PROVER: Generating Intermediate Steps for NLI with Commonsense Knowledge Retrieval and Next-Step Prediction

Deepanway Ghosal[1], **Somak Aditya**[2], **Monojit Choudhury**[3]

[1] ISTD, Singapore University of Technology and Design, [2] Department of CSE, IIT Kharagpur
[3] Turing India, Microsoft

deepanway_ghosal@mymail.sutd.edu.sg, saditya@cse.iitkgp.ac.in, monojitc@microsoft.com

Abstract

The Natural Language Inference (NLI) task often requires reasoning over multiple steps to reach the conclusion. While the necessity of generating such intermediate steps (instead of a summary explanation) has gained popular support, *it is unclear how to generate such steps without complete end-to-end supervision and how such generated steps can be further utilized*. In this work, we train and enhance a sequence-to-sequence next-step prediction model with external commonsense knowledge and search to generate intermediate steps with limited next-step supervision. We show the correctness of such generated steps through human verification, on MNLI and MED datasets (and discuss the limitations through qualitative examples). We show that such generated steps can help improve end-to-end NLI task performance using simple data augmentation strategies. Using a CHECKLIST dataset for NLI, we also explore the effect of augmentation on specific reasoning types. The code and human-evaluation dataset is available at https://github.com/deepanwayx/prover.

1 Introduction

Complex NLP tasks such as Natural Language Inference (NLI) often requires reasoning over multiple steps using multiple facts and implicit commonsense knowledge (Trivedi et al., 2020; Sap et al., 2019; Camburu et al., 2018). For such cases, it has been long argued that (Lipton, 2018), the state-of-the-art models should also output some sort of explanation (such as intermediate steps or a textual explanation) alongwith the final answer. The opaque performance and poor out-of-distribution generalization performance of Transformers-based models (Kaushik et al., 2019; Ribeiro et al., 2020) have refuelled this discussion. However, it is unclear how these intermediate steps can be generated for unconstrained natural language Premise-hypothesis pairs (such as in crowd-sourced NLI datasets) as it is non-trivial to collect crowd-sourced fine-grained explanations or generate them synthetically. Most importantly, it is unclear how such steps can be utilized further for the NLI task.

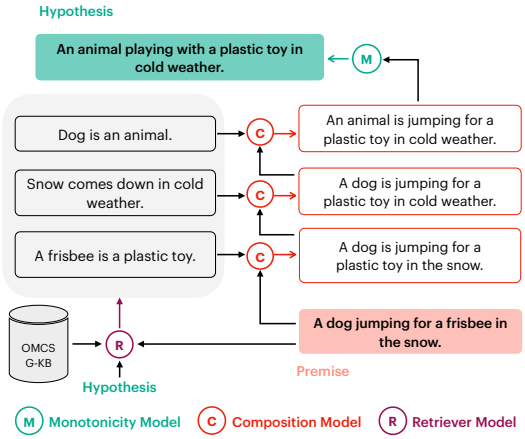

Figure 1: We illustrate a type of proof algorithm that we explore in this paper: *proof search with facts* (§4.2).

Crowd-sourced collection of intermediate steps (or explanations) (Camburu et al., 2018) comes with complications, as human-written explanation can be subjective, and it is hard to automatically verify or utilize such explanations. Recently, researchers have explored synthetic generation of intermediate steps (or *proof* trees) templated from first order logic theories (Clark et al., 2020; Tafjord et al., 2021; Saha et al., 2020). Here, the objective is to test whether Transformer models can perform deductive reasoning over natural language statements. Provided their examples come from an underlying symbolic system with closed world rules and facts, it is unclear how this strategy of generating proof tree can be extended to unconstrained natural language premise-hypothesis pairs. Similarly, EntailmentBank (Dalvi et al., 2021) is another dataset to benchmark performance of recent multi-hop prover models. EntailmentBank provides a set of supporting facts such that no external facts are required to prove the given hypothesis. Here, we consider

Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 872–884
November 1–4, 2023.

Premise (P)	Hypothesis (H)	Proof	Correctness C1	Correctness C2	Minimality M1	Minimality M2
The wind propels a sailing ship on a group of cruisers.	There are many boats out.	**P** → **I1:** A group of cruisers are in the water. → **I2:** A group of boats are in the water. → **H**	✔	✔	✔	✔
A young girl wearing a red shirt and cap smiling and holding a small toy is standing in front of a group of children playing behind her.	The young girl is wearing a red shirt.	**P** → **I1:** A young girl in a red shirt and cap is in front of others. → **I2:** The young girl is wearing a cap and red shirt. → **H**	✔	✔	✖	✖
Numerous customers browsing for products in a market.	People are shopping.	**P** → **I1:** A group of people are shopping. → **I2:** There are a bunch of people shopping. → **H**	✔	✔	✖	✔
A gentleman with his eyes closed playing an old fluglehorn into a microphone.	A man plays an instruement with his eyes closed.	**P** → **I1:** The gentleman is listening to music. → **H**	✔	✖	✖	✔

Table 1: Correctness and minimality of proofs as outlined in §2.

proving hypothesis with an open-ended context, as is typical in NLI or QA datasets.

We assume that an ideal NLI system reaches a series of intermediate conclusions to derive the final conclusion (entailment/contradiction/neutral). Such intermediate steps may require commonsense knowledge external to the input. These intermediate steps and external facts constitute a *proof*. Since such *proof*s can widely vary linguistically, logically and in length; to make verification (without groundtruth) and generation easier, we impose various constraints of *correctness*, *minimality*, and *atomicity* on what we expect as natural language *proof*s for a given premise-hypothesis pair. We primarily use human verification for evaluation. As human annotation of groundtruth proofs for NLI examples is non-trivial, we utilize single-step supervision to train a T5 encoder-decoder model (Raffel et al., 2019) on various available entailment datasets; learning various aspects of proof generation from SNLI, Monotonicity Entailment Dataset, and Entailment Bank. To enrich single-step generation with external knowledge, we build a fact retriever and sentence composition model; that retrieves facts from external commonsense knowledge bases and learns to deduce new facts. We then explore search-based methods that utilizes the retrieval-augmented T5 model to generate multiple-step *proof*s. Manual verification results show the efficacy of the proof generation process. We use these generated steps as additional training data and show improvement in end-to-end NLI task performance.

