Generating Natural Language Proofs with Verifier-Guided Search: Diverse Beam Search, Aggregation Functions, and Verifier-Weighting in NLProofS

Emre Onal
Princeton University (ORFE)

Max Gonzalez Saez-Diez
Princeton University (SPIA)

Maria Khartchenko
Princeton University (COS)

Abstract

Hallucination of invalid nodes in stepwise proof generation poses a problem for natural language processing. Yang et al. (2022) address this issue with NLProofS, mitigating hallucination by using an auxiliary verifier model to guide the stepwise proof generation towards generating valid proof steps. In this study, we replicate the baselines of their work and expand their exploration in three primary directions: (1) varying the prover-verifier proof score weighting in scoring nodes in the proof tree, (2) incorporating diverse beam search for proof tree generation, and (3) evaluating alternative functions for aggregating the scores of nodes in the proof tree. Our highest-performing model achieved an overall proof accuracy of 36.28% on the official Entailment Bank test dataset, therefore outperforming the (replicated) baseline score of 34.71% achieved by the original NLProofS model.

1 Introduction

Natural language proof generation is a difficult problem in NLP. This task requires a model to produce a proof tree utilizing a set of supporting sentences to arrive to an expected hypothesis. This is a particularly challenging task as models need to simultaneously generate relevant and valid proofs. Stepwise prover methods, compared to one-shot prover methods, are able to better generalize longer proofs but have had limited success on noisy real-world data. Models with stepwise provers have several limitations, such as hallucination of proof steps and the generation of irrelevant steps (Azamfirei et al., 2023). Yang et al. (2022) present a new method (NLProofS) that addresses both issues simultaneously by using a stepwise prover that generates proof steps in natural language, in conjunction with a verifier which scores each proof step's validity and mitigates hallucination of invalid steps (Yang et al., 2022).

As the first step of our research we replicate the estimates by Yang et al.. We obtain slightly higher scores ($\sim$1%) than the reported baseline results in the original paper despite using their pretrained model weights. Our references to "baseline results" throughout this paper refer to the test set values we obtained in our replication. Nonetheless, the original papers' baseline and our replications are comparable in magnitude. We expand upon their work by exploring several novel directions such as investigating prover-verifier score weighting, alternative proof tree node score aggregation functions, and alternative beam search methods. Our study performs ablation studies on these three expansion points, evaluating the resulting models' performance on EntailmentBank's (Dalvi et al., 2021) proof generation tasks.

2 Related Work

In the field of natural language proof generation, existing methods can be broadly categorized into two groups: single-shot methods, which generate the full proof in one step, and stepwise methods, which generate proofs iteratively. Examples of models with single-shot methods include the PRover model by Saha et al. (2020), which uses linear integer programming and jointly answers binary questions based on rules and then generates proofs for them, and the model proposed by Gontier et al. (2020) which uses text-to-text Transformer Language Models (TLM) to perform soft theorem-proving.

Stepwise methods in proof generation have also been explored by machine learning researchers over the past years. An example of a similar stepwise method to NLProofS is TeachMe, a teachable question-answering model that corrects its own errors (Mishra et al., 2022). Here, Mishra et al. also use a prover and verifier in conjunction, using the verifier to score the model's proof steps. However, unlike NLProofS, they select steps in the inference's proof search through greedy decisions. NLProofS (Yang et al., 2022) improves upon this by using beam search in their proof search algorithm which iteratively samples and explores proof trees from a large proof graph. At the date of its publication, this approach establishes a SOTA performance of 33% on task 2 of Entailmentbank, demonstrating the promising potential of step-wise proof generation.

3 EntailmentBank dataset

The EntailmentBank dataset was developed by Dalvi et al. in 2021 and consists of 1,840 entailment trees. These trees demonstrate how a question-answer (QA) pair is entailed from a small number of relevant sentences (or premises). Each entailment tree is built upon several fundamental principles. First, knowledge expressed by each new node reasonably follows from the content of its immediate children. Second, each step encodes a single inference that joins two facts. Third, complex conclusions can be drawn from simpler facts. Fourth and finally, each step in the tree is relevant to the QA pair of interest (Dalvi et al., 2021).

