as Novel Knowledge Contributors

We also use human interaction to acquire novel knowledge that does not exist in the background knowledge bases, \mathcal{K} and COMET.

When faced with missing knowledge in \mathcal{K} , CLUE uses the same technique from [Arabshahi et al. \(2021\)](#) with a rephrased, more comprehensive question, “what ensures $\langle \text{goal} \rangle$?” resulting in higher quality feedback. If the user’s response to the multiple-choice question is “None of the above”, it indicates that COMET has missing knowledge. CLUE then asks the user for an explanation and adds the new knowledge to \mathcal{K} and runs CORGI’s reasoning module to construct a proof. It is worth noting that since \mathcal{K} is orders of magnitudes smaller than SOTA knowledge bases, growing it with novel knowledge does not introduce any scalability issues.

4 Experiments and Results

Here we discuss our experimental setup, evaluation method, baselines and results. We work with 132 out of the 160 commands from the benchmark dataset (2021) that fall under three logic templates blue, orange and green (Figure 3). The red template contains commands for which there is no unifying logic template. Therefore, we cannot use it.

Evaluation: We do not use automated evaluation metrics and instead use human evaluations for two reasons. First, there are currently no metrics in the literature that assess whether the returned reasoning chains are “correct” from a commonsense perspective. Second, evaluating dialog systems is challenging. It is debated that metrics such as BLEU ([Papineni et al., 2002](#)) and perplexity often fail to measure true response quality ([Liu et al., 2016](#); [Li et al., 2016](#)).

Experiments: In the first experiment, we evaluate our multi-hop prover in isolation and without

conversational interactions. The human evaluators in this study are expert evaluators. In our second experiment, we test CLUE’s performance end-to-end with non-expert human users and investigate the efficacy of the conversational interactions. In order to be comparable with Arabshahi et al. (2021)’s study, we use the knowledge base of commonsense facts and rules, \mathcal{K} , released with the dataset. It contains 228 facts and rules. Our COMET model is pre-trained on knowledge graphs ConceptNet and ATOMIC. We used COMET’s open-source code¹ for training with the hyper-parameters reported there (Bosselut et al., 2019).

4.1 Multi-hop Prover Evaluation

As shown in Figure 3, each logic template consists of a conjunction of two logical implication statements. We use the terminology *end-to-end* proof to refer to proving both of the implications in the template and *half-proof* to refer to proving one.

Table 1 presents the number of proved logical implications `goal :- action`, with respect to the number of hops using unidirectional and bidirectional beam search. Expert-verified proofs refer to reasoning chains with similarity score of at least 0.8 that are validated by a human evaluator. The similarity threshold of 0.8 was tuned offline on a small subset of the benchmark dataset and picked from the following list [0.7, 0.8, 0.85, 0.9]. Automated proofs are the portion of the human evaluated proofs for which the highest scoring proof is the verified one. As shown, the number of automated proofs almost doubles when an expert human evaluator validates the proofs. This indicates that the model benefits from *human knowledge evaluators*. An instance of this scenario is shown in Figure 1 where the human user chooses the second ranking candidate as the correct proof. As expected, a portion of the commands cannot be proved using COMET alone even with human evaluators. This indicates that we are encountering missing knowledge. Therefore, there is need for *humans as novel knowledge contributors*. Moreover, Table 1 shows that bidirectional beam search is more successful than unidirectional search for lower hops greater than 1. This is because there is a higher chance of finding a good scoring match when the two directions meet due to an extended search space. For the same reason, the number of successful proofs drop when the number of hops

Table 1: Number of hops (N) required to obtain a half-proof for the implication `goal :- action` for two proof search strategies and 132 commands. Similarity score is computed using GloVe embeddings.