Our **contributions** include, 1) using next-step supervision to train a general-purpose T5 encoder-decoder based entailment model to generated entailment chains. 2) We use a fact retriever, a fact composition model; and search to generate sequence of commonsense-augmented intermediate steps. We use human verification metrics to show the correctness of the generated proofs. Lastly, 3) we also show that augmenting such proofs during training can help enhance an NLI system's performance on MNLI, and MED (under low-training data regimes), 4) while benefitting examples of specific reasoning types (shown through LoNLI).

2 Proofs in the NLI Context

Given a natural language premise-hypothesis pair, an ideal NLI system should follow a logical sequence of steps to determine whether the hypothesis is entailed by, contradicts, or is neutral w.r.t the premise. We call this logical sequence of steps as *proofs*. However, as Dalvi et al. (2022) observes, such *proofs* do not come with guarantees as given by formal axiomatic systems. To make generation and verification easier, we resort to defining some logical properties that valid *proofs* should have.

Given a premise (P), a hypothesis (H), and an implicit knowledge base KB [1], an NLI *proof* $\langle P, H \rangle$ (or $\langle P, \neg H \rangle$ for contradiction) is a sequence of sentences $(Y_1, \ldots, Y_m)$, where Y_j is either an inferred intermediate step (denoted by I_j), or an external fact or rule (denoted by F_j). Proof $\langle P, H \rangle$ can also be thought of a postorder traversal of an Entailment Tree (Dalvi et al., 2022). A valid *proof* should satisfy the following properties: i) (Correctness 1) each step is either a generic rule, or a fact which is a entailed by the premise; ii) (Correctness 2) hypothesis (or negated hypothesis) is a valid entailment of premise and intermediate steps; iii) (Minimality 1) each proof upto Y_j (including H) should be minimal proof for $\langle P, Y_j \rangle$, iv) (Minimality 2) the sentences in the intermediate step (I_j) should not be trivially decomposable (us-

[1] Following (Dagan et al., 2005), we assume KB encodes commonly assumed knowledge as facts and rules.

ing common linguistic or logical constructs)[2] and consecutive inferred steps should be sufficiently different, and v) (Order) a step can only be generated with the help of previous steps.

We provide few example proofs in Table 1, highlighting the properties of correctness and minimality. These constraints motivate our method to generate *proof*s for arbitrary P-H pairs with only next-step supervision. In Figure 1, we show an example proof which uses external (commonsense) facts (or rules) in sequence. For neutral cases, such properties are hard to define. Inspired from Kumar and Talukdar (2020), we proposed a method for intermediate step generation of neutral instances which is used only for data augmentation experiments.

3 Provers: Generating Proofs with Next-Step Supervision

A typical possibility to build provers is by training on an NLI dataset with groundtruth proofs, collected using crowd-sourcing or created using formal logic. However, crowd-sourced explanations are highly subjective and diverse; hence not easily verifiable. Similarly, following ProofWriter, generating formal logic-based proof is possible but hard to scale for arbitrary *P-H* pairs. We instead rely on various NLI datasets, from which models can learn multiple ways of generating entailments, contradictions, and intermediate steps. We then enhance such provers with facts retrieved from knowledge bases and sentence composition methods. We use these techniques effectively to search for more generalized multiple-step proofs. We start from the premise *P* and generate the steps of the proof recursively with the PROVER model. We use the term PROVER to collectively denote the models in §3.1 and §3.2 as discussed next.

3.1 Multi-Task Supervision

We train a single T5-Large model with various objectives to generate inferences from the premise. The objectives are specified by prefix tokens in the input text. The objectives are as follows.

Entailed Sentence Generation: The model is trained to generate possible entailments E from the premise P. The entailment instances of the SNLI dataset (Bowman et al., 2015) is used for training. The hypothesis H is considered as E during training. The input to the model is `entail: P` and the output to be generated is `H`.

Contradictory Sentence Generation: The model is trained to generate possible contradictions rom the premise P. We use the relevant instance pairs in the SNLI dataset for this objective. The input to the model is `contradict: P` and the output to be generated is the contradictory hypothesis `H`.

Monotonic Sentence Generation: The model is trained to generate monotonic inferences M from the premise P. The input is `monotonic: P` and the output to be generated is `M`. We use the Monotonicity Entailment Dataset (MED) (Yanaka et al., 2019) for this objective.

We merge and shuffle instances from the respective datasets to ensure that training is performed for all the objective functions simultaneously.

3.2 Fact Retriever and Sentence Composition

Fact Retriever The generator model (§3.1) can not generate proofs, which need reference to some external facts, or commonsense knowledge. Such knowledge may not be readily available in the generator model. It is thus necessary to assist the proof generation algorithm with relevant factual knowledge to generate accurate and complete proofs.

We use sentences in Open Mind Commonsense (OMCS) (Singh et al., 2002) and GenericsKB (Bhakthavatsalam et al., 2020) as the knowledge base (KB). The sentence embedding model *all-mpnet-base-v2* (Reimers and Gurevych, 2019) is used to retrieve facts for a given *(premise, hypothesis)* pair from the KB. For a sentence s, we retrieve the facts F from KB based on highest embedding cosine similarity in the following way: noun tokens – n_p, n_h are extracted from the premise and the hypothesis. n_p, n_h are divided into small groups of related words using clustering with word embeddings. For instance, the following groups are created for the example in Figure 1: {dog, animal}, {snow, cold}, {frisbee, toy, plastic}. Each of the groups is then merged together in a single string s for performing retrieval. We deduplicate semantically close facts before the next stage (§4.2).

Sentence Composition We train another T5-Large model to generate compositions from a pair of input sentences. The model is trained on sentence triplets from the Entailment Bank (Dalvi et al., 2021)

[2]M1 captures redundancy (see E4 in Tab. 1). M2 captures some aspects of *atomicity*. Sentences should not be compound in nature. We prefer "John is going to Paris. Mia is going to Paris." over "John and Mia are going to Paris". It also entails that intermediate conclusions should correspond to a semantic frame (which can not be trivially decomposable without losing context). However the latter is quiet hard to verify.

and RuleTaker (Clark et al., 2020) datasets. Few example triplets $\langle S_1, S_2, S_3 \rangle$ are: ⟨Bob is green, All green people are rough, Bob is rough⟩, and ⟨Eruptions produce ash clouds, Ash blocks sunlight, Eruptions block sunlight⟩. Here, input S_1 and S_2 can be composed to conclude output S_3. The sentence composition model, the fact retriever model and the monotonic entailment model is used to generate proofs (§4.2).