The EntailmentBank dataset is designed for three different tasks of increasing complexity. In Task 1 each QA pair only includes true premises, which at the same time are also relevant and necessary to form the hypothesis of interest. Task 2 increases the set of given premises to 25 herewith introducing so-called distractors, which are unrelated or unhelpful premises for the construction of the desired proof tree. In Task 3, a full-text corpus (WorldTree; see Xie et al. 2020) is provided and the model is required to identify and retrieve relevant supporting premises from the corpus for a given hypothesis.

4 Model

NLProofs is comprised of three core parts:

- **Stepwise Prover:** The prover consists of a fine-tuned pretrained T5 model which, given a set of premises, generates the next proof step as described in the proof search algorithm below.

- **Verifier:** The verifier consists of a fine-tuned pretrained RoBERTa model which assigns each proof step a "validity score" in $[0, 1]$. By assigning low scores to invalid steps, the verifier reduces the overall scores assigned to invalid or hallucinated steps generated by the prover.

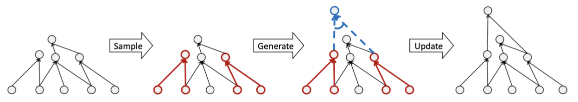


Figure 1: Proof Search Algorithm (Yang et al., 2022).

- **Proof Search Algorithm:** NLProofS differs from previous stepwise proof generation approaches in their proposed proof search algorithm (previous works utilized greedy methods for stepwise proof generation). NLProofS instead iteratively samples partial proofs from a large proof graph and generates several potential proof steps expanding these partial proofs, and updating the proof graph if a new proof step improves the score of a pre-existing node. The goal of this procedure is to iteratively explore a large space of possible proof trees to find a proof tree which maximizes the score of the hypothesis node. The proof graph is initialized with the greedy proof found by prover and the algorithm terminates once none of the generated proof steps in an iteration improve upon the score of any node in the proof graph, at which point the best proof of the hypothesis is extracted (Yang et al., 2022).

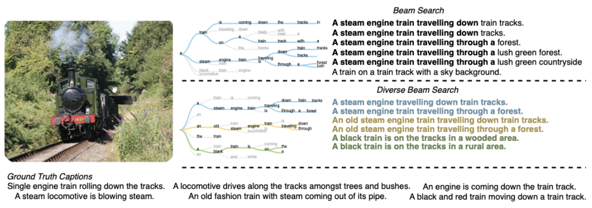
5 Model Ablation Study

We identify three major parts of expansion for the paper of interest:

1. **Prover-Verifier Weighting:** We evaluate a grid of verifier-weights (p) and corresponding prover-weights $(1 - p)$ for $p \in \{0, 0.1, 0.2, ...1\}$. This allows us to vary the relative influences of the prover and verifier models in the scoring of nodes in the proof tree.

2. **Diverse Beam Search:** We investigate diverse beam search as an alternative decoding algorithm. Concretely, we study the effect of simultaneously varying the number of beam groups and the diversity penalty.

3. **Aggregation Functions:** We investigate different functions for aggregating the scores of the nodes in the proof tree. Table 1 shows the aggregation functions we evaluate, all of



A steam engine train travelling down train tracks.
A steam engine train travelling down tracks.
A steam engine train travelling through a forest.
A steam engine train travelling through a lush green forest.
A steam engine train travelling through a lush green countryside
A train on a train track with a sky background.

A steam engine travelling down train tracks.
A steam engine train travelling through a forest.
An old steam engine train travelling down train tracks.
An old steam engine train travelling through a forest.
A black train is on the tracks in a wooded area.
A black train is on the tracks in a rural area.

Single engine train rolling down the tracks. A steam locomotive is blowing steam.
A locomotive drives along the tracks amongst trees and bushes. An old fashion train with steam coming out of its pipe.
An engine is coming down the train track. A black and red train moving down a train track.

Figure 2: Example of Diverse Beam Search (Vijayakumar et al., 2016)

which are designed to be monotonically non-increasing w.r.t. the step score and the scores of preceding nodes. This assumption reflects the intuition that our confidence in a node resulting from a proof step cannot exceed our confidence in the least reliable of the predecessor nodes or step score, which the proof step is reliant upon. NLProofS uses f_1 as its aggregation function.

