Generating Natural Language Proofs with Verifier-Guided Search

Kaiyu Yang, Jia Deng, and Danqi Chen
Department of Computer Science, Princeton University

Deductive Reasoning in Natural Language
How to draw conclusions from assumptions in natural language?
• Challenges
  1. Fuzzy, imprecise, requiring implicit knowledge
  2. No finite, well-defined inference rules as in formal logic
  3. Difficult for large pretrained language models

Generating Natural Language Proofs
• Task
  1. Input: a hypothesis \( h \) and a set of supporting facts \( C = \{s_{e1}, s_{e2}, \ldots, s_{en}\} \) in natural language
  2. Output: a proof tree \( T \) for deriving \( h \) from a subset of \( C \)
    a. The root node is \( h \); the leaf nodes are sentences in \( C \)
    b. Others are intermediate conclusions generated by the model

Our Method: NLProofS
• Overview
  1. Training: train a stepwise prover for generating candidate steps and a verifier for scoring the validity of steps
  2. Inference: search for proofs with high aggregated proof scores

• Stepwise prover
  1. Similar to existing methods for stepwise proof generation
  2. Finetune a T5 model to predict the next step
  3. Generate multiple candidate steps via beam search

• Verifier
  1. An independently trained neural network for checking the validity of proof steps, producing a score in \([0, 1]\)

• Proof search
  1. Initialize the proof graph with a greedy proof from the prover
  2. Sample a partial proof from the graph
  3. Generate multiple candidate steps using the prover
  4. Execute them to update the graph and keep track of scores produced by the prover/verifier
  5. Return the proof with the maximum proof score

Contributions
• NLProofS (Natural Language Proof Search)
  1. A new method for stepwise proof generation
  2. Generate relevant proof steps conditioning on the hypothesis
  3. Train an independent verifier to prevent hallucination

Experiments
• EntailmentBank\(^{(2)}\)
  1. Challenging, human-authored proofs
  2. Four evaluation metrics: Leaves, Steps, Intermediates, Overall
  3. We outperform existing methods
  4. Verifier-guided search is important for the improvement

<table>
<thead>
<tr>
<th>Method</th>
<th>Leaves</th>
<th>Steps</th>
<th>Intermediates</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntailmentWriter(^{(1)})</td>
<td>35.6</td>
<td>22.9</td>
<td>28.5</td>
<td>20.9</td>
</tr>
<tr>
<td>IRGR(^{(3)}) (concurrent work)</td>
<td>23.8</td>
<td>22.3</td>
<td>26.5</td>
<td>22.0</td>
</tr>
<tr>
<td>MetGen(^{(4)}) (concurrent work)</td>
<td>48.6</td>
<td>30.4</td>
<td>32.7</td>
<td>28.0</td>
</tr>
<tr>
<td>NLProofS (ours)</td>
<td>58.8</td>
<td>34.4</td>
<td>37.8</td>
<td>33.3</td>
</tr>
<tr>
<td>w/o search</td>
<td>56.5</td>
<td>33.7</td>
<td>36.4</td>
<td>31.8</td>
</tr>
<tr>
<td>w/o search w/o stepwise</td>
<td>45.6</td>
<td>29.7</td>
<td>32.2</td>
<td>27.1</td>
</tr>
<tr>
<td>w/o verifier score</td>
<td>55.8</td>
<td>33.8</td>
<td>36.1</td>
<td>31.9</td>
</tr>
</tbody>
</table>

The verifier helps
1. Mitigate hallucination
2. Avoid copying premises as the conclusion

<table>
<thead>
<tr>
<th>Method</th>
<th>Answer</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>FailRR(^{(1)})</td>
<td>99.2</td>
<td>98.8</td>
</tr>
<tr>
<td>ProofWriter(^{(1)})</td>
<td>99.8</td>
<td>99.7</td>
</tr>
<tr>
<td>NLProofs (ours)</td>
<td>99.3</td>
<td>99.2</td>
</tr>
</tbody>
</table>

RuleTaker\(^{(1)}\)
1. Simple, synthetic proofs generated by templates
2. We perform competitively with existing methods

Other observations
1. The prover might be the main bottleneck
2. Long proofs remain challenging


Experiments