Pruning		N				
		1	2	3	4	5
Unidirectional beam search	# Expert-verified proofs	50	21	13	2	1
	# Automated proofs	26	10	6	1	1
Bidirectional beam search	# Expert-verified proofs	N/A	67	16	9	3
	# Automated proofs	N/A	42	6	0	0

Table 2: Number of successful unidirectional half-proofs for the implication `goal :- action` for a given number of hops (N) among the top k proof candidates for 132 commands. Similarity score is computed using GloVe embeddings.

k \ N		N				
		1	2	3	4	5
1		26	10	6	1	1
2		32	12	7	1	1
3		35	12	9	1	1
4		37	14	9	1	1
5		39	14	10	1	1

is increased beyond a certain point. Please note, in bidirectional beam search $N = 1$ is not applicable because the smallest number of hops extracted for bidirectional beam search is 2. Moreover, if a statement is proved with lower number of hops, we do not prove it with a higher N . Therefore, the number of hops needed for proving a certain statement is not predefined, and is rather chosen based on the best semantic closeness among all extracted hops at test time. The automated half-proofs obtained from the prover are listed in Tables 6 and 7 in the Appendix.

In Table 2, we present the number of expert-verified half-proofs achieved with different number of hops (N) within the top k ranked results. Note that we exclude successful half-proofs for higher degrees of N if there exists a proof with fewer hops, and we include a proof as long as the verified result is within the top k results. Table 2 shows that the majority of successful half-proofs are achievable in $N \leq 3$.

In Table 3, we present the number of obtained full proofs for 2-hop bidirectional beam search (the best result from Table 1) using GloVe embeddings broken down by logic templates. Although COMET is successful at finding half proofs, the success rate decreases when two implications need to be proven in conjunction. As shown, COMET can (fully) prove 12.8% of the if-then-because com-

¹<https://github.com/atcbosselut/comet-commonsense>

Table 3: Number of End-to-End Proofs per Logic Template for 2-hop bidirectional beam search.

	Logic Template	#Successful	#False Positive	Total Count
Automated proof	orange	1	0	50
	blue	6	1	65
	green	6	2	17
Expert evaluated proof	orange	1	N/A	50
	blue	7	N/A	65
	green	9	N/A	17

Table 4: Number of half-proofs using bidirectional beam search: evaluated by expert human evaluators.

Embedding Space	goal :- action	\neg goal :- state
GloVe	67	17
BERT	52	14
Commonsense	59	12

mands. This emphasizes the necessity of using human conversational interactions for obtaining full proofs. Therefore, if COMET succeeds at proving half of the template, there is a chance to prove the other half with the help of a human user. This is because most of the commands belong to the long tail of tasks for which there is too little knowledge available in any commonsense KB.

Table 4 compares the number of successful proofs obtained using GloVe, BERT and commonsense embeddings. As shown, GloVe embeddings perform better than the others since commonsense and BERT embeddings perform poorly when there is little surrounding context for the words. The returned *objects* by COMET tend to have a few number of tokens, (around 1-2). Therefore, the similarity scores assigned by BERT and commonsense embeddings either result in more false positives or in pruning out good candidates. For example, BERT and commonsense embeddings both tend to assign a lower similarity score to “I sleep” and “sleep” than single token words like “ski” and “spoil”. But this is not an issue with GloVe embeddings. Therefore, in all the other experiments we have used GloVe embeddings. The effect of the embeddings is amplified as the number of hops increases. For example in Table 1, The number of expert-verified 2-hops proofs drops from 21 to 5 if we use BERT embeddings.

4.2 User Study

To assess CLUE’s end-to-end performance, we ran a user study in which human users engaged in a conversational interaction with CLUE and answered its prompts. We collected a total of 700 dialogues

from this study and report the percentage of the if-then-because commands that were successfully proved (end-to-end) as a result of the conversational interactions in Table 5. Our users worked on a total of 288 if-then-because commands that fit the blue, orange, and green logic templates. We had 129 unique commands and 29 participants in the study with at least 2 users per command. The 129 commands used in this study are a subset of the 132 commands (half-proofs) in Table 1. The remaining 3 statements were excluded because either the statements were longer than COMET’s maximum input token size (for the full-proof) or they did not trigger a CORGI/CLUE interaction so it does not make sense to include them in the user-study. Each participant worked on 9-10 statements taking approximately 30-40 minutes to complete them all. The participants were undergraduate, graduate or recently graduated students with a background in computer science or engineering. Please note that no skills beyond reading is required for interacting with CLUE; no particular domain expertise is needed either because most humans possess common sense regarding day-to-day activities.

