[V1][Spec Decode] EAGLE Tree Attention Design Doc

[Track]

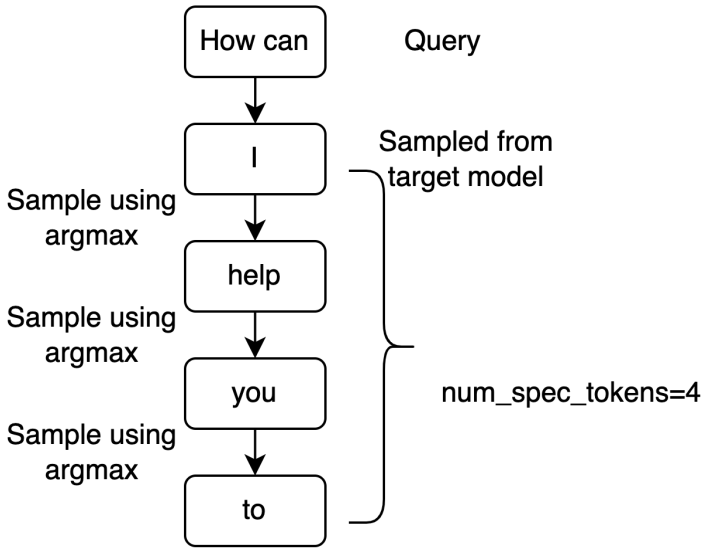
- ☑ ~~Construct tree structure~~
- ☑ ~~Select top-k instead of top-1 from the logits~~
    - ☑ ~~Level-0 and level-1~~
    - ☑ ~~Level-2 & 2+~~
- ☑ ~~Node/path selection after expansion~~
- ☐ Attention metadata & attention mask
    - ☐ Together with some ROPE embedding stuff
- ☐ Rejection logic
- ☐ KV cache & CUDA graph (future PRs)
- ☐ E2e and unit tests

Known issues:

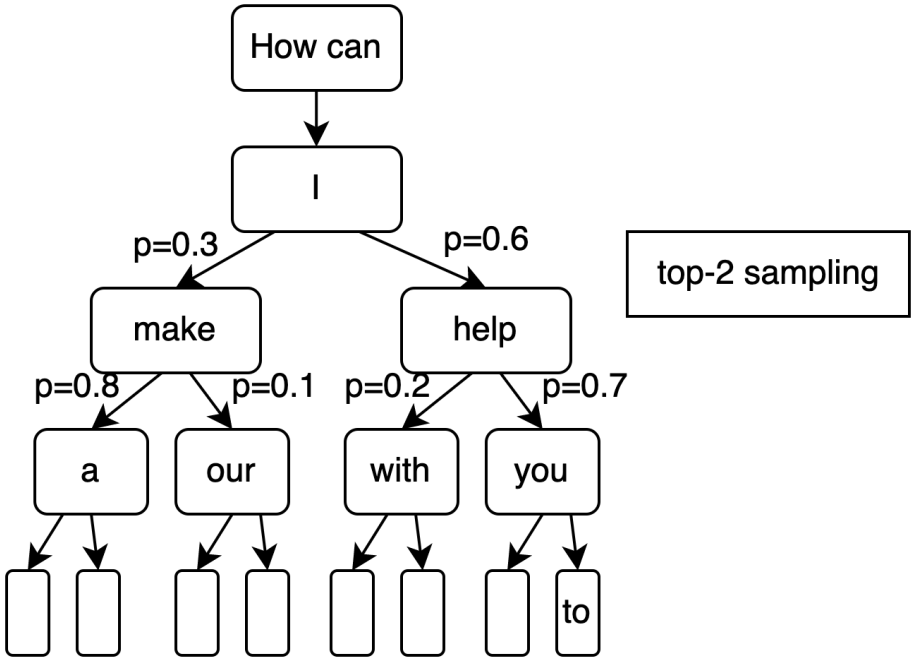
1. Attention metadata
2. KV cache
3. query_start_loc needs to be correctly set (needs double-check)

Background: from chain-draft to tree-draft

Chain-draft: (Current implementation)
3 forward pass
Propose 4 tokens



Tree-draft (static):
3 forward pass
Propose 2^4-1 = 15 tokens



Tree-draft (dynamic): - Focus of Implementation

Expand: Only expand the nodes in the last layer within the top-2 *global acceptance prob*

Stop condition: unclear in the paper.

