Applying GPT-3 and Dense Embeddings to NLProofS

Roma Bhattacharjee †, Arya Maheshwari †, Zachary S. Siegel †

Princeton University, † Equal Contribution

Background

A fascinating problem within NLP is proof generation and automated reasoning. This involves generating a multistep proof of a given hypothesis from a set of assumed statements or premises.

To more easily research this problem, Dalvi et al. [1] curated a dataset called EntailmentBank, derived from the WorldTree V2 corpus [6], which contains multistep entailment trees. In their paper, they also break the proof generation problem into 3 distinct tasks of varying difficulty: in *Task 1*, the only premises provided are those in the ground truth proof tree, and the task is to determine which intermediate steps to generate. *Task 2* provides 25 premises, which includes several "distractor" premises in addition to the required premises. Here, the system must also discern which facts are relevant and which are distractors, as well as generate the correct proof tree. *Task 3* is a larger version of *Task 2*—it provides as input all 12K premises from WorldTree, from which the proof generation system must retrieve premises to use for a given hypothesis. Follow-up work (including ours) follows this three-task framework in the evaluation of proof generation models.

Related Work

- In addition to curating EntailmentBank, Dalvi et al. trained BERT and RoBERTa-based models to perform retrieve relevant premises for Task 3. Given a hypothesis, the retrieval model returns a set of 25 premises from the corpus.
- Yang et al.'s NLProofS paper [7] introduces (1) a separate verifier model to score proof steps generated by their prover (to prevent "hallucinated" steps) and (2) a search algorithm to find the highest-scoring proof.

Overview

We conduct **two separate studies** based on Yang et. al's NLProofS model:

1. **Replacing NLProofS's T5 prover with GPT-3 + in-context learning.** NLProofS's architecture implements a stepwise prover by fine-tuning a T5 model. While Yang et al. did show that using GPT-3 to generate the entire proof at once ("single-shot") performed worse than NLProofS, they did not try using GPT-3 with in-context learning to only generate stepwise proofs. Thus, this study investigates NLProofS's performance after replacing just the T5 prover model with GPT-3.
2. **Retrieval using (dense) sentence embeddings.** Yang et al. do not explicitly focus on retrieval; in evaluating their Task 3 performance, they simply use the same 25 premises returned by Dalvi et al.'s RoBERTa-based retrieval model. This second study evaluates NLProofS's performance with alternative retrieval methods that use pre-trained dense sentence embeddings to see if gold tree premise recall can be improved.

Acknowledgements and References

We thank Prof. Danqi Chen for her guidance throughout this project and Tianyu Gao for correspondence regarding SimCSE.

[1] Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. Explaining answers with entailment trees. *EMNLP*, 2021.

[2] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[3] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning, 2021.

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

[5] Minghan Li Barlas Oguz Jimmy Lin Yashar Mehdad Wen-tau Yih Xilun Chen Sheng-Chieh Lin, Akari Asai. How to train your dragon: Diverse augmentation towards generalizable dense retrieval. *arXiv e-print 2302.07452*, 2023.

[6] Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter Jansen. WorldTree v2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5456–5473, Marseille, France, May 2020. European Language Resources Association.

[7] Kaiyu Yang, Jia Deng, and Danqi Chen. Generating natural language proofs with verifier-guided search. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Study 1: Methods

In the first study, we replace components of the prover with `gpt-3.5-turbo`. We theorize that since `gpt-3.5-turbo` was trained on a much larger set than the T5 model, it could use additional background knowledge to help at the proof completion task.

We train `gpt-3.5-turbo` using in-context learning since fine-tuning is cost prohibitive. We utilize the same training data and dataloader that the original NLProofS methodology loaded, and we randomly sample proof examples to feed into `gpt-3.5-turbo`. The number of training examples is limited by the maximum number of tokens `gpt-3.5-turbo` can take, 4096 tokens. This token limit allows for a maximum of about 8 examples.

In the first experiment, we replace the entire `T5` model with `gpt-3.5-turbo` for both the initial greedy tree generation and for the search method. However, as seen in the Table 1, this model performs poorly.

In the second experiment, we want to lower bound the performance so that our modifications will not cause the model to perform worse than before. We elect to use a "hybrid" model, where we keep `T5` for initial greedy tree generation, and then `gpt-3.5-turbo` in the tree search algorithm to improve upon the initial generated tree. We run these experiments as a function of the number of in-context examples.

Study 2: Methods

In this second study, we investigate the effects of replacing the premises used by Yang et al. for Task 3 (obtained via the outputs of Dalvi et al.'s retrieval) with an alternate approach that employs sentence embeddings to retrieve premises from the WorldTree corpus for a given hypothesis.

The main idea is to map all facts in the WorldTree corpus $\mathcal{C}$ and a given hypothesis h to dense vectors using a sentence embedding model, and to then score the relevance of candidate premises to the hypothesis through a vector similarity metric.