The users contributed a total number of 70 novel knowledge tuples to our background knowledge base and validated 64.86% of the proofs extracted using our multi-hop prover. CLUE is built on top of CORGI,² and is triggered only if CORGI fails. Therefore, the 19 and 4 commands proved by CLUE are not provable by CORGI; the commands proved by CORGI in each row is indicated in the parentheses. Therefore, CLUE successfully increases the reasoning success rate by 35%. Please note that the absolute gain in terms of success percentage compared with CORGI is 8%.

We did not experience any inconsistencies between user responses in the study because users received the same goal for a given command. It is difficult to generate contradictory proofs when the context is the same. However, proofs among different users do not have to be identical. As long as the proof fulfills the criteria and is “correct” from a commonsense perspective, it holds. Every user has different preferences, linguistic tendencies, and living habits, which can affect the generated proof. Consequently, proofs are tailored to users. Also, showing the five multiple choice options helps users understand what kind of knowledge CLUE is looking for, improving their responses.

²<https://github.com/ForoughA/CORGI>

Table 5: User-study results. Commands not provable by CORGI trigger a CLUE dialog, so commands proved by CLUE are in addition to CORGI’s. The numbers in the parentheses in #proved column is the number of statements that CORGI proved.

Dialog Phase	#Proved	#Tried	Proved/Tried
CORGI	65	288	0.2256
CLUE[orange]	17(+27)	126	0.1349
CLUE[green]	2(+0)	17	0.1176
CLUE[blue]	4(+38)	145	0.0276

5 Related Work

Efforts in developing commonsense reasoning started as early as the foundation of the field of artificial intelligence (Grice, 1975; Winograd, 1972; Davis and Marcus, 2015; Minsky, 1975). Commonsense reasoning is becoming more prominent as computers increase their interactions with us in our daily lives. For example, conversational agents such as Alexa, Siri, Google Home and others have very recently entered our daily lives. However, they cannot currently engage in natural sounding conversations with their human users mainly due to lack of commonsense reasoning. Moreover, they operate mostly on a pre-programmed set of tasks. On the other hand, instructable agents (Azaria et al., 2016; Labutov et al., 2018; Li et al., 2018, 2017b,a; Guo et al., 2018; Mohan and Laird, 2014; Miner and Laird, 2018; Mohan et al., 2012), can be taught new tasks through natural language instructions/demonstrations. One of the challenges these bots face is correctly grounding their natural language instructions into executable commands.

Our approach addresses a new reasoning task proposed by Arabshahi et al. (2021) that contains commands given to an instructable agent satisfying a general template. In contrast to this challenging task and TimeTravel (Qin et al., 2019), most commonsense reasoning benchmarks have traditionally been designed in a multiple choice manner. Moreover, they are not typically targeted at conversational agents. Refer to Storks et al. (2019) and Arabshahi et al. (2021) for a comprehensive list of commonsense reasoning benchmarks.

Our commonsense reasoning engine uses a SOTA neural knowledge model, COMET (Bosselut et al., 2019), as an underlying source of commonsense knowledge. COMET is a framework for constructing knowledge bases from transformer-based language models. In contrast to previous au-

tomatic knowledge base construction methods that rely on semi-structured (Suchanek et al., 2007; Hof-fart et al., 2013; Auer et al., 2007) and unstructured (Dong et al., 2014; Carlson et al., 2010; Mitchell et al., 2018; Nakashole et al., 2012) text extraction, COMET uses transfer learning to adapt language models to generate knowledge graph tuples by learning on examples of structured knowledge from a seed KB. COMET was recently used for persona-grounded dialog for chatbots (Majumder et al., 2020).

6 Conclusions

We introduce the Conversational mUlti-hop rEasoner (CLUE) for commonsense reasoning in conversational agents. CLUE uses a neural commonsense KB and symbolic logic rules to perform multi-hop reasoning. It takes **if-(state)**, **then-(action)**, **because(goal)** commands as input and returns a multi-hop chain of commonsense knowledge, indicating how the action leads to the goal when the state holds. The symbolic logic rules help significantly reduce the multi-hop reasoning search space and improve the quality of the generated commonsense reasoning chains. We evaluate CLUE with a user study with human users.

