Entailer: Answering Questions with Faithful and Truthful Chains of Reasoning

Oyvind Tafjord, Bhavana Dalvi Mishra, Peter Clark
Allen Institute for AI, Seattle, WA
{oyvindt,bhavanad,peterc}@allenai.org

Abstract

Our goal is a question-answering (QA) system that can show how its answers are implied by *its own internal beliefs* via a *systematic chain of reasoning*. Such a capability would allow better understanding of *why* a model produced the answer it did. Our approach is to recursively combine a trained backward-chaining model, capable of generating a set of premises entailing an answer hypothesis, with a verifier that checks that the model itself believes those premises (and the entailment itself) through self-querying. To our knowledge, this is the first system to generate multistep chains that are both *faithful* (the answer follows from the reasoning) and *truthful* (the chain reflects the system's own internal beliefs). In evaluation using two different datasets, users judge that a majority (70%+) of generated chains clearly show how an answer follows from a set of facts - substantially better than a high-performance baseline - while preserving answer accuracy. By materializing model beliefs that systematically support an answer, new opportunities arise for understanding the model's system of belief, and diagnosing and correcting its misunderstandings when an answer is wrong.

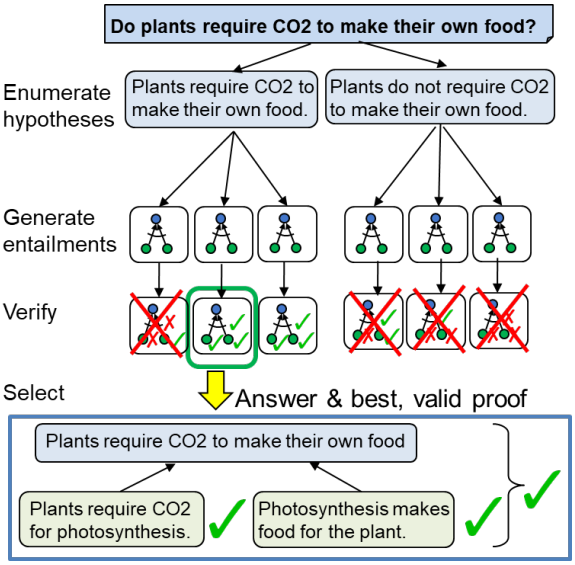

Figure 1: Given a question, Entailer searches for an answer hypothesis that is supported by an entailment proof. First it over-generates candidate proofs, then it removes those that the model itself does not "believe" (i.e., confirms via self-querying that it considers all the generated proof elements to be true). Finally it selects the best verified proof. Multistep proofs are generated by iteratively backward chaining on the premises (Section 3.2).

1 Introduction

Although pretrained language models (PTLMs) have shown remarkable question-answering (QA) performance, it is often unclear *why* their answers follow from what they know. While there has been substantial work on training models to also generate explanations for their answers (Wiegreffe and Marasović, 2021), or produce them via few-shot prompting, e.g., "chains of thought" (Wei et al., 2022), those explanations may not be *faithful* (the answer does not necessarily follow from them) and may not be *truthful*, in the sense that the language model itself does not believe[1] the explanation statements that it generated. Rather, our goal is to generate answers that *systematically follow* from the model's own internal beliefs, materializing those beliefs as explicit statements that can then be inspected. Such a capability offers new opportunities for understanding, diagnosing, and ultimately correcting errors in a language model's behavior.

Our approach uses a combination of generation and verification, implemented in a system called Entailer[2]. Chains are constructed by backward chaining from candidate answers, recursively using a language model (LM) trained for a single backward-chaining step. For each step, Entailer over-generates candidate entailments, then filters

[1]We here adopt a simple operational definition of belief, namely that a model believes X if it answers "yes" to the question "Is X true?". Other definitions could also be used.

[2]Entailer data and models are available at https://allenai.org/data/entailer

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 2078 - 2093
December 7-11, 2022

Q: A magnet will stick to (A) a belt buckle (B) a wooden table (C) a plastic cup (D) a paper plate
A: (A) A magnet will stick to a belt buckle **because**
    A belt buckle is sometimes magnetic. **because**
        Metal is sometimes magnetic.
        A belt buckle is made of metal.
    A magnet will stick to magnetic metals.

Q: You can make a telescope with a (A) straw (B) Glass (C) Candle (D) mailing tube
A: (D) You can make a telescope with a mailing tube. **because**
    A telescope is made of a tube for observing / seeing.
    A mailing tube is a kind of tube.

Q: Quartz may produce rainbows when light is shined (A) around the crystal's area (B) through any of its sides (C) in the room its in (D) in to a mirror at it
A: (B) Quartz may produce rainbows when light is shined through any of its sides. **because**
    A rainbow is produced when light shines through a prism. **because**
        The rainbow is made of all different colors in visible light.
        A prism can split light into different colors.
    A quartz is a kind of prism.

Figure 2: Questions (from the OBQA dataset) and Entailer's answers, showing its chain of reasoning.

out those that do not conform to its own internal knowledge ("beliefs") by self-querying, asking itself whether (a) the generated premises (leaves of the proof step) are true, and (b) each entailment step is valid (Figure 1). It then recursively backward-chains on premises until the overall proof confidence cannot be further improved (or a depth limit d is reached). Finally, the candidate answer supported by the highest-scoring chain of a reasoning is returned. As a result, the system has materialized some of its latent knowledge from which the selected answer follows. Most significantly, the resulting proof is thus both *faithful* (the answer follows from the proof) and *truthful* (the proof reflects the system's beliefs), providing a previously unavailable window into the model's beliefs about the world and their implications, e.g., Figure 2.

To train the Entailer model, we use a combination of the existing EntailmentBank dataset (Dalvi et al., 2021), plus a new crowd-annotated dataset that we construct by bootstrapping (train an initial model, generate candidate entailment examples with it, then annotate those examples as extra training data). The model is then frozen, and Entailer is then applied **zero-shot** to new datasets, i.e., Entailer is a treated as a general-purpose, fixed model specialized for reasoning, rather than requiring fine-tuning for new tasks.

We evaluate Entailer on two existing datasets, OBQA (Mihaylov et al., 2018) and QuaRTz (Tafjord et al., 2019). We find that its reasoning-based QA accuracy is similar to its direct QA accuracy, with the advantage that a supporting chain of reasoning is also produced. We also perform a human evaluation, and find that 70% of time users judge the chains to clearly show how an answer followed from their premises, substantially higher than for explanations produced by a comparable high-performance QA system, Macaw (Tafjord and Clark, 2021). Our contributions are thus:

1. The first system to generate chains of reasoning showing how answers are systematically implied by a *model's own internal beliefs*, making relevant model beliefs explicit. The chains are both ***faithful*** (the answer follows from the reasoning) and ***truthful*** (the chain reflects the system's own beliefs).
2. A new, crowdsourced dataset of multi-premise entailments, doubling the amount of data available in EntailmentBank (Dalvi et al., 2021), and including examples of both positive and negative entailments (EntailmentBank only includes positive examples)[3].

2 Related Work

Systematic Reasoning: Several recent systems have demonstrated the ability to perform *systematic* reasoning directly over natural language (Natural Language Inference (Manning and MacCartney, 2009)), namely deriving conclusions from known facts via step-wise application of well-defined inference operations. One approach is to retrain a black-box model end-to-end (Clark et al., 2020), but has been limited to small rulebases. An alternative approach, which we follow here, is to have an outside loop around a model, where the model generates individual inference steps (i.e., rules), and a con-

[3]The dataset is provided in the supplementary material.

troller chains them together. SCSearch (Bostrom et al., 2022), NLProofS (Yang et al., 2022), IRGR (Ribeiro et al., 2022), ProofWriter (iterative version) (Tafjord et al., 2020), and Selection-Inference (Creswell et al., 2022) do this in a forward-chaining fashion, MetGen (Hong et al., 2022) does this bidirectionally, while Braid (Kalyanpur et al., 2020) (like us) does this in a backward-chaining fashion. In all these systems, the required facts were expected to be provided explicitly to the model. In contrast, Entailer's reasoning uses its own internal, latent knowledge, as well as (optionally) externally provided facts. LeapOfThought (Talmor et al., 2020) demonstrated that reasoning with a combination of implicit and explicit knowledge was possible for simple 1-step inferences. We expand this for multi-step inference, and (unlike LeapOfThought) have the system also explicitly articulate the implicit knowledge it uses, and its chain of reasoning.