Name	Function Definition
f_1	$\min_1(s, v_1, ..., v_n) := \min_1(I)$
f_2	$s \cdot \min_1(v_1, ..., v_n)$
f_3	$s^k \cdot \min_1(v_1, ..., v_n)$
f_4	$s \cdot v_1 \cdot v_2 \, ...$
f_5	$\min_1(I) \cdot \min_2(I)$
f_6	$\min_1(I) \cdot \max_1(I)$
f_7	$\min_1(I) \cdot \min_2(I) \cdot \max_1(I) \cdot \max_2(I)$
f_8	$\min_1(I)^{2-s}$
f_9	$\min(\alpha \cdot s, v_1, ..., v_n)$

Table 1: Evaluated Aggregation Functions. $\min_j$ ($\max_j$) refers to the value of the j^{th} smallest (largest) argument. s refers to the step-score, and $v_1, ...v_n$ are the scores of the predecessor nodes in the proof step.

6 Methodology

We evaluate our modifications to NLProofS on proof generation using EntailmentBank task 2. We use task 2 for our exploration as the presence of distractor premises increases the importance of the verifier model. Furthermore, task 2 is the setting in which NLProofS demonstrates the greatest performance improvements over alternative proof generation models. As such, we explore the effects of our modifications (prover-verifier weighting, diverse beam search and alternative aggregation functions) using EntailmentBank task 2 to evaluate performance. It is important to note that none of our modifications require retraining the NLProofS model and we therefore leverage the pre-existing prover and verifier model checkpoints provided in the official NLProofS repository.

We, at first, ran several exploratory rounds of experiments, some of which are included in Appendix A. These initial experiments enabled us to determine several feasible and promising directions to explore. Based on these initial explorations, we set up a number of different experiments of interest. Each of these experiments was run three different times. We evaluated the resulting performances on the validation dataset using the official EntailmentBank evaluation code. [1] The results for these validation experiments were documented on our poster.

Based on our results on the validation data, we selected our most promising hyperparameter combinations and evaluated their performances on the test data.[2]

Once again, we repeated each experiment thrice. For each evaluation metric, we report the means and standard deviations of the three runs for our experiments.[3]

7 Results

7.1 Verifier Weighting Experiments

Yang et al. (2022) set the weighting of the verifier to 0.5. They perform experiments with the verifier weight set to 0 and 1, but do not explore the effect of varying this weight incrementally. We vary the verifier weight from [0,1] in increments of 0.1. A verifier weight of 0 means that the prover score uniquely determines a node's score in the tree, whereas a verifier weight of 1 means the converse. A verifier weight of 0.5 means that the prover and verifier scores have an equal influence on the score of a node. Table 2 shows the overall proof accuracy on the test set for different verifier weights.

We obtain the best results for a verifier weight of 0.7 meaning that the verifier is given slightly more influence on the step score than the prover. This results in a test set accuracy of 35.78% which is $\sim$1% more than the result we obtain using the baseline weighting of 0.5 (34.71%).

[1] https://github.com/allenai/entailment_bank

[2] Note that in this paper we only present the final results on the test set. The Appendix includes some results from the validation testing.

[3] Note that the original paper runs each experiment 5 times. Due to limits in our computing resources, we were forced to reduce the number to 3 runs each.

verifier weight	proof accuracy (%)
0.8	35.588 ± 0.588
0.7	**35.784** ± **0.612**
0.5 (baseline)	34.706 ± 0.294
0.3	34.706 ± 0.294
0.2	34.804 ± 0.340

Table 2: Overall Proof Accuracy on Test Set for Different Verifier Weights

We plot these results in Figure 3 (in all bar plots the error bars correspond to one standard deviation). Verifier weights of 0.7 and 0.8 clearly outperform the baseline results. Furthermore, as can be seen, the difference is substantial enough that the standard error bars do not overlap. In contrast, we do not observe any meaningful changes with respect to the baseline when we set the verifier weight to be 0.2 or 0.3.