Acknowledgments

Tom Mitchell is supported in part by AFOSR under grant FA95501710218. Antoine Bosselut and Yejin Choi gratefully acknowledge the support of DARPA under No. N660011924033 (MCS), JD.com, and the Allen Institute for AI.

References

- Forough Arabshahi, Jennifer Lee, Mikayla Gawarecki, Kathryn Mazaitis, Amos Azaria, and Tom Mitchell. 2021. Conversational neuro-symbolic commonsense reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.
- Amos Azaria, Jayant Krishnamurthy, and Tom M Mitchell. 2016. Instructable intelligent personal agent. In *Thirtieth AAAI Conference*.
- Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *ACL*.

- Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.
- Alain Colmerauer. 1990. An introduction to prolog iii. In *Computational Logic*, pages 37–79. Springer.
- Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.
- Xin Dong, Evgeniy Gabilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD Conference*, pages 601–610.
- Herbert P Grice. 1975. Logic and conversation. In *Speech acts*, pages 41–58. Brill.
- Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *NeurIPS*, pages 2942–2951.
- Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.
- Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings. *Convolutional Neural Networks and Incremental Parsing*.
- Igor Labutov, Shashank Srivastava, and Tom Mitchell. 2018. Lia: A natural language programmable personal assistant. In *Proceedings of 2018 EMNLP: System Demonstrations*, pages 145–150.
- Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of 2016 EMNLP*, pages 1192–1202.
- Toby Jia-Jun Li, Amos Azaria, and Brad A Myers. 2017a. Sugilite: creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI Conference*, pages 6038–6049. ACM.
- Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Wanling Ding, Tom M Mitchell, and Brad A Myers. 2018. Appinite: A multi-modal interface for specifying data descriptions in programming by demonstration using natural language instructions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 105–114. IEEE.
- Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A Myers. 2017b. Programming iot devices by demonstration using mobile apps. In *International Symposium on End User Development*, pages 3–17. Springer.
- Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of 2016 EMNLP*, pages 2122–2132.
- Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian McAuley. 2020. Like hiking? you probably enjoy nature: Persona-grounded dialog with commonsense expansions. In *Proceedings of 2020 EMNLP*, pages 9194–9206.
- Aaron Mininger and John E Laird. 2018. Interactively learning a blend of goal-based and procedural tasks. In *Thirty-Second AAAI Conference*.
- M Minsky. 1975. A framework for representing knowledge. the psychology of computer vision, s. 211 277, new york.
- Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.
- Shiwali Mohan and John Laird. 2014. Learning goal-oriented hierarchical tasks from situated interactive instruction. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- Shiwali Mohan, Aaron H Mininger, James R Kirk, and John E Laird. 2012. Acquiring grounded representations of words with situated interactive instruction. In *Advances in Cognitive Systems*.
- Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of 2014 EMNLP*, pages 1532–1543.
- Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. 2019. Counterfactual story reasoning and generation. In *Proceedings of 2019 EMNLP-IJCNLP*.

- Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of AAAI*, volume 33, pages 3027–3035.
- Robyn Speer, Joshua Chin, and Catherine Havasi. 2016. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*.
- Shane Storks, Qiaozi Gao, and Joyce Y Chai. 2019. Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*.
- Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.
- Terry Winograd. 1972. Understanding natural language. *Cognitive psychology*, 3(1):1–191.

Appendix

Multi-hop Prover Details

Prolog is a logic programming language that consists of a set of predicates. A predicate has a name (functor) and a set of $N > 0$ arguments. For example, `get(i, work, on_time)` is a predicate with functor `get` and 3 arguments. Predicates are defined by a set of logical rules or Horn clauses (Head $:-$ Body) and facts (Head), where Head is a predicate, Body is a conjunction of predicates, and $:-$ is logical implication. Prolog uses backward-chaining to logically reason about (prove) an input query, represented by a predicate. From a high level, a proof consists of a chain of logical facts and rules available in the background KB. Inspired by this prover, we extend this to free-form text proofs in this paper.