Recent work has shown that generating a free-form explanation ("chain of thought") before an answer can also improve performance on a variety of tasks (Wei et al., 2022; Cobbe et al., 2021; Nye et al., 2021). In these works, however, the explanations are unstructured, and there are no claims of faithfulness that the answer follows from the generation, nor that the explanations themselves represent model beliefs.

Materializing a Model's Internal Knowledge: Pretrained LMs contain a vast amount of knowledge, and can be thought of as a kind of knowledge base to tap into (Petroni et al., 2019). Recent work has shown that this latent knowledge can be materialized as explicit English sentences or a knowledge graph using generative techniques, e.g., COMeT (Bosselut et al., 2019), ParaCOMET (Gabriel et al., 2021). Our work with Entailer similarly materializes its latent knowledge, but here in a *goal-directed* way, namely by producing a faithful chain of reasoning from facts it validates ("believes") as true to an answer. This articulation can be seen as a kind of self-talk, where a self-generated context can improve QA (Shwartz et al., 2020). However, here our generations are not used as context for opaque problem-solving, but are assembled into a well-defined chain of reasoning.

Beliefs: We refer to the model's factual opinions as "beliefs" rather than "knowledge" because those opinions may be wrong. In general, an agent can be said to believe p if it acts as if p was true (Schwitzgebel, 2019). Following (Kassner et al., 2021), we take a simple, syntactic operationalization of this, namely the agent answers "yes" to the question "p?", but also note that more semantic versions could be used, e.g., the agent also answers "yes" to paraphrases and implications of p. In general, models can sometimes be inconsistent in their beliefs (Elazar et al., 2021; Kassner and Schütze, 2020; Ribeiro et al., 2019). For our purposes here, we simply note that such inconsistencies may occasionally exist, and that techniques for inconsistency resolution could be applied in future to reduce these, e.g., (Kassner et al., 2021; Li et al., 2019).

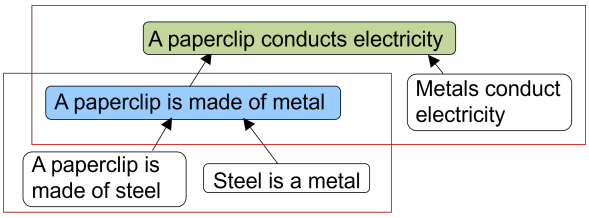

Figure 3: An *entailment tree* is a set of multi-premise, 1-step entailments (red boxes) showing how the hypothesis (root node, green) is entailed from the leaf nodes (white). If all the leaf nodes are true, and all the 1-step entailment relations are valid, then we say the tree is a *valid chain of reasoning* for the hypothesis.

3 Approach

Like several previous systems (Section 2), Entailer treats reasoning as Natural Language Inference (NLI). In NLI, the basic unit of knowledge is (represented as) a sentence rather than a structure, and a proof[4] is a tree of multi-step, multi-premise entailments, e.g., Figures 2 and 3.

Within this framework, given a question, Entailer first generates candidate answers, then tries to prove each one, selecting the answer with the highest-scoring proof. We now describe these steps.

3.1 Hypothesis Generation

Given a question, Entailer first generates candidate answers and converts these into declarative hypotheses (e.g., "Is the sky (A) blue (B) yellow" $\rightarrow$ { H_1 = "The sky is blue.", H_2 = "The sky is yellow."}).[5] An N-way multiple choice question yields N hypotheses. A true/false question yields

[4]We use the word "proof" for convenience but note that the term is somewhat approximate, as entailment "proofs" do not have the guarantees of formal, deductive proofs.

[5]Conversion of a QA pair to a declarative hypothesis D uses a custom T5-11B model trained on the QA2D dataset (Demszky et al., 2018).

Angle	**Input** $\rightarrow$ **Output** (example)
$H \rightarrow P$	"H: A paperclip is made of metal. P:" $\rightarrow$ "[PREMISE] A paperclip is made of steel. [PREMISE] Steel is a metal."
$H \rightarrow S_d$	"H: A paperclip is made of steel. V:" $\rightarrow$ 0.995
$PH \rightarrow S_e$	"H: A paperclip is made of metal. P: [PREMISE] A paperclip is made of steel. [PREMISE] Steel is a metal. I:" $\rightarrow$ 0.998

Table 1: Examples of the three input/output angles used by Entailer. The first generates a candidate entailment rule P⊢H given H. The second and third score whether each premise, and the entailment itself, is valid, using tokens V/I in the input to indicate that S_d/S_e is the desired output.

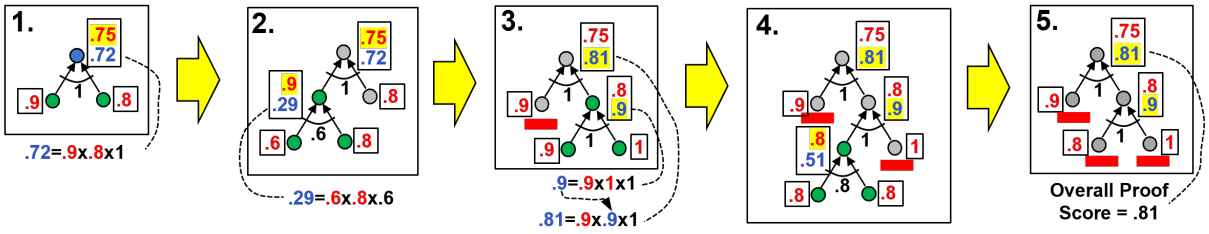

Figure 4: The entailment tree is grown recursively, the algorithm looking for the best tree (maximizes the overall score of the root node). Each node has a fixed, direct ("fast") score (in red), and (for internal nodes) a proof ("slow") score (in blue) computed from its children. The overall node score (highlighted) is the highest of the two. If expanding a node increases its overall score (e.g., step 3), that increase is propagated upwards and recursion continues. If expansions cannot improve a node's score further (e.g., steps 2 and 4), the expansions are pruned and that node becomes a leaf (red bars).

2 hypotheses. For open-ended questions, Entailer first collects N candidate answers generated by an external source (Macaw (Tafjord and Clark, 2021) using nucleus sampling (Holtzman et al., 2019)) then forms N hypotheses from them.

3.2 Generating Entailment Trees

3.2.1 Generating a Backward-Chaining Step

Models:

The core of Entailer is generating and validating a single entailment step that entails a hypothesis. We define the following data types:

- **H:** A hypothesis (English statement) to prove.
- **P:** A set of premises {p_1,...,p_i} (sentences) that together may *entail* the hypothesis H. Together, P and H form a one-deep *entailment step*, denoted by P ⊢ H.
- **Q:** A question posed to Entailer.
- **A:** A candidate answer for consideration.
- **C:** An optional context (set of sentences) relevant to the problem. This allows Entailer to also use external knowledge, if available, when generating a tree.

We train a model (details in Section 4) with the three input/output behaviors below (optional inputs shown in parentheses):

$(QAC)H \rightarrow P$: Given a hypothesis H, generate a set of premises P that may entail H

$(QAC)H \rightarrow S_d$: Score whether the model believes that a hypothesis H (or premise p_i) is true ($S_d > 0.5$) or not, (i.e. perform yes/no QA). We call this the **direct** score (range 0-1).