We evaluate three sentence embedding models for this task, pretrained on other (different) information retrieval tasks: `SimCSE` [2], `Contriever` [3], and `DRAGON-RoBERTa` [5].

We first employ these models to implement a baseline retrieval algorithm which simply returns the top k premises that, as embeddings, yield the highest cosine similarity score to the hypothesis embedding (see details in Algorithm 1; we use $k = 25$ like in Yang et al.). We evaluate performance by computing recall ("Recall@25") of the gold proof tree's premises (see Table 2).

Manual inspection of Alg. 1 results showed that the gold tree premises missed by our retrieval were often shorter facts less similar to the full hypothesis but necessary as building blocks in the final reasoning. Thus, to refine our approach of selecting solely for similarity to the full hypothesis h, we implement a modified algorithm that splits h into halves (by number of words) and selects some premises based on similarity *to each half* of h. See Algorithm 2 for details.

Finally, we evaluate each of these retrieval methods by using them in the full Task 3 NLProofS pipeline. In particular, we run the NLProofS model trained on Task 2 and evaluate its zero-shot performance on Task 3, using each set of retrieved premises obtained according to various combinations of the embedding models and retrieval algorithm (Alg. 1 or Alg. 2).

Algorithm 1 Baseline Embedding Retrieval

Input: corpus $\mathcal{C}$, hypothesis h, embedding model Emb, context size k
Output: list of k premises for h

$\mathbf{h} \leftarrow Emb(h)$
$\mathbf{C} := \{\mathbf{c}_i\}_{1 \leq i \leq |\mathcal{C}|} \leftarrow Emb(\mathcal{C})$
scores $\leftarrow \emptyset$ `(dict)`
for each $c_i \in \mathcal{C}$ **do**
  scores$[c_i]$ = $sim_{cos}(\mathbf{c}_i, h)$
end for
scores $\leftarrow$ `sort`(*scores*)
  in descending order by value
return top k keys of *scores*

Notes on **Algorithm 2**: we select w of the final k premises based on similarity to each half of h (requiring $2w \leq k$) while still selecting $k - 2w$ premises based on the full h. We check that premises are not selected multiple times (e.g. when selecting based on h_r, rule out all premises already selected based on h_l) to ultimately obtain k distinct premises.

Algorithm 2 Split-Hypothesis Embedding Retrieval

Input: corpus $\mathcal{C}$, hypothesis h, embedding model Emb, context size k, each-half focused context size w
Output: list of k premises for h

$\ell_{\text{full}} \leftarrow$ Retrieve($\mathcal{C}$, h, Emb, $k - 2w$)
$\mathbf{h_l}, \mathbf{h_r} \leftarrow Emb$(left half of h), Emb(right half of h)
$\mathbf{C} := \{\mathbf{c}_i\}_{1 \leq i \leq |\mathcal{C}|} \leftarrow Emb(\mathcal{C})$
scores$_l$, *scores*$_r$ $\leftarrow \emptyset$ `(dict)`
for each $c_i \in \mathcal{C}$ **do**
  scores$_l[c_i]$ = $sim_{cos}(\mathbf{c}_i, h_l)$
  scores$_r[c_i]$ = $sim_{cos}(\mathbf{c}_i, h_r)$
end for
scores$_l$ $\leftarrow$ `sort`(*scores*$_l$) *by value*
scores$_r$ $\leftarrow$ `sort`(*scores*$_r$) *by value*
ℓ_l = top w keys in *scores*$_l$ (*)
ℓ_r = top w keys in *scores*$_r$ (*)
  (*) computed s.t. ℓ_l, ℓ_r, ℓ_{full} *are all distinct*
return $\ell_{full} \cup \ell_l \cup \ell_r$

Study 1: Results

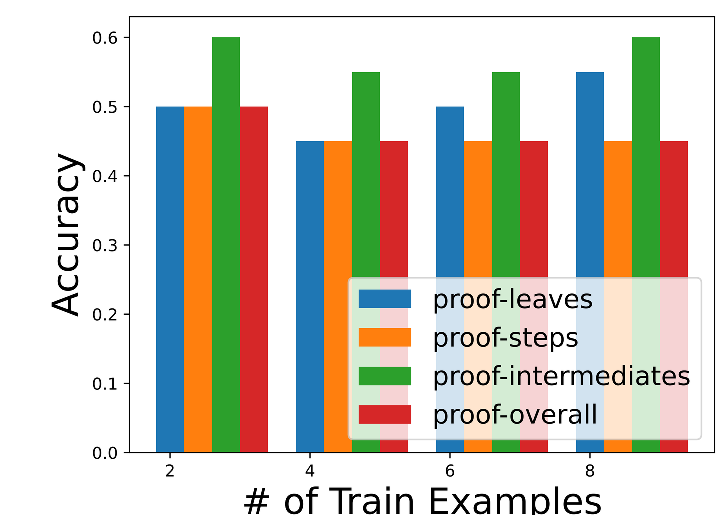

Figure 1. Few Shot Performance of T5 (Greedy) + GPT (Search)