4 Proof Generation

We generate proofs with two different methods as described below, and provide some examples of generated proofs in Figure 1 and Table 2.

4.1 Unconstrained Proof Search

The *entailment* and *monotonic sentence* generation setup use the premise P to create possible inference chains. We use this generation setup recursively with level/beam search to find multistep proofs (I_1, I_2) as follows: i) Given P, we first generate one-step implications $\hat{i_1}$ from the T5 generator. We denote the n closest implications to the hypothesis H as the filtered set i_1. The closeness is computed using cosine similarity with H using the *all-mpnet-base-v2* (Reimers and Gurevych, 2019) model. ii) i_1 is used to generate the next set of implications $\hat{i_2}$, which is filtered further to obtain i_2. iii) The n implications in i_2 and their respective source sentences in i_1 form the multistep proof set (I_1, I_2) for level search. The top n implications (according to closeness with H) from the merged set of i_1 and (i_1, i_2) form the proof set for beam search.

We use the terminology *unconstrained proof search* for the above chaining algorithm with sentence embedding based closeness measure. We use $n = 10$ in our experiments. The generation and filtering process can be performed repeatedly to form proofs with more steps: $(I_1, I_2, ..I_m)$. However, we observe diminishing results after I_2, as steps tend to become repetitions of each other.

4.2 Proof Search with External Facts

Given a premise, hypothesis pair (P, H), we denote the facts retrieved from KB as F. Let F consist of m distinct facts $\{F_1, F_2, .., F_m\}$. We now use the *sentence composition* model in two stages to generate the proof: **i) Fact Filtering Stage**: We compose the premise with the retrieved facts individually. The fact is used in the next stage if the composition is closer to the hypothesis than the premise and the hypothesis is entailed by the composition; **ii) Search Stage**: The selected facts are ranked based on their corresponding composition's distance from the hypothesis. The facts are then iteratively used with the *sentence composition* model and *monotonic sentence* generator models. For each step, we ensure that the step is closer to the hypothesis than the preceding step. The detailed algorithm is presented in Algorithm 1.

Algorithm 1: Proof Search with Facts Algorithm

Given: 1) $\langle P, H \rangle$ pair and knowledge base $\mathcal{KB}$.
2) Entailment Predictor Model $E(x, y)$ = RoBERTa Large MNLI
3) Next-step Entailment Model $NS(x)$ = T5 Large in §3.1
4) Distance $D(x, y)$ = cosine similarity using *all-mpnet-base-v2*. (Higher scores indicate closer pairs.)
Output: Proof Steps $\langle I_1, \ldots, I_m \rangle$
Retrieve $F = \{F_1, .., F_m\}$ from $\mathcal{KB}$ based on §3.2
Fact Filtering Stage:
Useful Facts U, *Useful Facts Distances* U_d = [ ], [ ]
// $\oplus$ denotes appending to list.
for F_i *in* F **do**
  Compose $P, F_i \rightarrow I_i$
  if $D(I_i, H) > D(P, H)$ & $E(I_i, H) =$ *entailed* **then**
    $U \leftarrow U \oplus [F_i]$; $U_d \leftarrow U_d \oplus D(I_i, H)$
end
Sort U based on higher to lower U_d
Initialize Step $S = P$, score $= 0$, All Steps $= [P]$
Search Stage:
for F_i *in* U **do**
  $I_i \leftarrow compose(S, F_i)$
  Next step $M_i \leftarrow NS(I_i)$
  $d_1,\ d_2,\ d_3 = D(S, H),\ D(I_i, H),\ D(M_i, H)$
  if $d_2 > d_1$ & $d_2 > d_3$ & $d_2 > score$ **then**
    $S \leftarrow I_i$
    $\mathbb{S} \leftarrow \mathbb{S} \oplus [I_i]$
    score $\leftarrow d_2$
  else if $d_3 > d_1$ & $d_3 > d_2$ & $d_3 > score$ **then**
    $S \leftarrow M_i$
    $\mathbb{S} \leftarrow \mathbb{S} \oplus [M_i]$
    score $= d_3$
end
$\mathbb{S} \leftarrow \mathbb{S} \oplus [H]$
Proof $\leftarrow \mathbb{S}$

5 Experimental Results

We perform two sets of experiments – i) we use human evaluation to evaluate different aspects of the proofs generated from our proposed method, and ii) we use the generated proofs as additional labeled data for NLI and analyze its effect on the NLI classification task performance. We compare our results against the baseline ENTAILER (Tafjord et al., 2022) model (a representative of Class (B) models, see §6 Fig. 2). We do not compare with Class (A) algorithms, which require a pre-specified complete set of facts and rules (such as METGEN,