$(QAC)PH \rightarrow S_e$: Score whether the model believes a candidate entailment (P ⊢ H) is valid ($S_e > 0.5$) or not, i.e., P validly entails H (range 0-1).

Examples of these three angles are in Table 1.

Algorithm:

To generate a single backward-chaining step we adopt an overgenerate-and-filter approach, also found useful elsewhere (Yang et al., 2022; Cobbe et al., 2021; Li et al., 2022). First, given H, we use the angle $H \rightarrow P$ to generate a set of premises P that may entail H. We then check that the model believes all the premises $p_i \in P$ using the $H(=p_i) \rightarrow S_d$ angle, and that it also believes the inference step $P \vdash H$ itself is valid (independent of whether the p_i are true or not) using the $PH \rightarrow S_e$ angle. The proof score, denoting how well the 1-step proof supports the hypothesis, is the product of the premises' and entailment scores:

$$s_{\text{r-1deep}}(H) = (\Pi_i s_d(p_i)).s_e(P \vdash H)$$

We repeat this k times using nucleus sampling to obtain a diversity of alternative proof steps, and then select the highest-scoring one $P \vdash H$, as illustrated in Figure 1.

3.2.2 Backward Chaining

This one-step backward chainer is embedded in a larger control algorithm that grows a multi-step entailment tree, searching for the best tree. This algorithm is illustrated in Figure 4 and described below. The full algorithm is in Appendix A.

Each node N in the tree has two associated scores, the direct ("fast") score s_d, denoting the model's direct belief in N (in red in Figure 4), and (for internal nodes) the proof ("slow") score s_r, denoting how confidently the model can *prove* N (in blue), computed from its children. The overall score s is the max of the two. The proof score s_r is recursively defined as the product of its children's overall scores times the entailment score:

$$s_r(N) = (\Pi_i s(p_i)).s_e(P \vdash N)$$

The algorithm starts with an initial hypothesis node H, then iteratively expands each leaf node N, looking for a proof that scores higher than the direct score s_d of that node. In other words, can the system prove N with a more confidence than simply "guessing" if N is true? If it can, the node's overall score s (max of the two) will increase, that increase is be propagated up the tree, and the expansion is retained, e.g., step 3 in Figure 4. If expansions cannot improve a node's score further (e.g., steps 2 and 4), the expansions are pruned and that node becomes a leaf (red bars in Figure 4).

Note that because premises are self-generated rather than externally provided, this stopping condition has a different semantics to earlier work, e.g., (Kalyanpur et al., 2020; Bostrom et al., 2021): Rather than stopping when externally known facts are reached, Entailer stops when strongly believed facts are reached, and more backward chaining can no longer improve a hypothesis' confidence.

This whole procedure is repeated for each candidate answer hypothesis (Section 3.1). Finally the system selects the answer corresponding to the hypothesis with the top-scoring proof $s(H)$.

4 Model Training

The core of Entailer is the model for one-step inference (Section 3.2.1), with the three functionalities listed in Table 1. As Entailer is intended to be a general-purpose system, the model is trained one-time for these three functionalities, and then frozen. It is then applied zero-shot to new datasets, e.g., the evaluation in Section 5.

4.1 Data Sources

4.1.1 EntailmentBank

To train Entailer's model, we leverage (the training partition of) the EntailmentBank dataset (Dalvi et al., 2021). EntailmentBank contains 1840 multiple-choice science questions (1313 in the training partition) along with their correct answer, a hypothesis expressing the question and answer in declarative form, and a multistep entailment tree expressing how the hypothesis follows from a set of facts drawn from a corpus of science facts (the WorldTree corpus (Xie et al., 2020)). Using the notation introduced earlier, each EntailmentBank example is of the form:

$$< Q, A, H_0, \{ (P_i \vdash H_i) * \} >$$

where $(P_i \vdash H_i)$* denotes a *set* of entailment steps forming a tree (with root $H_i = H_0$), describing how the corpus facts entail the hypothesis H_0.

4.1.2 Crowdsourced Data

EntailmentBank only contains positive examples of entailments. To obtain negative examples, we first ran an earlier, positive-only-trained version of Entailer to generate 1-step proofs of (hypotheses for) *incorrect* answer options in EntailmentBank's questions (4-way multiple choice), resulting in (necessarily bad) proofs for the three false hypotheses. (This was done using just the H→P angle, without verification). Note that Entailer will generate some kind of proof even for false hypotheses, e.g.,

A rabbit has six legs **because:**
1. A rabbit is an animal
2. Animals have six legs

These invalid proofs will be incorrect either because a generated fact is false, and/or because the inference itself is not a valid entailment. To distinguish these, we use crowdworkers to assess whether the generated facts were true, and if the entailment itself was valid. The 1313 questions result in 3939 proofs for false hypotheses. Dropping the few with more than two premises (to simplify the crowdsourcing interface), crowdworkers annotated 3673 proofs, using labels T/F/? for each premise and T/F/? for the entailment itself. Each proof was annotated by 3 crowdworkers, then an additional 3 workers provided additional annotations for cases with no initial majority verdict. Dropping premises/entailments without a final majority verdict, we end up with 7013 additional labeled premises for the $H \rightarrow S_d$ angle, and 3391 additional labeled entailments for the $PH \rightarrow S_e$ angle.

The crowdworker interface is in Appendix B.

4.1.3 Optional Fields

We also augment the training data with duplicate examples but with additional, optional input fields:

QA: The input QA question/answer-option pair, as well as the hypothesis H

C: A *context* consisting of up to 5 relevant sentences, allowing explicit external knowledge to be provided if available.

To generate examples of C during training, we use a mixture of sentences drawn from (a) the gold (target) entailment (i.e., the gold premises), and (b) sentences retrieved from a corpus of similar-styled knowledge (namely all the leaf sentences used in the EntailmentBank entailment trees), mixed in different ratios so the model is exposed to a mixture of high and medium relevance sentences.

In this work we do not use any context C at test time, but it is utilized in concurrent work for providing feedback to the overall system (Mishra et al., 2022).

Further details of training are given in Appendix C1.

4.2 Model Details

We train a T5-11B multi-angle model, Entailer, following the multi-task setup similar to Macaw (Tafjord and Clark, 2021) for the three functionalities described in Table 1.[6] Details of hyperparameters and other settings are provided in Appendix C2.

5 Evaluation

Our goal is to generate answers supported by faithful, truthful chains of reasoning, without significantly impacting performance. Our two corresponding reserach questions are:

1. How does Entailer's proof-based answer accuracy compare with the direct QA accuracy (both zero-shot)?
2. How good are the generated entailment-based proofs? And are they better than those produced by a purely generative model?

For the first question, we evaluate using two existing multiple-choice datasets, namely OBQA (Mihaylov et al., 2018) and QuaRTz (Tafjord et al., 2019). These datasets contain multiple-choice questions that (typically) require multihop reasoning, rather than being simple lookup questions. We use the test partitions[7] with sizes 500 questions (OBQA) and 557 questions (QuaRTz).

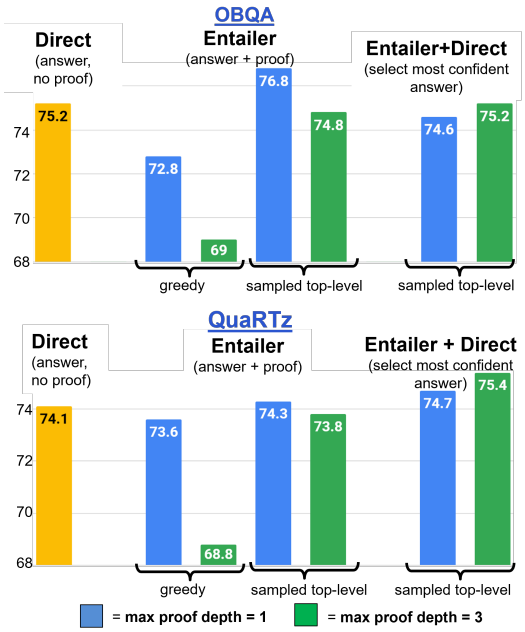

Figure 5: QA accuracy of Direct QA, Entailer, and the two combined on two datasets.