This chain-structured definition of a proof is inspired by a Prolog proof. A knowledge tuple here is analogous to a logical rule in Prolog (Head $:-$ Body) where the *subject* and *object* correspond to either the Body, or the Head depending on the *relation*. We categorize COMET relations into two classes, namely pre-conditions and post-effects. If the *relation* is a post-effect, the *object* is the Head and the *subject* is the Body. If the *relation* is a pre-condition, the reverse is true. For example, the relations “Because I wanted” and “is used for” in Fig 4 are post-effects, whereas the relations “Before I needed” and “requires action” are pre-conditions. The *subject* (Body) consists of a single predicate (instead of a conjunction of predicates in Prolog). Since there is no conjunction in the Body of the rules, the logical proof reduces to a chain (as opposed to a proof tree in Prolog). Moreover, the semantic closeness of o_{i-1} and s_i is inspired by the unification operation (Colmerauer, 1990) in Prolog and is analogous to the soft unification operation of CORGI’s neuro-symbolic theorem prover. It is worth noting that this analogue falls short of Prolog’s variable grounding and is an interesting avenue for our future work.

Dialogue System Details

The control flow of CLUE’s dialog system in Figure 2 is shown in detail in Figure 6. The dialog system uses the logic templates and the results returned by COMET to interact with the human user and ask questions.

For example, consider proving the first implication of the blue template (`\neg goal $:-$ state`

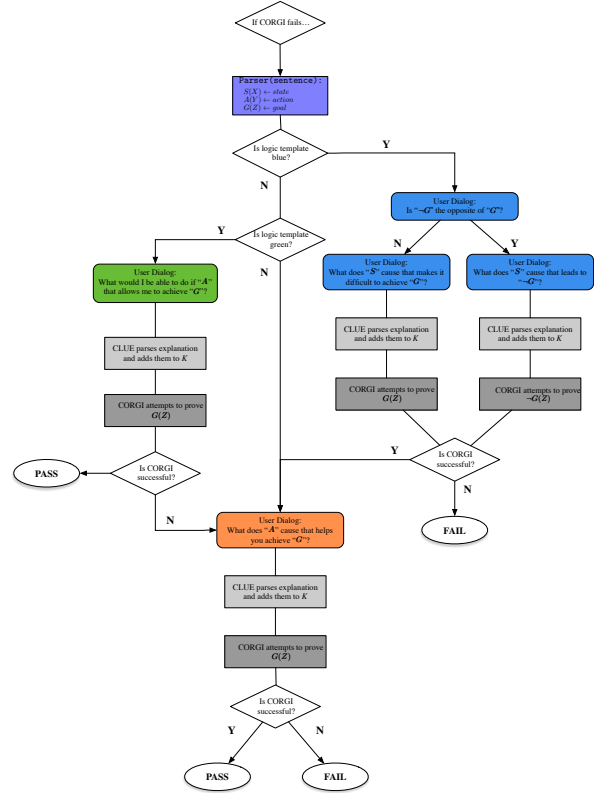


Figure 6: Full Control Flow Diagram for the Dialogue System in Figure 2.

) for the command “If it snows tonight then wake me up early because I want to get to work on time”. CLUE first tries to find the negation of the goal (`get to work on time`) by querying `COMET(r, goal)` for relations (r) indicating negation (such as `NotCapableOf` and `NotIsA`). CLUE then picks the highest ranking returned COMET statement (I am late) and asks the user “is $\langle \neg \text{goal} \rangle$ the opposite of $\langle \text{goal} \rangle$?(y/n)” (Is “I am late” the opposite of “I get to work on time”? (y/n)). If the user responds ‘yes’, then CLUE asks “what does $\langle \text{state} \rangle$ cause that leads to $\langle \neg \text{goal} \rangle$?” (what does “it snows tonight” cause that leads to “I am late”?) and expects an explanation from the user in response. If the user responds ‘no’ to the first question, then CLUE asks “what does $\langle \text{state} \rangle$ cause that makes it difficult to achieve $\langle \text{goal} \rangle$?” (What does “it snows tonight” cause that makes it difficult to achieve “I get to work on time”?) and expects an explanation from the user in response. The reason for querying COMET for the negated goal is that negation in Prolog is implemented based on negation as failure. Therefore, to be consistent, CLUE converts the goal to its negated statement and proves that instead.