#	Premise (P) & Hypothesis (H)	Proof	Remarks
(i)	**P:** A female guitarist is playing on stage. **H:** A woman is playing her instrument.	**F1:** A guitar is an instrument. **F1** & **P** → **Monotone** → **I1** A woman is playing an instrument. **I1** → **H**	Correct proof with fact composition.
(ii)	**P:** Bicyclist ride the course near the ocean as the day comes to an end. **H:** The cyclist was riding near the ocean at sunset.	**F1:** Sunsets can happen at the end of the day. **F1** & **P** → **I1:** A cyclist ride the course near the ocean during the sunset. **I1** → **H**	Correct proof with fact composition.
(iii)	**P:** A baby girl and little boy are standing next to a guitar and a drum. **H:** The girl is near an instrument.	**F1:** A drum is a percussion instrument. **F1** & **P** → **I1:** A baby girl and little boy are standing next to a percussion instrument. **I1** → **I2:** A baby girl standing next to a percussion instrument. **I2** → **H**	Correct proof with fact composition.
(iv)	**P:** An old woman in a white hat and purple and blue clothes is sitting down by a wooden building. **H:** There is a building.	**P** → **I1:** A woman is sitting by a wooden building. **I1** → **I2:** The building is made out of wood. **I2** → **H**	Unconstrained proof without fact composition.
(v)	**P:** A professional swimmer spits water out after surfacing while grabbing the hand of someone helping him back to land. **H:** A person is swimming.	**F1:** A hand is part of the arm. **F1** & **P** → **Monotone** → **I1** A person is swimming. **I1** → **H**	An unrelated fact is retrieved that is not useful for the proof.
(vi)	**P:** A black-haired man is entertaining a crowd with a hula hoop. **H:** A man has black hair.	**P** → **I1:** A man with black hair is performing. **I1** → **H**	**P** and **H** are lexical or syntactic paraphrases.
(vii)	**P:** A person in a green robe sits on a couch with a blanket. **H:** A person in a robe sits on a couch.	**P** → **I1:** A woman in a green robe sits on a couch. **I1** → **I2:** A girl in a green robe sits on a couch. **I2** → **H**	Introduction of undue specialization or hallucinations.
(viii)	**P:** A man gets a skateboard up on a big rock. **H:** The skateboarder is near a rock.	**P** → **I1:** A person puts his skateboard on a rock. **I1** → **I2:** Person skateboarding on a rock. **I2** → **H**	Requires spatial reasoning about physical objects skateboard and rock.

Table 2: Examples of generated proofs. Analysis can be found in §5.1. Incorrect/unrelated snippets are shown in red.

IRGR, NLPROOFS)[3].

5.1 Qualitative Analysis.

We perform a qualitative analysis of generated proofs and show some examples in Table 2. We show correct proofs with fact composition in examples (i) - (iii). The first proof uses a monotonically generated sentence as one of the intermediate steps. In example (ii) the fact: *Sunsets can happen at the end of the day* is used for proving the hypothesis. For example (iii), the fact: *A drum is a percussion instrument* is retrieved and used for the proof. A more concise proof could have used the alternative fact: *A drum is an instrument*. However, this alternative fact does not appear in our retrieval corpus of OMCS and GenericsKB. Hence, some proofs have constituting facts that are somewhat over-informative. Finally, we show an example of unconstrained proof without facts composition in example (iv). We observe that fact composition provides consistently superior results, especially when ontological knowledge (IsA, HasA, is-part-of relations) is involved.

We also perform error analysis and show some common error patterns in Table 2. In example (v), an unrelated fact is retrieved which is not useful for the proof. The premise and the hypothesis are lexical/syntactic paraphrases in example (vi). Hence, no intermediate steps are required as proof. In example (vii), undue specialization or hallucinations are introduced in the intermediate steps, as the word *person* is changed to a *woman* and *girl*. Example (viii) requires spatial reasoning about physical objects *skateboard* and *rock*.

5.2 Human Verification of Proofs

We perform human verification on a subset of generated proofs in the SNLI dataset. We train four CS graduate students (trained in NLP) with explicit instructions (in Appendix). We select top-2 proofs from *unconstrained proof search* (PROVER UPS) with beam search for 500 randomly selected SNLI entailment instances – resulting in a total of 1000 proofs. We also evaluate 500 proofs from the *proof search with facts* (PROVER PSF) method and the ENTAILER method. Human annotators score each proof based on the following: (i) correctness, (ii) minimality, (iii) number of useful facts, and (iv) if the hypothesis follows from the premise and retrieved facts. The (iii), (iv) scores are judged for the PROVER PSF and ENTAILER methods. We refer the reader to Appendix E for detailed instructions for human verification.

We normalize the scores on a scale of 0-100

[3]Our experiment with premise and retrieved facts for NLPROOFS show inferior performance, which is expected.

Method	Correct	Minimal	Useful	Follows
PROVER UPS	69.45	75.32	-	-
PROVER PSF	**84.66**	74.66	**45.18**	**82.54**
ENTAILER	74.23	72.66	31.81	77.27

Table 3: Human verification judgements. Numbers are shown in % normalized between 0-100.

and show the results in Table 3. Human verification results in a normalized correctness score of 69.45% for PROVER UPS and a significantly improved 84.66% for PROVER PSF. The minimality scores for both methods are around 74-75%. The Useful metric is built in a way such that the presence of redundant facts is penalized. We find that 45.18% of the facts are useful [4], suggesting that proofs contain redundant facts and compositions. Overall, the retrieved facts can be used to conclude the hypothesis in 82.54% cases. In contrast, the ENTAILER method achieves correctness of 74.23%, which is around 5% better than PROVER UPS, but still much lesser than PROVER PSF. The minimality score of ENTAILER is around the same range as our models, but the Useful and Follow score for is much poorer compared to PROVER PSF.

5.3 Usefulness of the Prover for NLI Tasks

As illustrated in §3.1, the PROVER model is trained on various objective functions to generate inferences from the premise. Here, we show that these inferences could be used to improve end-to-end NLI task accuracy. We consider the *entailed*, *contradictory*, *neutral*, and *monotonic sentence generation* setup of §3.1 and apply it on the premises of the SNLI dataset to create corresponding inferences. For a particular premise P, the inferences generated from the above four setups are considered to have a label of entailment, contradiction, neutral, and entailment, respectively.

The inferences generated from the PROVER are considered as additional labeled data for supervised learning in NLI tasks. In particular, we finetune RoBERTa-Large model on the MNLI (Williams et al., 2018) dataset in a low-data regime. The models are trained with either (i) MNLI-only data or (ii) a mix of MNLI and PROVER generated data. We also benchmark the ENTAILER model as the main comparative baseline where (iii) mix of MNLI and ENTAILER generated data is used for training.

We evaluate the models on MNLI, Monotonicity Entailment Dataset (MED) and LoNLI (Tarunesh et al., 2021). The incorporation of PROVER generated data helps in improving performance across most settings in these datasets. We also observe that PROVER generated data is significantly better than ENTAILER generated data in MED and LoNLI. We also train and evaluate DeBERTa models for which the results are shown in Appendix D.