For the second question, we collect human judgements of whether the hypothesis clearly follows from facts in the proof, and compare its proofs with explanations generated by a high-quality baseline model, Macaw (Tafjord and Clark, 2021).

5.1 QA Accuracy

We evaluate Entailer's proof-based QA accuracies under various conditions, and compare them against its direct QA accuracy:

1. Direct QA: Here we measure the model's ability to directly answer the test questions (zero shot) using the $H \rightarrow S_d$ angle. One can think of this as the **"fast thinking"** answer. Note that this capability of the frozen model was trained on the same data as the rest of Entailer, so is a fair comparison.

2. Entailer: Here we measure Entailer's ability to answer the test questions (again zero shot) by generating, scoring, and comparing entailment proofs for each answer option. One can think of this as the **"slow thinking"** answer. We vary:

- **maximum proof depth** $d = 1$ (blue in Figure 5) or 3 (green)
- **degree of search:** (a) **greedy:** use the first ($k = 1$) generated entailment at each step, or

[6] The scores use the token probability of generated "true" or "false" output text.

[7] For QuaRTz, for synchronization with other work (Mishra et al., 2022), we use a portion of the training set as our test partition here. As we are applying Entailer zero-shot, this makes no material difference experimentally.

(b) **sampled top level:** pick the best of $k = 6$ entailments for expanding the root hypothesis node. ($\approx$ six times more computationally expensive than (a)).

Note that these proofs are are *faithful* explanations of why an answer was chosen, as the selected answer by definition is always the one with the highest-scoring proof.

3. Entailer + Direct: Here we combine the two by selecting the overall most confident answer of Direct and Entailer (using $c_d(H)$ or $c_r(H)$, Appendix A). Here, the proofs are not always faithful explanations of an answer, as the chosen answer may not be the one with the highest-scoring proof.

5.2 Results

The results are shown in Figure 5, and suggest that:

Proof Scores are competitive with direct answer scores. In Entailer's best configuration (sampled top-level $k = 6$, max depth $d = 3$, last bar), its reason-based answers have an accuracy of 75.2/75.4 for OBQA/QuaRTz respectively, not significantly different to the Direct scores of 75.2/74.1. This is important, as it suggests there is no significant accuracy penalty for proof-based answers.

Sampling proofs helps: Using sampling for the top-level proofs (last 4 bars) always outperforms greedy proof selection by a small amount.

Allowing deeper proofs does not significantly affect accuracy: Although deeper proofs[8] slightly help in the combination of Entailer+Direct, and may provide more information to a user, the accuracy differences are not significant.

When Entailer + Direct are combined, by selecting the most confident answer (last two columns), we lose the guarantee of faithfulness, as the selected answer may not be the one with the highest-scoring proof. In practice, this occurs for 16.8% of the questions (OBQA), 14.2% (QuaRTz). In addition, we find this combination does not have significant performance gains, so has no obvious advantage in these experiments.

Note that we are measuring *zero-shot* performance on our test datasets, so our results are not comparable with leaderboard entries.[9] More importantly, though, our goal is not a state-of-the-art zero-shot model, but rather a model that can show how answers *systematically follow from its own internal beliefs*. Our results suggest this is possible with high reliability, and, as an additional bonus, without loss of zero-shot accuracy. Thus, rather than just answers, we now get answers supported by faithful chains of reasoning.

5.3 Human Judgements

For our second question of evaluating the quality of Entailer's proofs, we compare against explanations generated by Macaw,[10] a public, state-of-the-art QA system with explanation capability (Tafjord and Clark, 2021). Examples of explanations from both systems are in Appendix D. Six annotators compared 1-deep Entailer proofs with explanations from Macaw, scoring each along four dimensions, and then comparing the two:

1. Does the conclusion clearly follow from the premises?
2. Are all the premises correct?
3. Are all of the premises relevant?
4. Does the explanation introduce something new and useful, i.e., does more than just restate the conclusion in different words?
5. Which explanation do you prefer?

Answer options for questions 1-4 were yes/somewhat/no, and for question 5 were first/similar/second. The ordering of explanations were scrambled so the annotators did not know which explanation was which, and in fact they were unaware that the different explanations had been generated by different systems. We collected annotations for 100 pairs of explanations for correct answers to 100 OBQA dev questions. The annotators were recruited from our institute, spanning a broad range of age (20-60) and experience.

The results (Figure 6) suggest several findings:

1. In absolute terms, Entailer's conclusions were

[8]The distribution of proofs with different depths 1, 2, and 3 was 162, 232, and 106 respectively for OBQA, and 123, 256, and 178 respectively for QuaRTz.

[9]For a comparable datapoint on OBQA, (Brown et al., 2020) report zero-shot GPT3 scores of 57.6% and few-shot 65.4% on OBQA, so Entailer's ∼75% scores are a strong zero-shot baseline for an 11B model. More recent work using rationale-augmented prompt ensembling with much larger models (Wang et al., 2022) has reached few-shot (with rationales) scores of 91.0% (PaLM-540B) and 88.4% (GPT3-175B) on OBQA.

[10]Note that comparing with a baseline explanation generator trained on our data, i.e., just using $H \rightarrow P$ angle without verification, would not be informative, as such explanations would necessarily be worse: By definition, the verifiers remove explanations with either (believed to be) false premises and/or bad inferences, hence removing the verifiers will increase the frequency of false premises and bad inferences (confirmed through sample analysis). Hence we instead use a strong, external system as a comparative reference point.

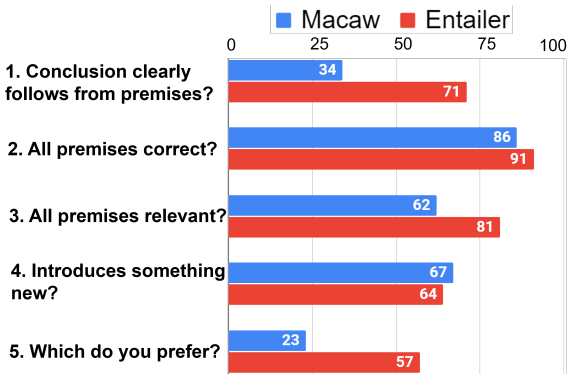

Figure 6: Users' assessments of Entailer's proofs (red), compared to Macaw's explanations (blue), showing percent of times annotators answered "Yes". Entailer's conclusions were judged to "clearly follow from the premises" in over 70% of the proofs (first bar), substantially more than Macaw's explanations (34%).

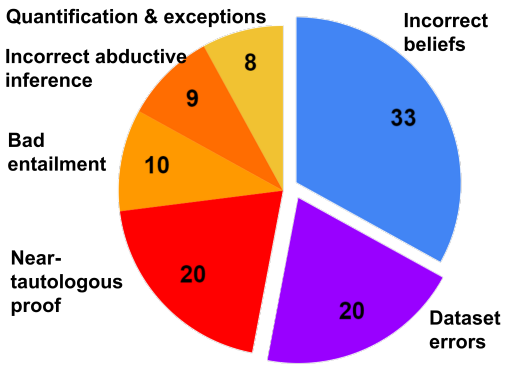

Figure 7: Causes of Entailer's failures (%) among belief (blue), reasoning (red shades), and dataset (purple).

judged to **clearly follow from the premises in the large majority (over 70%)** of the explanations, and substantially more than Macaw's explanations (34%). This potentially contributes to system trustworthiness, where understanding *how* evidence supports a conclusion is critical.