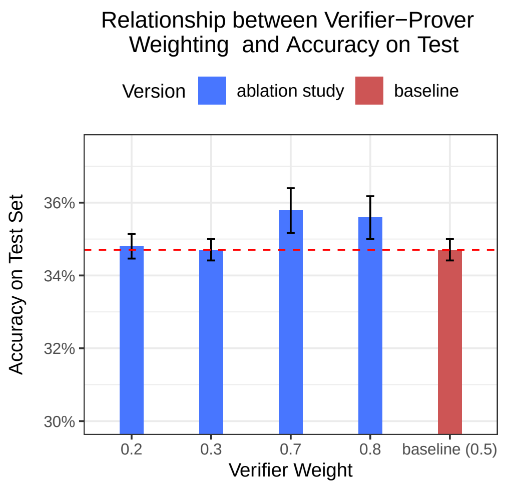


Figure 3: Overall Proof Accuracy on Test Set for Different Verifier Weights

Overall, these experiments imply that we can improve upon the performance of the original algorithm by tuning the balance between the prover and the verifier scores. Intuitively, the optimal value for this weighting should vary between dataset and task (e.g. in a setting without any distractor leaves, we might expect the verifier model to be less important for scoring nodes).

7.2 Diverse Beam Search

Diverse Beam Search is a diversity-promoting version of beam search. Its goal is to produce more diverse sequences than traditional beam search while maintaining a comparable run-time and memory usage (Vijayakumar et al., 2016). We introduce this method to expand on the baseline model's traditional beam search algorithm. We retain the same number of beams as in Yang et al. (2022), namely 10 beams, and simultaneously vary the number of beam groups (BG) and the diversity penalty (DP). In our tests on the validation data we test different numbers of BG (2, 5, 10) and different diversity penalties (10.0, 5.0, 2.0, 0.8, 0.2, 0, -0.2, -0.8, -2.0, -5.0, -10.0). We selected only the ones that outperform the baseline on the validation data for further analysis. Table 3 shows the overall proof accuracy on the test set for diverse beam search experiments using our best hyperparameters.

BG	DP	proof accuracy (in %)
2	−10.0	35.882 ± 0.294
2	−5.0	**36.275** ± **0.340**
2	−2.0	36.176 ± 0.294
2	−0.8	35.784 ± 0.170
2	−0.5	36.078 ± 0.340
2	−0.2	36.078 ± 0.170
2	0.8	33.824 ± 0.588
baseline (1 BG, 0 penalty)		34.706 ± 0.294

Table 3: Overall Proof Accuracy on Test Set for Different Diverse Beam Search Settings

Our initial experiments on the validation dataset show that the algorithm performs particularly well when a negative instead of a positive diversity penalty is applied. This is somewhat surprising. Usually, researchers implement this method with positive diversity penalties in order to force the algorithm to consider a more diverse set of sequences (see Figure 2). We hypothesize that in this case, using a positive penalty might lead the algorithm to consider more "diverse" but hallucinated steps. It would an interesting future exploration to investigate diverse beam search combined with increased verifier-weightings. The interdependence between NLProofS and diverse beam search remains somewhat unclear. However, the increased performance in our diverse beam search setting demonstrates that this could be a promising direction for future research.

We find that our best performing setting uses a diversity penalty of −5.0. With this choice of hyperparameter, we achieve a final accuracy of 36.28% on the test set, herewith outperforming the baseline by over 1.5% and proving that diverse beam search can be a useful and viable way to improve performance.

7.3 Aggregation Functions

Yang et al. (2022) use f_1 (see Table 1) as the aggregation function during the inference step. We expand on that work by proposing and testing eight other score aggregation functions which are all monotonically non-increasing with respect to the step score and the scores of its children (as in (Yang et al., 2022)). We ran the aggregation functions that performed the best on the validation set on the test set and concluded that function f_3 with $k = 3$ (the cubed step score multiplied by the minimum of the leaves' scores) achieved the best results: a proof accuracy of 35.17%. However, as can be seen in table 4, all functions score roughly the same (with $f_3(k = 3)$ marginally outperforming the others albeit with a larger standard deviation). Our proposed alternative aggregation functions do not appear to significantly improve upon the baseline method's accuracy.

f_*	function	proof accuracy (in %)
f_1	baseline	34.706 ± 0.294
f_3	$s^2 \cdot \min(v_1, ..., v_n)$	35.000 ± 0.588
f_3	$s^3 \cdot \min(v_1, ..., v_n)$	**35.196 ± 0.679**
f_3	$s^4 \cdot \min(v_1, ..., v_n)$	34.902 ± 0.170
f_5	$\min_1(I) \cdot \min_2(I)$	$\underline{35.098 \pm 0.449}$
f_8	$\min(v_1, ..., v_n)^{2-s}$	35.000 ± 0.588

Table 4: Overall Proof Accuracy on Test Set for Different Aggregation Functions

8 Additional Remarks

In the following section, we want to shed some light on an interesting dynamic that we observe in the diverse beam search results. This is particularly interesting as the experiments we ran using this decoding algorithm allowed us to beat the baseline by the largest margin. Concretely, Figure 4 depicts the interaction between diversity penalty and leaf accuracy obtained on the test set.