[4] our retriever is frozen and the commonsense knowledge base may not contain the appropriate useful fact

Trained On			MNLI	MED
MNLI %	Proofs %	Proofs From		
1	0	-	84.47	40.02
1	1	ENTAILER	84.73	39.10
1	1	PROVER	**85.22**	**42.89**
5	0	-	87.18	41.11
5	5	ENTAILER	86.93	39.72
5	5	PROVER	**87.58**	**42.91**
10	0	-	87.44	43.01
10	10	ENTAILER	88.02	40.95
10	10	PROVER	**88.48**	**44.30**

Table 4: F1 scores for MNLI (val-matched) and MED datasets with the RoBERTa-Large model. Numbers on the **MNLI %** column indicate the percentage of the MNLI train set instance used for training. Numbers on the **Proofs %** column indicate the equivalent number of generated instances used for training. The instances are generated from our PROVER method or the baseline ENTAILER method. Scores are average of 3 runs.

Results for MNLI and MED: Table 4 shows results across various combinations of amounts of MNLI data and PROVER or ENTAILER generated data. MNLI + PROVER generated data always helps in improving performance over the setting of using only MNLI data. The incorporation of ENTAILER generated data also helps in improving performance for the MNLI validation set in some of the settings. However, the addition of ENTAILER generated data doesn't help at all for generalization in the MED dataset, where we observe drop in performance compared to the MNLI only training settings. PROVER generated data is thus more effective than ENTAILER generated data for MNLI and MED all the settings. Notably, the F1 scores in MED is more than 3% better with MNLI + PROVER compared to MNLI + ENTAILER.

Effect of Augmentations on Reasoning Types: PROVER generates single-step entailments (I_{t+1}) from a sentence (I_t). The pairs I_t, I_{t+1} encode fine-grained commonsense knowledge. Therefore, we observe the effect of augmentations on different types of reasoning as required in the NLI task (and by extension for NLU). The LoNLI dataset consists of CHECKLIST templates of (premise, hypothesis) pairs, associated examples, and corresponding entailment, neutral, or contradiction labels. The templates are categorized according to 17 reasoning

MNLI, Proofs % Proofs From Category ↓	1, 0% -	1, 1% ENTAILER	1, 1% PROVER	5, 0% -	5, 5% ENTAILER	5, 5 % PROVER	10, 0% -	10, 10% ENTAILER	10, 10% PROVER
Boolean	63.82	53.28	50.72	52.81	67.88	65.64	63.98	72.12	72.77
Conditional	36.23	52.15	34.40	18.16	52.71	34.12	47.01	37.56	49.18
Coreference	50.02	50.00	52.48	51.10	53.70	51.43	72.90	57.04	77.60
Negation	99.87	96.76	99.60	99.24	96.58	99.69	89.56	81.66	94.68
Numerical	60.98	68.38	66.18	53.52	56.76	69.83	55.19	52.58	63.64
Quantifier	66.33	76.09	69.64	63.00	51.97	65.64	63.39	36.85	63.86
Relational	94.08	68.24	96.51	94.26	82.56	93.40	96.68	60.02	95.61
Spatial	56.93	42.81	59.52	69.91	62.24	65.01	63.93	61.56	61.90
Syntactic	91.52	92.05	94.38	99.25	90.21	96.13	99.97	98.63	100.0
Temporal	35.83	25.28	37.57	37.70	47.19	47.62	56.53	55.16	64.51
World	97.55	99.44	99.60	99.30	99.63	99.34	98.73	99.60	99.50
Implicature*	20.51	28.15	23.97	26.41	36.80	25.28	25.40	31.68	30.04
Presupposition*	86.68	91.99	86.35	93.43	92.95	95.77	99.84	95.13	100.0
Average	66.18	64.97	**66.99**	67.01	68.55	**69.92**	71.78	64.58	**74.87**

Table 5: Reasoning category wise F1 scores on the Lo-NLI dataset for the RoBERTa-Large model. Scores are average of 3 runs. We omit categories, for which relevant knowledge is absent from MNLI, SNLI, and EntailmentBank. Implicature and Presupposition are marked as * as they are rare in the source datasets.

Task	BLEU1	METEOR	ROUGE	CIDEr	Acc
Entailment	28.29	17.70	38.56	61.50	93.39
Contradiction	21.46	10.55	25.76	26.24	83.35
Monotonicity	25.74	18.35	38.58	88.07	91.96
Composition	53.54	39.37	81.10	128.42	-

Table 6: Results for *Prover* on various tasks it was supervised on as described in §3.1 and §3.2.

categories, such as quantifier, temporal, and syntactic. We report results for evaluation on LoNLI with models trained on subsets of MNLI and optionally the PROVER or ENTAILER generated data in Table 5. We observe a similar trend, where incorporation of PROVER generated data helps in improving the overall performance with significant gains across a number of reasoning categories. The improvement in numerical, relational, and temporal categories are most prominent. We also find an almost 2% overall improvement for 5% MNLI + 5% Proofs over 10% MNLI + 0% Proofs.

Performance of Prover on Multi-Tasks: We report the performance of PROVER on the constituting tasks it was initially trained on (Sections 3.1 and 3.2) in Table 6. We generate potential entailment and contradictory sentences from the premises in SNLI and monotonic sentences from the premises in MED. We measure the quality of the generations using the generative metrics BLUE unigram, METEOR, ROUGE-L, and CIDEr. We also measure the NLI label of the (*premise*, *generated sentence*) pair using a pre-trained NLI prediction model. The percentage of generated instances that corresponds to the correct label is shown in the **Acc** or Accuracy column in Table 6. In particular, 93.39% of the generated entailment instances are classified as entailment, and 83.35% of the generated contradictory instances are classified as a contradiction. The accuracy scores combined with the generative metrics let us conclude that the PROVER model is indeed able to generate high-quality inferences as required.

6 Related Work

We explore generation of multiple-step natural language proofs for the NLI task for real-world cases, without full context and without complete supervision. We discuss types of proof datasets, proof generation algorithms and how proofs can be further utilized.

Categories of Proofs. The NLP community has transitioned from summarized explanations (Wiegreffe and Marasović, 2021) to multiple-step natural language-based or structured proof trees (NLProlog). Human-provided explanations (Camburu et al., 2018) are hard to validate; and structured explanations are hard to scale. For certain closed-world setting (Tafjord et al., 2021; Saparov and He, 2022) and symbolic domains, using synthetically generated explanations as supervision has been a popular choice. In these datasets (and EntailmentBank, ProntoQA), the common assumption is that, all sentences (or facts) required to prove (or disprove) a statement is provided in context. Similarly, Nye et al. (2021) explored NATURALPROOFS, where each proof step consists of both natural language and mathematical symbols. The underlying reasoning task being mathematical, makes the steps more well-defined. Here, we consider an open-ended context and define constraints over natural language proofs to ease validation and generation.