2. ≈90% of Entailer's self-verified premises were judged correct by users. Of the remainder, virtually all were labeled "unsure" (only 1 Entailer fact was labeled not correct), indicating that the there are **few false beliefs** in proofs for correct answers. Rather, vague facts (e.g., "Claws are used for cracking open shells") make up the remainder.

3. **Entailer's explanations were clearly preferred** (57% to 23%, last bar) over Macaw's. In particular, Entailer's arrangement of facts into a tree discourages irrelevant information (bar #3).

Finally we note Entailer's proofs are *faithful* (showing how an answer was derived) and *truthful* (reflecting the system's own beliefs), while Macaw's explanations are post-hoc generated text. These all suggest the value of entailment trees as trustable explanations of a system's behavior.

5.4 Analysis

5.4.1 Failure Analysis

If Entailer answers a question incorrectly, either a model belief (**belief error**) and/or the reasoning itself (**reasoning error**) must necessarily be incorrect, unless the question itself is malformed (**dataset error**). To better understand these, we classified the 51/500 cases in OBQA where Entailer selected the wrong answer while the Direct answer was correct, and found:

1. **belief errors (33%)**, where an **incorrect belief** led to a wrong answer, for example:

> In a desert... plants grow closer together **because**
> 1. A desert...contains a low amount of water.
> 2. As the amount of water decreases, plants are forced to grow closer together to conserve water.

Note that here the reasoning is correct but the second is premise is false.

2. **dataset errors (20%)**. In several cases, the question was ambiguous (e.g., does "animal" include humans?) or more than one answer option was valid (OBQA is a crowdsourced dataset).

3. **reasoning errors (47%):**

3a. **Near-tautologies (20%):** of the form "X **because** 1. X', 2. ...", where X' is a near-repeat of the hypothesis. In such cases, the proof offers little new information.

3b. **Basic reasoning errors (10%):**, e.g.,

> During ...fall, new leaves begin to grow **because**
> 1. During fall, old leaves begin to fall.
> 2. New leaves begin to grow.

Here entailment is simply invalid.

3c. **Incorrect abductive inferences (9%):** of the form A **because** $(A \rightarrow B)$ and B. While sometimes useful, these inferences are not sound and can produce incorrect conclusions, e.g.,

> Cooking food requires a fire to be built **because**
> 1. Cooking food requires heating the food.
> 2. Fire can be used to heat things.

3d. **Quantification and exceptions (8%):** where both beliefs and reasoning seem reasonable, but the specific case does not hold, e.g.,

> Seeds are often found inside a strawberry **because**
> 1. Seeds are often found inside fruits.
> 2. A strawberry is a kind of fruit.

5.4.2 When do proofs do better?

At its best, Entailer decomposes an uncertain hypothesis H into premises P which it is very confident about. For example, Entailer is unsure

whether *A suit of armor conducts electricity*, but it confidently believes the generated premises:

A suit of armor conducts electricty **because** 1. A suit of armor is made of metal 2. Metals conduct electricity

Thus an uncertain conclusion is replaced by more certain premises, and we see Entailer performing particularly well for such questions. However, there are questions that are largely "fact lookup", where a proof may be less helpful. For example, the model is already very confident about the hypothesis *Jupiter is the largest planet in the Solar System*; decomposing this into *1. The largest planet has the greatest mass* plus *2. Jupiter has the greatest mass in the solar system* has not obviously made answering easier. In fact, Entailer's algorithm is specifically designed to account for this, only growing a proof when the confidence in the premises improves the confidence in H, thus tailoring its degree of reasoning to the complexity of the problem at hand.

6 Towards Teachable Systems

If a system can show how its answers systematically follows from its own beliefs, articulating those beliefs in the process, then this opens a window into the model's internal state for the user. As a result, new opportunities arise for identifying and correcting the model's misunderstandings, so the system gradually improves over time (sometimes referred to as explanatory interactive machine learning (XIL) (Teso and Kersting, 2019)). One vehicle for doing this is to use Entailer's (currently unused) context field C at runtime: If the user asks a question, receives a wrong answer, and sees an incorrect belief in the proof, they would provide the corrected belief, then re-ask the question with the corrected belief in the context. This encourages the model to use the corrected belief in its new answer and proof, rather than just repeat the same bad belief. Such overrides would then be stored and retrieved from a persistent memory to use for future questions also. A simple, hypothetical dialog illustrating this is shown in Figure 8. This is an exciting avenue made possible by this work, currently used by the TeachMe system(Mishra et al., 2022).

U: Can a magnet attract a penny?
Entailer tries to prove H (A magnet attracts a penny) and the negation neg(H), and reports the best proof
E: A magnet can attract a penny ***because:***
1. A penny is made of copper
2. Copper is magnetic
Do you agree?
Here Entailer gives the wrong answer, arising from an incorrect belief #2. The user offers a correction:
U: No. Copper is not magnetic
System tries again, trying to prove H and neg(H) with context C = "Copper is not magnetic". C biases the model away from its prior mistake, effectively changing the model's belief. The new best proof is reported.
E: A magnet cannot attract a penny ***because:***
1. A penny is made of copper.
2. Copper is not magnetic
Do you agree?
The system now gets the right answer, having learned that copper is not magnetic from the user. C is added to memory for use in future questions, via information retrieva.
U: Yes

Figure 8: A hypothetical dialog illustrating how a user (**U**) might identify and correct Entailer's (**E**) incorrect beliefs through interaction. Here Entailer initially gets the wrong answer due to an incorrect model belief ("Copper is magnetic"). The user offers a correction, which is then provided as context when re-asking the question, effectively overriding the prior bad belief (the user has "taught" the model). By storing such corrections in a memory, such belief updates persist over time.

7 Conclusion

Our goal is a system that can show how its answers systematically follow from its own internal beliefs, and materialize those beliefs in the process. Entailer is, to our knowledge, the first system to demonstrate this capability, achieved using an "over-generate and verify" approach for each backward-chaining step. Our evaluation suggests that Entailer's proof-based answer accuracy is similar to the "direct" QA accuracy, with the additional benefit of providing a faithful, truthful chain of reasoning showing how its answers follow from its internal beliefs. The significance of this is that these chains provide a window into the model's internal beliefs and their relationships, allowing users to both verify a system's answer, or if the system provides an incorrect answer, to identify the incorrect belief leading to that error. This in turn offers new opportunities for a user to correct the system's misunderstanding by overriding a faulty belief, e.g., by adding a memory of user corrections/overrides (Tandon et al., 2022), or by editing the model parameters directly (Mitchell et al., 2021), a step towards interactive systems that can both explain to, and learn from, users over time (Mishra et al., 2022). Entailer data and models are available at https://allenai.org/data/entailer. We look forward to future developments in these directions.

Limitations

We have shown how a neural system can expose how its answers systematically follow from its own internal beliefs, providing a window into the model's system of beliefs. While exciting, there are several limitations with the current work and opportunities for the future.

First, the system is not perfect at generating coherent chains of reasoning, sometimes producing entailments that are invalid or nearly tautologous (Section 5.4.1 and Figure 7). Improved proof generation and scoring techniques would help address this.

Second, like others, we use textual entailment as the basic reasoning operation, but the definition of entailment remains somewhat imprecise (a valid entailment is one that "a person would typically infer." (Dagan et al., 2013)), contributing to noise in the model's training data. A more precise characterization of reasoning validity would help in both generation and evaluation of reasoning chains.

Third, we assume the model is generally consistent about its beliefs, but in some cases the model may verify contradictory statements, making it less clear what the model actually believes in such cases. We currently do not handle such situations. Use of a global notion of belief (rather than per question) would be a valuable avenue to explore, e.g., (Kassner et al., 2021).

Fourth, as a practical matter, recursive construction of proofs is computationally expensive (≈360 seconds/question for up to depth-3 proofs for 4-way multiple-choice, Appendix A.2). Improvements to the search algorithm would allow faster experimentation, and also help deploy the model in a practical setting.