Extended Experiments

Tables 6 and 7 show 32 examples of the 67 expert-verified half-proofs reported in Table 1. These half-proofs are obtained using bidirectional 2-hop beam search and logically prove the goal :- action implication. Recall that `action` refers to the Then-portion of the command and `goal` refers to the Because-portion of the command. The embeddings used to measure semantic closeness in the prover is GloVe. The second intermediate hop obtained by CLUE’s prover is shown in column 4 of the tables. Each row indicates a successfully half-proved if-then-because command along with its intermediate commonsense reasoning hop. Let us explain the proofs through an example. Consider the example on Row 5 of Table 6: “if the temperature is going to be below 40 degrees in the evening but above 40 degrees in the morning then remind me to bring a jacket because i want to stay warm on my commute”. The relations (column 3) and COMET outputs (column 4) for this example indicate that the `action` of reminding the user to bring a jacket *causesDesire* for the user to *wear the jacket* which is a *Prerequisite* for the goal of staying warm on the user’s commute. In other words CLUE is able to understand why reminding someone to bring a jacket would allow them to stay warm (because they would be able to wear the jacket). This is a logical proof for the implication goal :- action according to the proof definition of the prover proposed in this paper. For more examples, please refer to Tables 6 and 7.

Table 6: Half Proofs goal :- action , Part 1. The half-proofs are obtained using our proposed prover and are obtained using bidirectional pruning. The input if-then-because commands are listed in the first column. The semantic closeness scores are obtained with GloVe embeddings. (r_1, r_2) on the third column are relations tuples on the final proof path obtained from the $COMET(r_1, action)$ and $COMET(r_2, goal)$ queries.

Commands	Semantic Closeness Score	(r_1, r_2)	$(COMET(r_1, action), COMET(r_2, goal))$
if i have an upcoming exam then remind me to prepare 3 days ahead because i want to prepare for it	1.0	(CausesDesire, HasSubevent)	(prepare for exam, prepare for exam)
if the air temperature is forecast to be warmer than 70 tonight then remind me to turn on the air conditioner because i want to stay cool	1.0	(CapableOf, CreatedBy)	(cool air, cool air)
if i haven't been to the gym for more than 3 days then remind me to go to the gym because i want to stay fit	0.9220791	(CapableOf, HasPrerequisite)	(work out, work out regularly)
if i am going to school then remind me to take my office keys with me because i want to be able to unlock my office door	1.0	(CausesDesire, CausesDesire)	(go to work, go to work)
if the temperature is going to be below 40 degrees in the evening but above 40 degrees in the morning then remind me to bring a jacket because i want to stay warm on my commute	1.0	(CausesDesire, Prerequisite)	(wear jacket, wear jacket)
if i get an email from my boss about our upcoming deadline then notify me about the email because i want to read the email	1.0	(Causes, CausesDesire)	(open email, open email)
if my calendar is clear today, then remind me to go to gym in the afternoon, because i want to keep myself healthy	1.0	(Desires, Desires)	(exercise, exercise)
if i start using google maps to go home then tell alexa to turn on the heat because i want my home to be warm when i arrive	1.0	(Desires, CreatedBy)	(heat, heat)
if there is heavy traffic in the route that i use to office then remind me to leave early because i want to reach office on time	0.9641539	(CausesDesire, CausesDesire)	(go to work early, go to work)
if papers related to what i'm working on are posted on the proceedings of any nlp or ml conference then tell me about the papers immediately because i want to stay up-to-date on current research	1.0	(HasSubevent, CausesDesire)	(read, read)
if i'm not in bed at 12pm then remind me to go to bed because i want to go to bed early	1.0	(Desires, CreatedBy)	(sleep, sleep)
if a new paper related to what i'm working on is posted on arxiv then notify me about the new paper immediately because i want to stay up-to-date on current research	1.0	(HasSubevent, CausesDesire)	(read, read)
if the price of something i want to buy drops then notify me about the price drop because i want to buy it when the price is low	0.9652006	(MotivatedByGoal, CausesDesire)	(buy something, buy something else)
if there is heavy traffic on my current commute path then give me a less congested path because i want to minimize my driving time	1.0	(MotivatedByGoal, HasSubevent)	(i drive fast, i drive fast)
if i have more than ten unread emails then set an one hour email replying event on calendar because i want to be responsive to emails	1.0	(CausesDesire, HasPrerequisite)	(send email, send email)
if i have set an alarm for taking my pills then make sure the alarms are off after i have finished my pills because i want to make sure i don't have false alarms	1.0	(HasPrerequisite, HasPrerequisite)	(turn off alarm clock, turn off alarm clock)
if the weather temperature forecast in the next 10 days is above 30 degrees celsius then remind me to turn the heater off because i don't want to make the house warm	0.93141675	(MotivatedByGoal, HasSubevent)	(it be cold, it get cold)
if i am driving home and i have an email about a grocery shopping list then remind me to stop at the grocery store because i want to buy the items on my grocery shopping list	0.99999994	(CapableOf, UsedFor)	(buy grocery, buy grocery)
if there is a sale on sketchbooks between now and august then notify me about the sale because i need to get sketchbooks for my fall class	0.8800874	(CausesDesire, HasPrerequisite)	(buy something, buy them)
if i have set an alarm for a time between 2am-8am on weekends then notify me that i have an alarm set because i want to correct the alarm	0.93820596	(CreatedBy, CreatedBy)	(turn off alarm, turn off alarm clock)
If the air temperature is forecast to be colder than 40 degrees then tell me to close the windows because I want to stay warm	1.0	(Desires, HasPrerequisite)	(get warm, get warm)
if the air temperature is forecast to be warmer than 70 tonight then remind me to turn on the air conditioner because i want to stay cool	0.9137391	(UsedFor, UsedFor)	(cool down room, cool down)
if there is a natural disaster back at home then remind me to donate money because i want to give back to my community	0.9001999	(CapableOf, Causes)	(i donate money, i give money)