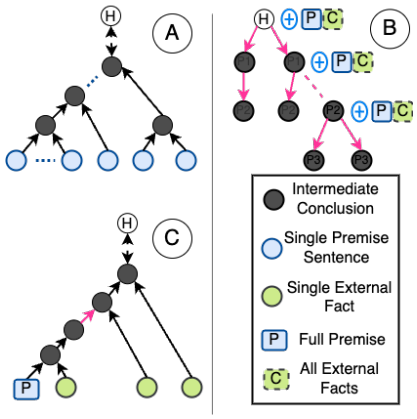

Figure 2: Types of proof generation algorithms based on how they utilize external knowledge and premise sentences. Class (A) use a select-compose-iterate strategy and do not use open-ended set of facts (Hong et al., 2022; Yang et al., 2022). Class (B) uses backward-chaining, while taking the entire premise and (optional) external facts into context in each step (Tafjord et al., 2022). Ours is Class (C), where we use forward chaining, and add single retrieved fact in each step (or uses monotonic entailment generator to generate entailments).

Iterative Proof Generation and Search. Tafjord et al. (2021) utilize synthetically generated end-to-end proof trees for supervision. Authors propose an iterative chaining method, utilizing the T5 model to generate the next step of a proof, and chain them to generate full proofs. Following the work, several recent forward and backward chaining algorithms are proposed which generates a proof tree given a premise and a hypothesis. We show two classes of existing algorithms (compared to ours) in Fig. 2. Similar to Tafjord et al. (2021), class (A) algorithms (METGEN, NLProofS, IRGR) (Hong et al., 2022; Yang et al., 2022; Neves Ribeiro et al., 2022) are focused on evaluating deductive reasoning capabilities where a set of explicit rules and facts are provided to the model; and model uses *selection-composition-search* to generate the proof tree. Sprague et al. (2022) proposes ADGV to extend class (A) algorithms with abductive step, where the missing knowledge can be *generated* using a context and hypothesis. However, factuality of these sentences are hard to verify. ENTAILER (Tafjord et al., 2022) is one algorithm (class (B)) where the full context sentences is optional. This is why we compare with ENTAILER. The model may use its inherent knowledge, premise sentences (or optional context sentences). ENTAILER uses backward chaining algorithm to derive the premise sentences that successfully entail the hypothesis. Our method falls in class (C), where (similar to (B)) we use external sentences and are not limited to full-context specifications as input. We generate the tree by adding individual external facts in each step (or generating single-step entailments), adding a flavor of sentence selection by Class (A).

Utilization of Proofs to Improve End-task Accuracy. He et al. (2021a) and Kumar and Talukdar (2020) has demonstrated the utility of generating proofs (or natural language explanations). In Knowledge-graph based QA context, He et al. (2021a) show how a teacher network trained on additional intermediate hops can be used to enhance the performance of a student network, that is exposed only to the final output supervision. Most importantly, Kumar and Talukdar (2020) uses label-specific explanations and use them directly to generate the NLI conclusion. While the authors show how explanations are used to generate the conclusion, the authors do not generate fine-grained steps. Authors also compare with a non-recent baseline by Camburu et al. (2018) on MNLI, instead of SOTA methods such as DeBERTA-large.

We study multi-step reasoning for the NLI task; where connecting a premise and hypothesis may require external knowledge. Our underlying T5 model is motivated by ProofWriter. However, we do not provide the required rules explicitly, do not restrict the natural language input in any form and do not use end-to-end supervision. The fact retrieval method is inspired from Guu et al. (2020); Gontier et al. (2020). Gontier et al. (2020) proposes to combine a non-parametric retriever model with a parametric generator for knowledge-augmented NLP tasks. We use a retrieval method with frozen parameters as we do not have knowledge-augmented sentence composition supervision for learning retrieval and generation in an end-to-end fashion.

7 Conclusion

We propose a method to generate knowledge-enriched multiple-step textual proofs (intermediate conclusions) for the NLI task utilizing only next-step supervision. We train a T5 model to generate the next step given a premise-hypothesis pair, and use external commonsense knowledge augmentation to search for more generalized proofs. To ease generation and verification, we introduce constraints over expected proofs, and associated metrics. Human verification shows the effectiveness of our proposed method. We also show that

our generated proofs can be used to improve NLI task performance using standard data augmentation techniques (on low-data scenarios), benefiting targeted reasoning types.

8 Limitations

Firstly, we evaluated our model on the NLI task of English language. Multilingual NLI models have been proposed in the last few years, on which we want to evaluate our framework as future work. However, the absence of large commonsense corpus for retrieval could present some difficulty for non-English languages. Secondly, we have evaluated our model on SNLI which contains simple premise hypothesis pairs. In the future our work could be extended to tackle more complex sentences or paragraphs and mathematical reasoning datasets for proof generation.

References

Sumithra Bhakthavatsalam, Chloe Anastasiades, and Peter Clark. 2020. Genericskb: A knowledge base of generic statements. *arXiv preprint arXiv:2005.00660*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. *arXiv preprint arXiv:2104.08661*.

Bhavana Dalvi, Oyvind Tafjord, and Peter Clark. 2022. Towards teachable reasoning systems. *CoRR*, abs/2204.13074.

Nicolas Gontier, Koustuv Sinha, Siva Reddy, and Chris Pal. 2020. Measuring systematic generalization in neural proof generation with transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 22231–22242. Curran Associates, Inc.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021a. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 553–561. ACM.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021b. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Ruixin Hong, Hongming Zhang, Xintong Yu, and Changshui Zhang. 2022. METGEN: A module-based entailment tree generation framework for answer explanation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1887–1905, Seattle, United States. Association for Computational Linguistics.

Divyansh Kaushik, Eduard Hovy, and Zachary C. Lipton. 2019. Learning the difference that makes a difference with counterfactually-augmented data.

Sawan Kumar and Partha P. Talukdar. 2020. NILE : Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8730–8742. Association for Computational Linguistics.