Finally, we have speculated that showing users faithful, truthful chains of reasoning might be a basis for a conversational system, where users could correct and teach the system in cases where it was wrong. However, this is currently just a conjecture - futures explorations into how this might be realized would be valuable.

Ethics Statement

Like any other large-scale language model, despite the best intentions, there is a risk of our model producing biased or offensive statements as part of its explanations. We release our models for research purposes only.

Acknowledgements

This research was made possible, in part, by funding from Open Philanthropy. We also thank Google for providing the TPUs for conducting experiments.

References

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In ACL.

Kaj Bostrom, Zayne Sprague, Swarat Chaudhuri, and Greg Durrett. 2022. Natural language deduction through search over statement compositions. ArXiv, abs/2201.06028.

Kaj Bostrom, Xinyu Zhao, Swarat Chaudhuri, and Greg Durrett. 2021. Flexible generation of natural language deductions. In EMNLP.

Tom B. Brown et al. 2020. Language models are few-shot learners. In NeurIPS.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In IJCAI'20.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. ArXiv, abs/2110.14168.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. ArXiv, abs/2205.09712.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing Textual Entailment: Models and Applications. Morgan and Claypool.

Bhavana Dalvi, Peter A. Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In EMNLP.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. ArXiv, abs/1809.02922.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, E. Hovy, H. Schutze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. ArXiv, abs/2102.01017.

Saadia Gabriel, Chandra Bhagavatula, Vered Shwartz, Ronan Le Bras, Maxwell Forbes, and Yejin Choi. 2021. Paragraph-level commonsense transformers with recurrent memory. In AAAI.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In ICLR.

Ruixin Hong, Hongming Zhang, Xintong Yu, and Changshui Zhang. 2022. Metgen: A module-based entailment tree generation framework for answer explanation. ArXiv, abs/2205.02593.

Aditya Kalyanpur, Tom Breloff, and David A. Ferrucci. 2020. Braid: Weaving symbolic and neural knowledge into coherent logical explanations. arXiv: Computation and Language.

Nora Kassner and H. Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In ACL.

Nora Kassner, Oyvind Tafjord, Hinrich Schutze, and Peter Clark. 2021. BeliefBank: Adding memory to a pre-trained language model for a systematic notion of belief. In EMNLP.

Tao Li, Vivek Gupta, Maitrey Mehta, and Vivek Srikumar. 2019. A logic-driven framework for consistency of neural models. In EMNLP.

Yujia Li, David H. Choi, et al. 2022. Competition-level code generation with alphacode. ArXiv, abs/2203.07814.

Christopher D. Manning and Bill MacCartney. 2009. Natural language inference. Stanford University.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In EMNLP.

Bhavana Dalvi Mishra, Oyvind Tafjord, and Peter Clark. 2022. Towards teachable reasoning systems: Using a dynamic memory of user feedback for continual system improvement. In EMNLP.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2021. Fast model editing at scale. ArXiv, abs/2110.11309.

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. ArXiv, abs/2112.00114.

F. Petroni, Tim Rocktäschel, Patrick Lewis, A. Bakhtin, Y. Wu, Alexander H. Miller, and S. Riedel. 2019. Language models as knowledge bases? In EMNLP.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Rui Dong, Xiaokai Wei, Henry Zhu, Xinchi Chen, Zhiheng Huang, Peng Xu, Andrew O. Arnold, and Dan Roth. 2022. Entailment tree explanations via iterative retrieval-generation reasoner. ArXiv, abs/2205.09224.

Marco Tulio Ribeiro, Carlos Guestrin, and Sameer Singh. 2019. Are red roses red? Evaluating consistency of question-answering models. In ACL.

Eric Schwitzgebel. 2019. Belief. Stanford Encyclopedia of Philosophy. Https://plato.stanford.edu/entries/belief/.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In EMNLP, pages 4615–4629.

Oyvind Tafjord and Peter Clark. 2021. General-purpose question-answering with Macaw. ArXiv, abs/2109.02593.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019. QuaRTz: An open-domain dataset of qualitative relationship questions. ArXiv, abs/1909.03553.

Oyvind Tafjord, B. D. Mishra, and P. Clark. 2020. ProofWriter: Generating implications, proofs, and abductive statements over natural language. ArXiv, abs/2012.13048.

Alon Talmor, Oyvind Tafjord, P. Clark, Y. Goldberg, and Jonathan Berant. 2020. LeapOfThought: Teaching pre-trained models to systematically reason over implicit knowledge. In NeurIPS.

Niket Tandon, Aman Madaan, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In ACL Workshop on Commonsense Representation and Reasoning (CSRR'22). (also arxiv:2201.06009).

Stefano Teso and Kristian Kersting. 2019. Explanatory interactive machine learning. Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Rationale-augmented ensembles in language models. ArXiv, abs/2207.00747.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. ArXiv, abs/2201.11903.

Sarah Wiegreffe and Ana Marasović. 2021. Teach me to explain: A review of datasets for explainable NLP. ArXiv, abs/2102.12060.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter Jansen. 2020. WorldTree V2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In LREC.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. ArXiv, abs/2205.12443.

Appendix A. Entailer's Backward Chaining Algorithm

Algorithm 1 Entailer's backchaining algorithm for searching for the best proof $tree(H)$ with score $s(H)$ for a hypothesis H.

1: **procedure** PROVE(H) $\rightarrow$ score $s(H)$ & proof $tree(H)$:
2: i. **find** direct score $s_d(H)$ & confidence $c_d(H)$:
3: direct score $s_d(H)$ = **call** model $H \rightarrow S_d$
4: direct confidence $c_d(H) = max(s_d(H), (1 - s_d(H)))$ *// confidence of predicted label*
5: ii. **find** reasoned score $s_r(H)$, confidence $c_r(H)$, & subtrees $tree(p_i)$:
6: **call** 1STEP(H) $\rightarrow P$ to find best premises P that entail H
7: **find** $s_e(P \vdash H)$ via model $PH \rightarrow S_e$ *// score the entailment*
8: **if** $s_e(P \vdash H) > c_d(H)$ **or** H=$H_0^\dagger$ *// if reasoning conf $c_r(H)$ might beat direct conf $c_d(H)$ (lines 13,11), backchain*
9: **then forall** $p_i \in P$ **do**
10: **call** PROVE(p_i) to find $s(p_i), tree(p_i)$ *// get scores and subtrees for each p_i*
11: reasoned score $s_r(H) = (\Pi_i s(p_i)).s_e(P \vdash H)$ *// overall score for reasoned answer*
12: **else** $s_r(H) = 0$
13: reasoning confidence $c_r(H) = s_r(H)$ *// reason confidence = score*
14: iii. **return** direct or reasoned answer (pick most confident):
15: **if** $c_r(H) > c_d(H)$ **or** H=$H_0^\dagger$ *// if reasoning confidence is higher (or top level)*
16: **then return** $s(H)$=$s_r(H)$ & $tree(H)$=$(\{tree(p_i)\} \vdash H)$ *// reasoned score + subtrees*
17: **else return** $s(H)$=$s_d(H)$ & $tree(H)$=H *// direct score + H as terminal node (the subtrees are discarded)*
18:
19: **procedure** 1STEP(H) $\rightarrow$ premises P *// Find premises P that together entail H*
20: **repeat** k times: *// over-generate with nucleus sampling*
21: a. find premises P via model $H \rightarrow P$ *// find premises*
22: b. find $\{s_d(p_i \in P)\}$ via model $H(= p_i) \rightarrow S_d$ *// score those premises*
23: c. find $s_e(P \vdash H)$ via model $PH \rightarrow S_e$ *// score the entailment itself*
24: d. discard if any of $\{s_d(p_i)\}, s_e(P \vdash H) < 0.5$ *// now filter*
25: e. $s_{r\text{-}1deep}(H) = (\Pi_i s_d(p_i)).s_e(P \vdash H)$ *// Final score for the 1-deep proof*
26: **select** premises P with highest $s_{r\text{-}1deep}(H)$ *// Pick the best of the k proofs*

†We insist on at least a 1-deep tree by, for the top-level hypothesis H=H_0, ensuring that lines 8 and 15 succeed.