Table 7: Half Proofs goal :- action, Part 2. The half-proofs are obtained using our proposed prover and are obtained using bidirectional pruning. The input if-then-because commands are listed in the first column. The semantic closeness scores are obtained with GloVe embeddings. (r_1, r_2) on the third column are relations tuples on the final proof path obtained from the $COMET(r_1, action)$ and $COMET(r_2, goal)$ queries.

Commands	Semantic Closeness Score	(r_1, r_2)	$(COMET(r_1, action), COMET(r_2, goal))$
if an author i like is doing a reading in my city then let me know about the reading because i want to see them	0.8933883	(HasPrerequisite, HasPrerequisite)	(go to bookstore, go to store)
if i have an upcoming bill payment then remind me to pay it because i want to make sure i avoid paying a late fee	1.0	(CausesDesire, Desires)	(pay bill, pay bill)
If it snows tonight then wake me up early because I want to get to work early	0.95353657	(Causes, Causes)	(i go to work early, get to work early)
if i have a meeting then remind me fifteen minutes beforehand because i want to be prepared for the meeting	0.9151366	(ReceivesAction, UsedFor)	(prepare for meet, prepare for)
if we are approaching fall then remind me to buy flower bulbs because i want to make sure i have a pretty spring garden	1.0	(CausesDesire, CausesDesire)	(plant flower, plant flower)
if i receive emails about sales on basketball shoes then let me know about the sale because i want to save money	0.9345853	(CausesDesire, UsedFor)	(buy something, i buy something)
if i receive an email related to work then notify me about the email immediately because i want to stay on top of my work-related emails	1.0	(CausesDesire, CausesDesire)	(open email, open email)
if the forecast is dry and greater than 50 degrees f on a weekend day then remind me to line-dry the laundry because i want our clothes to smell good	1.0	(MotivatedByGoal, CapableOf)	(smell good, smell good)
if i have more than three hours meeting on my calendar for a day then remind me to relax for an hour in the evening because i want to achieve work life balance	0.96545255	(CapableOf, CapableOf)	(make me feel good, make me look good)