Zachary C. Lipton. 2018. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Rui Dong, Xiaokai Wei, Henghui Zhu, Xinchi Chen, Peng Xu, Zhiheng Huang, Andrew Arnold, and Dan Roth. 2022. Entailment tree explanations via iterative retrieval-generation reasoner. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 465–475, Seattle, United States. Association for Computational Linguistics.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena.

2021. Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Nils Reimers and Iryna Gurevych. 2019. 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.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. Prover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 122–136. Association for Computational Linguistics.

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

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *CoRR*, abs/2210.01240.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237. Springer.

Zayne Sprague, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2022. Natural language deduction with incomplete information. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8230–8258, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3621–3634. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2022. Entailer: Answering questions with faithful and truthful chains of reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2078–2093, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ishan Tarunesh, Somak Aditya, and Monojit Choudhury. 2021. Trusting roberta over bert: Insights from checklisting the natural language inference task. *arXiv preprint arXiv:2107.07229*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2020. Is multihop QA in DiRe condition? measuring and reducing disconnected reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8846–8863, Online. Association for Computational Linguistics.

Sarah Wiegreffe and Ana Marasović. 2021. Teach me to explain: A review of datasets for explainable nlp. In *Proceedings of NeurIPS*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019. Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 31–40, Florence, Italy. Association for Computational Linguistics.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 89–105, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Prover

The multitask T5 generator model(§3.1) can be trained on additional objective functions for improved performance and better generalization. The objectives are as follows:

Explanation Generation: The model is trained to generate an explanation X given the premise P and hypothesis H. The input is `explain: P <sep> H` and the output to be generated is `X`. The non-neutral instances of the E-SNLI (Camburu et al., 2018) dataset is used for this objective.

Entailment Bank Proof Generation: The model is also trained to generate one-step proofs given a premise and a conclusion. Suppose conclusion c can be derived from premise sentences s_1, s_2. For this instance, the input to the model is `proof:` s_1 `<sep> c` and the output to be generated is s_2. Another instance is also created by interchanging s_1 and s_2 in the input and output. We consider all leaf sentences, intermediate conclusions, and the final hypothesis of the proof trees in the Entailment Bank dataset (Dalvi et al., 2021) to create these instances.

These two objectives are more useful for explanation / intermediate step generation for neutral premise-hypothesis pairs.

B Constrained Proof Generation

The *explain:* and *proof:* generation setup can be used to directly generate the proof in using the premise P and hypothesis H. We observe that the generator model learns to exploit the proof generation process by generating a single sentence as proof, due to the nature of the training data in E-SNLI and Entailment Bank. We thus evaluate the generated proofs against the gold test set annotations in E-SNLI with a number of text generation evaluation metrics – BLEU, METEOR, ROUGE, CIDER and semantic similarity (SIM) using *all-mpnet-base-v2*. The results are reported in Table 7.

Prefix	BLEU1	METEOR	ROUGE	CIDEr	SIM
explain	40.63	24.91	38.98	66.03	63.72
proof	26.31	15.37	26.98	77.05	47.20

Table 7: Results for constrained proof generation. The generated proofs are matched against the gold annotations in E-SNLI. **SIM** indicates semantic similarity. All scores except CIDEr are shown on a scale of 0-100.

C Datasets used

All datasets used and results reported in this paper are for English language.

LoNLI: The dataset contains templates of *(premise, hypothesis)* pairs, associated examples, and corresponding entailment, neutral, or contradiction labels. For instance, one such entailment template is - *premise: Name moved from Country1 to Country2; hypothesis: Name now lives in Country2*. All examples created from this template are labeled as entailment. There are 363 templates in total (166 entailment, 163 contradiction, 34 neutral) each having 1000 examples. The templates are categorized according to reasoning categories, such as lexical, syntactic, boolean, causal, etc. The dataset was inspired by the behavioral testing methodology of NLP systems proposed in CHECKLIST (Ribeiro et al., 2020).

D Additional Results

We show some additional results to show the usefulness of the PROVER for various NLI tasks. This is an extension of the results reported in §5.3. As described earlier, we use the PROVER generated data as additional labeled data for supervised learning in NLI tasks. We train a DeBERTa-Large model (He et al., 2021b) with i) MNLI-only data or ii) a mix of MNLI data and PROVER or ENTAILER generated data. The results are reported for MNLI and MED datasets in table 8, and for Lo-NLI dataset in table 9.

For the DeBERTa-Large model, we found similar conclusions that we made for the RoBERTa-Large model. The incorporation of PROVER generated data helps in improving performance across most settings in the three datasets. In the low data

Trained On				
MNLI %	Proofs %	Proofs From	MNLI	MED
1	0	-	88.67	42.31
1	1	ENTAILER	87.56	41.74
1	1	PROVER	88.99	43.26
5	0	-	89.06	42.63
5	5	ENTAILER	89.44	41.32
5	5	PROVER	89.80	44.54
10	0	-	89.81	44.30
10	10	ENTAILER	89.85	42.58
10	10	PROVER	**90.28**	**45.27**

Table 8: F1 scores for MNLI (val-matched) and MED datasets for the DeBERTa-Large model. Scores are average of 3 runs.

MNLI, Proofs % Proofs From Category ↓	1, 0% -	1, 1% ENTAILER	1, 1% PROVER	10, 0% -	10, 10% ENTAILER	10, 10 % PROVER	50, 0% -	50, 50% ENTAILER	50, 50% PROVER
Boolean	65.99	64.30	61.81	64.24	62.41	66.71	81.07	71.01	76.78
Conditional	37.28	41.25	53.69	50.67	66.02	64.19	49.89	66.86	77.30
Coreference	64.23	50.25	56.11	78.23	65.22	78.45	78.66	77.68	83.52
Negation	94.69	96.74	96.80	97.33	95.56	96.24	95.73	95.15	88.73
Numerical	61.57	61.70	62.90	72.48	76.18	82.64	83.68	71.59	86.33
Quantifier	65.54	61.36	69.32	71.93	73.43	73.58	68.49	60.97	71.75
Relational	98.27	91.80	98.95	98.52	62.54	98.74	97.93	96.09	97.09
Spatial	66.47	49.60	66.28	59.89	45.46	53.69	49.01	78.77	61.93
Syntactic	99.88	99.86	99.13	99.97	96.00	99.99	99.99	99.98	100.0
Temporal	51.41	43.20	46.99	61.00	46.89	64.60	79.75	76.96	75.35
World	99.88	99.58	99.60	99.47	99.82	99.24	99.22	99.98	99.40
Implicature*	31.76	23.39	29.15	30.33	52.97	30.42	30.24	36.69	33.61
Presupposition*	85.70	87.27	90.41	99.04	99.81	97.08	98.82	99.80	99.90
Average	70.97	66.95	**71.63**	75.62	72.49	**77.35**	77.88	79.35	**80.90**