A.1 Generating One Backward-Chaining Step

The procedure to find a 1-step inference is called 1STEP(H) in Algorithm 1 (line 19). Given a hypothesis H, we use the angle $H \rightarrow P$ to over-generate a set of k alternative backward-chaining steps $P \vdash H$ using nucleus sampling (line 21). We then check that the model believes all the premises $p_i \in P$ using the $H(= p_i) \rightarrow S_d$ angle, and that it believes the inference step $P \vdash H$ itself is valid (independent of whether the p_i are true or not) using the $PH \rightarrow S_e$ angle (line 24). The overall score, denoting how well the 1-step proof supports the hypothesis, combines the premise end entailment scores as follows (line 25):

$$s_{r\text{-}1deep}(H) = (\Pi_i s_d(p_i)).s_e(P \vdash H)$$

The highest-scoring proof $P \vdash H$ is returned.

A.2 Backward Chaining

Procedure PROVE(H) in Algorithm 1 generalizes this to multi-step entailments by recursively generating support for each premise p_i in P (line 10). The stopping condition is when the model's direct confidence[11] in p_i is greater than the proof-derived confidence in p_i, i.e., $c_d(p_i) > c_r(p_i)$. When this condition is met, the subtree $P' \vdash p_i$ for p_i is discarded and p_i becomes a leaf node of the tree (line 17). We also impose a maximum depth d on the tree.

This whole procedure is repeated for each candidate answer hypothesis (Section 3.1). Finally the system selects the answer corresponding to the hypothesis with the top-scoring proof $s(H)$.

On our datasets, the average runtime per question (4-way multiple-choice) is ≈80 seconds (depth 1 proofs, sample size k=6) or ≈360 seconds (up to depth 3 proofs, sample size k=6) on a 48GB GPU, due to the large number of candidate inference steps generated during the search.

[11] As the direct score $s_d(p_i)$ in p_i ranges from 0 (definitely false) to 1 (definitely true), we define the corresponding confidence to with respect to the predicted label, i.e., $c_d(p_i) = 1 - s_d(p_i)$ when $s_d(p_i) < 0.5$, otherwise $c_d(p_i) = s_d(p_i)$. In contrast, the proof score ranges from 0 (no proof) to 1 (perfect proof), hence the proof confidence is simply the proof score, $c_r(p_i) = s_r(p_i)$. Note that unlike $s_d(p_i)$, $s_r(p_i) = 0$ means no proof, not false.

Appendix B. Crowdsourcing Instructions for Verifying Entailments (Section 4.1.2)

Instructions (click here to collapse/expand instructions)

We have a computer program that tries to answer multiple-choice science questions by *reasoning*. To do this, it tries to combine two facts that it thinks are true, to deduce the answer to the question. For example, consider this question:

> Which of the following conducts electricity?
> (A) a pencil (B) a nail **[CORRECT]** (C) a book

Here the system got the right answer (B), and explains its reasoning as follows:

> **BECAUSE** a nail is made of metal
> **AND** metals conduct electricity
> **IT FOLLOWS THAT** a nail conducts electricity;

This is an example of a **correct** line of reasoning: if nails are metal, and metals conduct electricity, then nails will conduct too.

However, the computer often makes mistakes! **All** the examples in this HIT are when the computer chose the **wrong** answer to the question!

This HIT is to diagnose where the computer went wrong. It could be that

- one of the facts the computer started with was wrong, and/or
- its reasoning was wrong (i.e., the conclusion simply doesn't follow from the facts) and/or
- in the special case where the facts *and* the reasoning look correct, there could be some additional, missing fact which the computer forgot to take into account.

Simply check the boxes to help us diagnose the problem! Here are three examples of the HIT:

EXAMPLE 1:

First read this **SCIENCE QUESTION:**

> Which animal can fly?
> (1) eagle **[CORRECT]** (2) dog (3) cat

Here, (1) is the correct answer. But the system got it wrong, and thought the answer was (2)! Here is its line of reasoning - please help debug it! First, let's check the **individual facts** that the system guessed:

	Is the sentence generally true?			
	Yes	**No**	**Unsure**	
BECAUSE dogs are animals	◉	○	○	
AND animals can fly	○	◉	○	Note this is not generally true! So select "No"
IT FOLLOWS THAT dogs can fly ← This one is false!				

Now let's check the system's **reasoning**. **IMAGINE** that all the facts *were* true:

> **IF IT WERE TRUE THAT** dogs are animals
> **AND IT WERE TRUE THAT** animals can fly
> **THEN WOULD IT FOLLOW THAT** dogs can fly

Would the conclusion follow ***IF*** all the facts were true?

◉ yes ○ no ○ it depends ○ unsure/incomprehensible

Finally: **If** *all* the facts are correct, **and** the reasoning is correct, then where did the system go wrong?

- ◉ Not applicable - some of the system's facts and/or reasoning *were* wrong.
- ○ I don't see a problem - this "wrong" answer actually looks correct to me!
- ○ While the facts are *generally* correct, they don't hold for this particular situation
- ○ There's some additional, missing fact the system should have used (enter it below):

Comment: Here, this special condition (facts and reasoning both correct) doesn't apply, as the second fact ("animals can fly") is *not* generally true. Hence select "Not applicable".

[EXAMPLES 2 and 3 ommited here for brevity]

Some important notes:

- *Please answer with care:* ***Some HITs will be checked by hand, and work may be rejected if there are to many errors***
- *Feel free to use the Internet/Google to make sense of the science question and what the correct answer is.*
- *To select "Yes", the sentence only needs to be generally true (in a commonsense way), not absolutely always true.*
- *Feel free to use the Internet/Google to check if a fact's true or not.*
- *If a fact is ungrammatical but you can still understand it, score it as if it were ungrammatical.*
- *If the sentence doesn't make sense / is incomprehensible / is very ambiguous, select "Unsure".*
- *Ignore upper-case/lower-case differences. For example, you would mark the fact:*
  - ***BECAUSE:** o is the chemical symbol for oxygen...*

  as generally true ("Yes"), even though strictly it should say "O" (upper-case) rather than "o" (lower-case) for oxygen.
- *To check the reasoning, imagine someone convinces you that all the facts are true (even if they are not). If they convinced you, would you therefore believe the conclusion? If yes, then the reasoning is good. Otherwise it is not!*
- *Or to put it another way: Checking the reasoning requires "suspending disbelief" in incorrect facts. Does the chain of reasoning seem convincing, if the facts were correct?*
- ***All** the conclusions ("IT FOLLOWS THAT") are intended to be false ("This one is false!"). If you see one which looks true to you, and the facts and reasoning are correct, then select "I don't see a problem" for the last question!*

Thank you for your help! You rock!

Appendix C: Model Training

C.1 Dataset Preparation

Here we describe in detail how Entailer's training data is assembled.

1. We start with the training partition of the EntailmentBank dataset, containining 1313 multiple choice questions each with an entailment tree for their correct answer choice.

2. We convert the question + each answer option to a hypothesis using four different, alternative methods:
   - Entailer's current QA2D model, a reconstruction of that by (Demszky et al., 2018) (Section 3.1).
   - An in-house, rule-based QA2D utility
   - An earlier version of Entailer's QA2D model.
   - The original hand-written hypothesis supplied in the EntailmentBank dataset.

   If a source generates the same hypothesis for two different answer choices, we choose not to trust it for that question and discard it.