It is interesting to see that, at least for the point estimate, we observe a clear decrease in the leaf accuracy when comparing the baseline to our ablation experiments. For this metric, the baseline estimate appears to be better than all our tested ablations. However, when accounting for the baseline's high standard deviation, it seems as though these leaf accuracies do not differ significantly.

In contrast to this, if we look at the proof-step accuracy, plotted in figure 5, we see clear differences between our ablations and the baseline. Concretely,

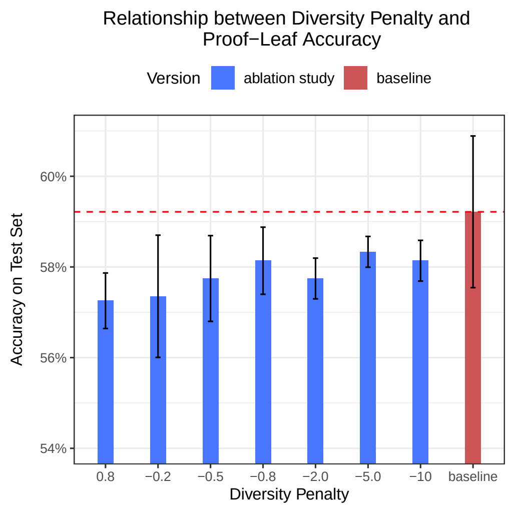


Figure 4: Overall Proof-Leaf Accuracy on Test Set for Different Diversity Penalties

we see how all our methods with negative diversity penalties seem to substantially outperform the baseline model by at least one or more percentage points.

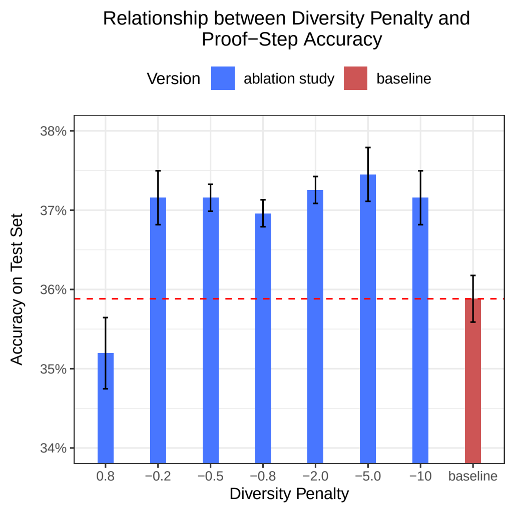


Figure 5: Overall Proof-Step Accuracy on Test Set for Different Diversity Penalties

These observations tell one main story: applying a negative diversity penalty seems to allow the algorithm to build better proofs given the same set of hypotheses. While we knew from the discussion before that the overall accuracy was better, it could have been the case that in our ablations we were somehow simply picking better leaves and were, for that reason, achieving a higher accuracy on the overall proof. However, Figures 4 and 5 show that this is not the case. These results are meaningful as they increase our confidence in the assertion that we were successful in improving upon the baseline quality of the proof-tree estimator presented by Yang et al. (2022).

9 Conclusion

In this paper, we first replicate the findings by Yang et al. (2022) and then present a number of different ablations to their baseline model. Concretely, we study how varying verifier-prover weightings and aggregation functions, and implementing diverse beam search can impact the overall proof accuracy. We explore several alternatives to the aggregation function implemented in the original work. However, these modifications do not seem to yield any significant improvement in performance over the baseline.