Table 9: Reasoning category wise F1 scores on the Lo-NLI dataset for the DeBERTa-Large model. Scores are average of 3 runs. We omit categories, for which relevant knowledge is absent from MNLI, SNLI, and EntailmentBank. Implicature and Presupposition are marked as * as they are rare in the source datasets.

regime, the additional instances from PROVER helps in improving average performance in the MED dataset. The additional instances from ENTAILER do help in improving the performance in some cases, but not always. In contrast, the performance in MED drops when ENTAILER generated data is used.

The incorporation of PROVER generated data also helps in improving the overall performance with significant gains across a number of reasoning categories in the Lo-NLI dataset.

E Instructions for Human Verification

We performed human verification for proofs with two intermediate steps. The following instructions were given as it is to the human annotators.

E.1 Definition of Entailment

Entailment is a directional relation between two sentences - S1 and S2. The relation holds whenever the truth of the second sentence S2 follows from the first sentence S1. In other words, if a human reading S1 infers that S2 is true, then (S1, S2) is an entailment pair. Note that, (S1, S2) being an entailment pair does not necessarily mean that the reverse pair (S2, S1) is an entailment pair. Some examples are given below:

- S1: A football game with multiple males playing.

  S2: Some men are playing a sport.

  Label: (S1, S2) is an entailment pair.

  However, (S2, S1) is not an entailment pair because playing a sport does not necessarily mean playing football. More examples:

- S1: A woman is walking outside.

  S2: A person outdoors.

  Label: (S1, S2) → Entailment.

- S1: An older and younger man smiling.

  S2: Two men are smiling and laughing at the cats playing on the floor.

  Label: (S1, S2) → Not Entailment.

  If S1 and S2 are the same sentences or very similar sentences with minimal difference in tokens then consider that as entailment.

- S1: An older and younger man smiling.

  S2: An older and younger man smiles.

  Label: (S1, S2) → Entailment.

E.2 Definition of Uniqueness

S1 and S2 are two given sentences. If S1 and S2 are very similar in the token space then consider (S1, S2) as not unique. If they are not very similar then they are unique. This is a symmetrical relation. The uniqueness of (S1, S2) is the same as (S2, S1).

- S1: A football game with multiple males playing.

  S2: Some men are playing a sport.

  Label: Unique.

- S1: An older and younger man smiling.

  S2: An older and younger man smiles.

  Label: Not unique.

- S1: A man is playing a guitar.

  S2: A man is playing a musical instrument.

  Label: Unique.

- S1: A man is playing a guitar.

  S2: A guitar is being played by a man.

  Label: Not Unique.

Please note the difference between the last two examples carefully. The usage of the word musical instrument makes the third pair unique.

E.3 Instructions

Consider a proof from the *proof search with facts* method. Each instance has four elements: i) Premise, ii) Intermediate Steps, iii) Hypothesis, and iv) Facts used. We are aiming to evaluate the quality of the proof.

We illustrate a proof example which uses two facts – Fact 1, Fact 2, and has two intermediate steps – Step 1, Step 2. Considering this, we have 3 sentence pairs from the proof:

- X1 : (Premise, Step 1)
- X2 : (Step 1, Step 2)
- X3 : (Step 2, Hypothesis)

Each instance has to be scored in five aspects using the three above pairs and the two facts:

1. Correctness: Give a score among [3, 2, 1, 0]. This is assigned by checking the entailment label of the three pairs.
   - X1, X2, X3 are all entailment: Score is 3.
   - Only X1, X2 are entailment: Score is 2.
   - Only X1 is entailment: Score is 1.
   - All other cases: Score is 0.

   Note that the priority of the pairs are: X1 $>$ X2 $>$ X3. If X2, X3 are both entailment but X1 is not then we will give a score of 0.

2. Minimality: Give a score among [3, 2, 1, 0]:
   - If X1, X2, X3 are all unique pairs: Score is 3.
   - Score of 2, or 1, or 0 analogous to correctness, but conditioned on uniqueness.

3. Useful Facts: Give a score between [2, 1, 0] denoting how many facts are useful for the proof.
   - Both facts are useful: Score is 2
   - Only one fact is useful: Score is 1.
   - Neither facts are useful: Score is 0.

4. Hypothesis follows: Does the hypothesis follow from the premise and the listed facts? This score is inspired from Dalvi et al. (2022).
   - Clearly follows: Score is 2
   - Somewhat follows: Score is 1.
   - Does not follow: Score is 0.

We do not have any facts in the proof for the *unconstrained proof search* method. So, we measure only correctness and minimality for such proofs.

The scores for Correctness, Minimality, and Useful Facts are scaled appropriately for proofs with more intermediate steps and facts. During evaluation, we normalize all the scores between 0-1 (0-100 in %) by appropriately considering the number of intermediate steps and facts.

Details of the Annotators : The human verification was performed by five graduate students who are trained in natural language processing. All the students are fluent in English. They were paid hourly rates as deemed by our university.

F Experimental Setup

We use beam search to generate outputs from the T5-Large models. A beam length of 10 is used. The T5-Large models were trained with the Adafactor optimizer (Shazeer and Stern, 2018) with a learning rate of 5e-6. We retrieve top 8 facts (according to cosine similarity) from the retriever model for composition. We compute the generative evaluation metrics using this package: https://github.com/Maluuba/nlg-eval

G Computational Resources

We use a single Quadro RTX 8000 GPU for our experiments. We train the PROVER T5 model and composition T5 model for 15 and 5 hours in this GPU. The T5-Large and RoBERTa-Large models have 770M and 355M parameters, respectively.