3. We "shred" the EntailmentBank entailment trees into individual 1-step P $\vdash$ H entailments, producing 4175 1-step (valid) entailments using $\approx$9000 true premises.

4. The crowdsourced P $\vdash$ H instances, with annotations on whether premises are true and entailment is true, are also added (Section 4.1.2).

5. Each EntailmentBank proof also comes with a set of associated relevant facts (sentences), only some of which are used in the entailment proofs. We use these to create a "relevant context" containing these sentences, sorted into two buckets:
   high-relevance: sentences actually used in the proof of the correct answer
   medium-relevance: the remaining sentences

   This context is the same across all answer options at this point.

6. For every hypothesis and premise appearing in this dataset, we run a simple BM25 IR search against a larger science text corpus (about 1.5 million sentences from a science-filtered Wikipedia) to obtain noisier sentences to use as a **low-relevance** context.

7. For the final training dataset, we create 4 different contexts, using 4 sampling strategies. First we create a "full" context of sentences sorted into three buckets, with added noise:
   high: high-relevance facts + 10% chance of a random sentence from medium/low
   medium: medium-relevance facts + 20% chance of random sentences from high/low
   low: 5 low-relevance facts + 20% chance of random sentence from high/medium

   Then, we use four sampling strategies to create actual contexts for the Entailer training data. Each strategy is defined by the per-sentence chance of a sentence coming from one of the high/medium/low buckets. For example, a context sampled with 0.2/0.4/0.6 means that 20% of its sentences came from high, 40% from medium, an 60% from low. If no sentence is selectable (e.g., a bucket is exhausted), we use one from low. The final context is syntactically expressed as "[HIGH] <*high sentences*> [MEDIUM] <*medium sentences*> [LOW] <*low sentences*>". The 4 sampling strategies are:
   1/1/1 (full context, all available sentences)
   0.5/0.5/0.5 (half of sentences from each category)
   0.2/0.4/0.6 (more emphasis on lower categories)
   0/0/1 (only low sentences)

   For the training set we store these as a list of contexts, to be sampled at random when generating the training instances. We do the same for hypotheses, so these are also sampled at random.

8. The model is trained across the following angles, each sampled equally: {H$\rightarrow$P, H$\rightarrow$V, HP$\rightarrow$I, QAH$\rightarrow$P, QAH$\rightarrow$V, QAHP$\rightarrow$I, HC$\rightarrow$P, HC$\rightarrow$V, HPC$\rightarrow$I, QAHC$\rightarrow$P, QAHC$\rightarrow$V, QAHPC$\rightarrow$I}

   This is just the core angles H$\rightarrow$P, H$\rightarrow$V, HP$\rightarrow$I, with optional QA and/or C added. The individual premise verification (leaves) uses the H$\rightarrow$V angle (where H is now a premise), for those angles we don't have associated proofs P so we limit to the angles: H$\rightarrow$V, QAH$\rightarrow$V, HC$\rightarrow$V, QAHC$\rightarrow$V

   The full dataset is provided in the supplementary material, and will be released publically.

C.2 Model Details

We train a T5-11B multi-angle model, Entailer, following the multi-task setup similar to (Tafjord and Clark, 2021) for the three functionalities described in Table 1. We used the default hyperparameters (including the Adafactor optimizer) in the T5 library,fine-tune the model for 20K steps with batch size of 8, selecting the checkpoint with highest validation score (usually the final step).

At inference time, Entailer's primary function is to generate candidate entailments for each answer choice for a given question, which are then scored by the same model using its two verification angles. To generate multiple explanations for a single input (answer hypothesis, IR retrieved context), in addition to the greedy decoding we use nucleus sampling (Holtzman et al., 2019). For the experiments in this paper we set "temperature= 2.0", and "top_p= 0.95".

Appendix D. Examples of Macaw Explanations and Entailer Proofs

A random selection of explanations, both good and bad, from the two systems (Section 5.3).

OBQA Question	Macaw Explanation (Free-Form Sentences)	Entailer Explanation (1-deep Entailment Tree)
An example of data collection is (A) Deleting case files on the computer (B) Touching evidence without gloves (C) speaking with a witness (D) Throwing documents in the trash	An example of data collection is speaking with a witness. BECAUSE: - Data collection is when a researcher speaks with a witness.	An example of data collection is speaking with a witness. BECAUSE: - An example of data collection is interviewing a witness. - Interviewing a witness requires speaking with that witness.
An animal can hunt by cracking open a (A) claw (B) house (C) shell (D) bone	An animal can hunt by cracking open a shell. BECAUSE: - A shell is hard. - Claws are used for cracking open shells.	An animal can hunt by cracking open a shell. BECAUSE: - An animal can catch prey by cracking open a shell. - If an animal catches prey then that animal hunts that prey.
Plants need light for (A) love (B) glucose (C) reading (D) driving	Plants need light for glucose. BECAUSE: - A plant requires sunlight for photosynthesis. - A plant requires sunlight for photosynthesis.	Plants need light for glucose. BECAUSE: - Plants require light for photosynthesis. - Photosynthesis produces glucose.
which of these animal's habitat is peat likely to be found? (A) a dog's habitat (B) a cat's habitat (C) a human's habitat (D) an alligator's habitat	Peat is likely to be found in an alligator's habitat. BECAUSE: - Peat is found in swamps. - A swamp is a kind of environment.	Peat is likely to be found in an alligator's habitat. BECAUSE: - Alligators live in swampy environments. - Peat is likely to be found in a swampy environment.
Squirrels store acorns (A) they build their homes from them (B) they use them to keep warm (C) to have nourishment thru the cold months (D) because they like the aesthetic	Squirrels store acorns to have nourishment thru the cold months. BECAUSE: - An acorn is a kind of food for squirrels. - An animal needs to eat food for nutrients. - An animal requires enough nutrients to maintain good health. - An animal needs to eat food for nutrients.	Squirrels store acorns to have nourishment thru the cold months. BECAUSE: - Squirrels store acorns for the winter. - An animal requires nutrients for survival during the winter.
What could be used as an electrical conductor (A) A penny (B) Shoe laces (C) Wood (D) A button	A penny could be used as an electrical conductor. BECAUSE: - Metal is an electrical conductor.	A penny could be used as an electrical conductor. BECAUSE: - A penny is made of copper. - Copper is an electrical / electrical energy conductor.
plants are the source of (A) turkeys (B) tomatoes (C) cows (D) chickens	Plants are the source of tomatoes. BECAUSE: - A tomato is a kind of fruit. - Fruit contains seeds. - A plant is a kind of producer. - A producer is a source of food energy in an ecosystem.	Plants are the source of tomatoes. BECAUSE: - Plants are the source of food. - Tomatoes are a kind of food.
A source of heat at the resort is the (A) jacuzzi (B) pool (C) chair (D) umbrella	A source of heat at the resort is the jacuzzi. BECAUSE: - A hot tub is a source of heat.	A source of heat at the resort is the jacuzzi. BECAUSE: - A jacuzzi is a kind of hot tub. - A hot tub is a source of heat.
Which is recyclable (A) fiberglass (B) soda can (C) Styrofoam food container (D) motor oil	A soda can is recyclable. BECAUSE: - A soda can is made of aluminum. - Aluminum is a kind of material. - Recyclability is a property of a material and includes ordered values of recyclable.	A soda can is recyclable. BECAUSE: - Aluminum cans are recyclable. - A soda can is made of aluminum.
A lake, pond, or stream can all be used to (A) collect fluid from (B) make extra food (C) rain water down (D) create new houses	A lake, pond, or stream can all be used to collect fluid from. BECAUSE: - A body of water is a kind of environment. - Aquatic plants live in bodies of water. - A stream is a kind of body of water. - A lake is a kind of body of water.	A lake, pond, or stream can all be used to collect fluid from. BECAUSE: - A lake, pond, or stream is a source of fluid. - A source of something can be used to collect that something.