In contrast, several of our additions do appear to improve upon the NLProofS baseline model. First, setting the verifier score to 0.7, therefore giving it slightly more influence than the prover when calculating the stepscore, resulted in a proof accuracy of 35.78%. Second and most notably, our implementation of diverse beam search with 2 beam groups and a diversity penalty of −5.0 achieved a proof accuracy of 36.28%. This result reveals an interesting direction for future work investigating diverse beam search for stepwise proof generation and the underlying mechanism connecting the diversity penalty with the overall proof accuracy. In addition, there is room to explore other variations of beam search such as Determinantal Beam Search (Meister et al., 2021).

Acknowledgements

We would like to thank Professor Danqi Chen for her guidance and advice on our project and more generally for providing us an amazing introduction to the field of natural language processing.

Code

All of our code for this project can be found at: `https://github.com/mashakhart/NLP-final-project`, which was forked from NLProofS at `https://github.com/princeton-nlp/NLProofS`

References

Razvan Azamfirei, Sapna R. Kudchadkar, and James Fackler. 2023. Large language models and the perils of their hallucinations. volume 27. BioMed Central Ltd. Funding Information: ChatGPT was used in the writing of this manuscript. All content generated by ChatGPT is marked as such.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nicolas Gontier, Koustuv Sinha, Siva Reddy, and Chris Pal. 2020. Advances in neural information processing systems. volume 33, pages 22231–22242. Curran Associates, Inc.

Clara Meister, Martina Forster, and Ryan Cotterell. 2021. Determinantal beam search. volume abs/2106.07400.

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

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. pages 122–136, Online. Association for Computational Linguistics.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

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 *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5456–5473, Marseille, France. European Language Resources Association.

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

A Appendix

A.1 Diverse Beam Search Exploration

In the initial exploration part, we found that setting the diversity penalty to -100 provided the best results. That is the reason why it is fixed here while other variables are being varied. Later analysis showed no difference between a diversity penalty of -10 and smaller diversity penalties, which is why penalties below -10 were not further explored. Also, note that the values here are substantially larger than the reported accuracy scores on the test set. This is due to two reasons: first, these results are validation accuracy results, and second, these results are not evaluated with the "official" Entailment Bank evaluation code.

Num. Beams (topk)	BG	div. pen	Proof Acc. (%)
Basic model as used in paper with beam search			40.64
10 (10)	2	5.0	40.64
10 (10)	2	100	39.57
10 (10)	5	5.0	37.43
20 (20)	5	2.0	38.50
10 (10)	2	-1.0	40.64
10 (10)	2	-10	41.17
10 (10)	2	-1000	41.17
10 (10)	2	-200	41.17
10 (10)	2	-125	41.17
10 (10)	2	-105	41.17
10 (10)	2	-75	41.71
10 (10)	2	-90	41.17
10 (10)	2	-110	41.71
10 (10)	5	-100	40.64
10 (10)	10	-100	38.50
20 (20)	2	-100	41.17
20 (20)	5	-100	41.71
20 (20)	10	-100	40.64
20 (1)	2	-100	39.03
20 (2)	2	-100	39.03
20 (5)	2	-100	40.64
20 (10)	2	-100	40.64
10 (10)	1	-100	41.17
2 (2)	1	-100	40.64
5 (5)	1	-100	41.71
12 (12)	3	-100	41.71
12 (12)	4	-100	42.24
12 (12)	6	-100	40.64

Table 5: Diverse Beam Search Exploration. BG is the number of beam groups.

A.2 Max. Number of Steps

In Table 6, we set the number of beam groups to 2. The diversity penalty is kept constant as well. These two columns are dropped for clarity purposes.

A.3 Beam Search Exploration

We experimented with the number of beams for the traditional beam search algorithm. However, contrary to what we expected, increasing the number of beams did not increase overall proof accuracy. This avenue was not explored further. Even though 5 and 10 beams gave the same result, we kept the number of beams to 10 (as in the original paper) for comparison purposes.