Prover Model	Leaves		Steps	
	F1	AllCorrect	F1	AllCorrect
T5 (Greedy + Search)	**86.21**	**50.00**	<u>43.15</u>	<u>35.00</u>
T5 (Greedy) + GPT (Search)	<u>81.21</u>	<u>46.67</u>	**45.40**	**36.67**
GPT (Greedy + Search)	63.74	20.00	15.00	10.00

Prover Model	Intermediates		Overall
	F1	AllCorrect	AllCorrect
T5 (Greedy + Search)	**68.15**	<u>41.67</u>	<u>35.00</u>
T5 (Greedy) + GPT (Search)	<u>66.84</u>	**43.33**	**36.67**
GPT (Greedy + Search)	50.38	10.00	10.00

Table 1. Results on incorporating GPT-3 prover into NLProofS pipeline. Best performance **bolded**, runner-up <u>underlined</u>.

Study 2: Results

Model	Retrieval Algorithm	Recall@25
Dalvi et. al* [1]	–	**0.732**
SimCSE	Baseline (Alg. 1)	0.570
Contriever	Baseline (Alg. 1)	0.597
Contriever	Split-Hyp (Alg. 2)	0.603
DRAGON-RoBERTa	Baseline (Alg. 1)	0.615
DRAGON-RoBERTa	Split-Hyp (Alg. 2)	<u>0.643</u>

Table 2. Recall of gold tree premises for each retrieval model. * indicates that model was trained on the EntailmentBank dataset. Best recall **bolded**, runner-up <u>underlined</u>.

Retrieval Method	Leaves		Steps		Intermediates		Overall
	F1	AllCorrect	F1	AllCorrect	F1	AllCorrect	AllCorrect
Dalvi	**47.73**	**14.97**	**15.89**	**11.23**	**46.45**	**20.32**	**11.23**
SimCSE	34.65	8.02	8.65	6.95	36.96	15.51	6.95
DRAGON-RoBERTa	37.38	7.49	9.70	5.88	40.60	<u>17.11</u>	5.88
DRAGON-RoBERTa (split)	<u>39.96</u>	<u>9.09</u>	10.43	6.95	<u>43.25</u>	16.04	6.95
Contriever	37.46	8.56	<u>11.66</u>	7.49	40.34	<u>17.11</u>	7.49
Contriever (split)	37.64	<u>9.09</u>	10.43	<u>8.02</u>	39.47	16.04	<u>8.02</u>

Table 3. Results on running NLProofS pipeline using various retrieval methods. Best performance **bolded**, runner-up <u>underlined</u>.

Conclusions and Future Work

Study 1

`gpt-3.5-turbo` performs worse than the `T5` model for the prover task, and we suspect this is because it is not trained with enough examples with in-context-learning. Likely, the model was trained on tasks that were not similar enough to the EntailmentBank dataset for it to quickly generalize. However, the hybrid model does well because the addition of `gpt-3.5-turbo` can do no worse than the initial greedy tree generated by `T5`.

With these considerations, some future work includes:

1. **Fine Tuning:** OpenAI allows users to fine tune the `gpt-3` weights for a specific dataset, and we suspect this model would outperform our current results since it could see many more examples of EntailmentBank compared to in-context learning.
2. **GPT-4:** Once publically released, `gpt-4` has a context size four times larger than `gpt-3`, and the underlying model is more powerful. We suspect that these two advantages would allow `gpt-4` to outperform our current method.

Study 2

Overall, our embedding-based retrieval methods have weaker performance compared to Dalvi et al.'s retrieval approach, with roughly 10% – 15% worse values of Recall@25. However, this is likely a result, at least in part, of Dalvi et al.'s model being specifically trained on retrieval from WorldTree; our algorithm based on untailored embeddings, meanwhile, may miss required premises that bear little semantic resemblance to the hypothesis. This motivates item (1) of our future work.

We do observe slight performance differences between the five methods evaluated, and the split-hypothesis algorithm does appear to provide a small boost (particularly for `DRAGON-RoBERTa`, both in retrieval and NLProofS pipeline results), motivating item (3) of our future work.

Based on these conclusions, some avenues for future work include:

1. **Fine-tuning embeddings on EntailmentBank data**: Previous information retrieval work has noted the poor transferability of sentence embeddings across diverse tasks, which we believe leads to a key performance limitation in our use of pretrained embeddings. Thus, fine-tuning sentence embedding models by training specifically on EntailmentBank "relevant fact" data could boost our retrieval's recall performance.
2. **Integrating an iterative retrieval method into NLProofS** (inspired by Ribeiro et al. [4]): Rather than retrieving all the necessary premises up front, the model could leverage intermediate steps to retrieve other related premises, creating an iterative retrieval process embedded within proof generation.
3. **Devising a more sophisticated hypothesis splitting algorithm**: Currently, we split hypotheses in half simply by word count. Future work could find the optimal split location (perhaps a semantic split that most differentiates the meanings of each half) or explore splitting complex hypotheses in multiple locations.