Num. beams (topk)	Max Num. Steps	Proof Acc. (%)
10 (10)	20 (baseline)	0.4224
10 (10)	30	42.24
10 (10)	50	41.71
10 (10)	10	40.64
10 (10)	15	42.24

Table 6: Max. Number of Steps Exploration

num beams	Proof Acc. (%)
5	41.17
10	41.17
15	40.10
30	39.57

Table 7: Beam Search Exploration

A.4 Initial Verifier Weighting Exploration

weight	Proof Acc. (%)
0	40.10
0.1	40.10
0.2	40.64
0.3	41.17
0.4	40.64
0.5	40.64
0.6	40.64
0.7	41.17
0.8	40.64
0.9	39.57

Table 8: Verifier Weighting Exploration

		Leaves		Steps		Intermediates		Overall
category	version	F1	AllCorrect	F1	AllCorrect	F1	AllCorrect	AllCorrect
baseline		90.226 ± 0.139	59.216 ± 1.672	48.915 ± 0.310	35.882 ± 0.294	39.706 ± 0.294	70.600 ± 0.300	34.706 ± 0.294
aggregation	f_8	89.924 ± 0.372	58.725 ± 1.451	49.128 ± 0.359	36.569 ± 0.612	39.118 ± 0.588	69.918 ± 0.227	35.000 ± 0.588
	f_5	89.587 ± 0.381	56.471 ± 0.294	48.888 ± 0.868	36.863 ± 0.449	39.118 ± 0.509	70.311 ± 0.352	35.098 ± 0.449
	$f_{3,\ k=2}$	89.607 ± 0.107	57.647 ± 0.778	48.724 ± 0.400	36.373 ± 0.340	39.118 ± 0.588	70.037 ± 0.498	35.000 ± 0.588
	$f_{3,\ k=3}$	89.742 ± 0.430	57.843 ± 1.358	48.695 ± 0.429	36.765 ± 0.509	39.314 ± 0.612	70.286 ± 0.162	35.196 ± 0.679
	$f_{3,\ k=4}$	89.732 ± 0.198	57.157 ± 0.449	48.589 ± 0.303	36.667 ± 0.170	39.118 ± 0.000	70.303 ± 0.490	34.902 ± 0.170
diverse beam search ($BG = 2$)	$p = -0.2$	89.751 ± 0.119	57.353 ± 1.348	50.172 ± 0.266	37.157 ± 0.340	**40.000 ± 0.000**	**70.909 ± 0.271**	36.078 ± 0.170
	$p = -0.5$	89.705 ± 0.105	57.745 ± 0.945	49.781 ± 0.198	37.157 ± 0.170	39.902 ± 0.449	70.324 ± 0.100	36.078 ± 0.340
	$p = -0.8$	89.804 ± 0.102	58.137 ± 0.740	49.637 ± 0.178	36.961 ± 0.170	39.706 ± 0.000	$\underline{70.766 \pm 0.073}$	35.784 ± 0.170
	$p = -10.0$	89.861 ± 0.154	58.137 ± 0.449	$\underline{50.147 \pm 0.301}$	37.157 ± 0.340	39.804 ± 0.340	70.605 ± 0.035	35.882 ± 0.294
	$p = -2.0$	89.656 ± 0.112	57.745 ± 0.449	49.464 ± 0.472	$\underline{37.255 \pm 0.170}$	39.804 ± 0.340	70.703 ± 0.456	$\underline{36.176 \pm 0.294}$
	$p = -5.0$	89.837 ± 0.230	58.333 ± 0.340	**50.256 ± 0.247**	**37.451 ± 0.340**	$\underline{39.902 \pm 0.170}$	70.696 ± 0.196	**36.275 ± 0.340**
	$p = 0.8$	90.059 ± 0.283	57.255 ± 0.612	48.330 ± 0.426	35.196 ± 0.449	38.039 ± 0.612	70.199 ± 0.093	33.824 ± 0.588
verifier	0.2	90.143 ± 0.196	58.039 ± 1.189	48.565 ± 0.535	35.588 ± 0.778	39.706 ± 0.509	70.703 ± 0.278	34.804 ± 0.340
	0.3	$\underline{90.317 \pm 0.300}$	$\underline{59.412 \pm 0.778}$	49.238 ± 0.409	35.784 ± 0.340	39.118 ± 0.294	70.701 ± 0.219	34.706 ± 0.294
	0.7	**90.340 ± 0.160**	**60.196 ± 0.449**	49.213 ± 0.036	36.961 ± 0.340	39.706 ± 0.294	70.454 ± 0.194	35.784 ± 0.612
	0.8	90.091 ± 0.318	58.627 ± 1.797	48.898 ± 0.302	36.471 ± 0.778	39.706 ± 0.294	70.434 ± 0.117	35.588 ± 0.588

Table 9: Summary of all Results on Test Set