Rerank: select nodes within the top-8 *global acceptance prob*

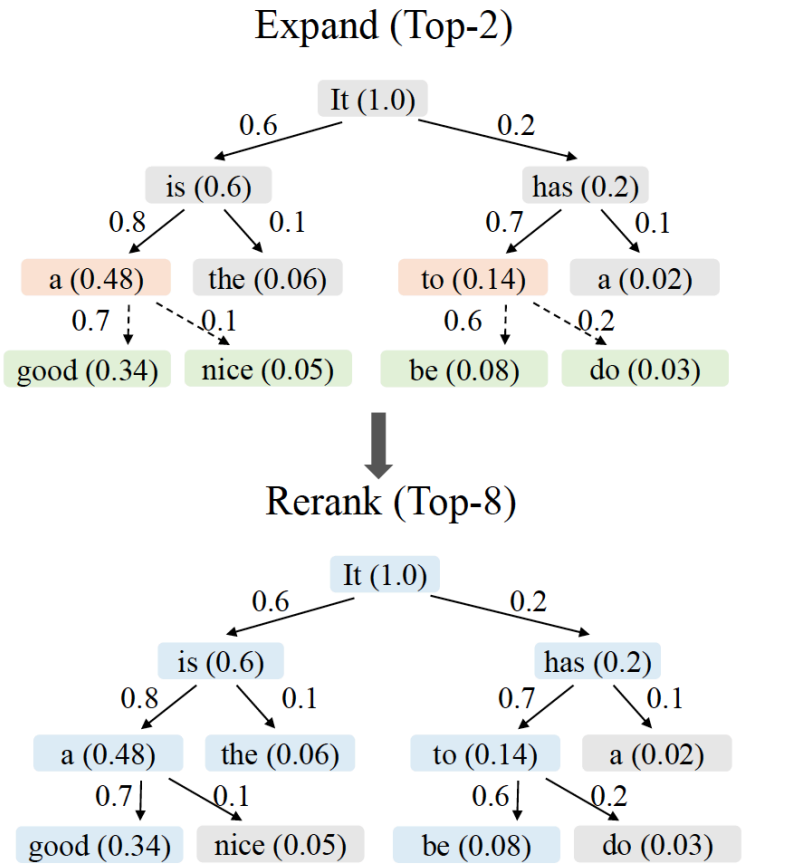
Token selection: If the total number of nodes is larger than 8 (num_spec_tokens), we get the top-8; If not, padding the rest.

Tree Flattening: BFS traversal to create sequence for verification

global acceptance prob: product of probability along the path to root

Note: the *prob* is not normalized among siblings

Attention mask: nodes can only see their ancestors



Flatten to 1D

It	is	has	a	the	to	good	be

Attention mask

	It	is	has	a	the	to	good	be
It	✓							
is	✓	✓						
has	✓		✓					
a	✓	✓		✓				
the	✓	✓			✓			
to	✓		✓			✓		
good	✓	✓		✓			✓	
be	✓		✓			✓		✓

Things to discuss: (please feel free to add more…)

1. Based on their evaluations, dynamic tree seems to outperform static one in all tested scenarios, so should we skip the static one and go directly to the dynamic tree? (If not, we may need an “on/off” switch to enable/disable the tree.)
2. (Outdated) Previously, ‘num_spec_tokens’==’num_forward_pass’, now we may need another config ‘num_spec_forward_pass’/’num_spec_tree_depth’, and ‘num_spec_tokens’ is the number after re-ranking
    a. Followup: we should compare with chain-draft by keeping ‘num_spec_tokens’ the same, or ’num_forward_pass’?
3. More space/KV cache may need to be allocated (maybe can solve later)
4. In the stopping condition, we should stop when either “num_spec_tokens” or ‘num_spec_forward_pass’ is reached.
5. (Outdated) Top-2 sampling means each node may have 0 or 2 children. Do we hard code it as 2 as the number of tokens will grow exponentially with this number? Top-2 expanding means to select at most 2 nodes in the last layer to expand, we may set it as a tunable configuration.

Core Modification:

- eagle.py:
    - Extra data structure to maintain the tree information (only alive inside propose(), and will be reflected together with the output - draft_token_ids)
    - Looping condition
    - Algorithm to select leaves (top-k)
    - Apply attention mask when draft model forwarding
- GPUModelRunner.py:
    - Need to prepare the tree structure for verification
    - Modified rejection sampler: need to reflect to the attention mask
- SpeculativeConfig:
    - Extra configuration

- Testing
    - E2E: both throughput and acceptance length should be increased

A. Implementation Details

Vanilla: We use models from the Huggingface.transformers library with the PyTorch backend and pre-allocated KV cache. Other methods also use these models as their base.

(Standard) Speculative Sampling: We use the assisted generation feature from the HuggingFace Transformers library.

PLD, Lookahead, Medusa, and Hydra: We use the default settings and the officially released weights.

EAGLE: Vicuna and LLaMA2-Chat draft models use the officially released weights, while LLaMA3-Instruct is trained using the ShareGPT dataset (consistent with Medusa and Hydra).

EAGLE-2: For the 7B (8B), 13B, and 70B original LLMs, we set the total number of draft tokens to 60, 50, and 48, respectively, with a draft tree depth of 6, and select 10 nodes during the expansion phase.

How many branch-out children each node has? Unknown.

It cannot be 2, because 1+2+4+8+10+10+10=45<60

We take it as the same as num_spec_expand for now.

Config

use_tree_draft: bool # Whether to use tree-draft (True) or chain-draft (False)
num_spec_tokens: int # Total number of tokens to propose (existing parameter)

Tree-draft specific
spec_tree_depth: int # Number of forward passes (tree depth)
num_spec_expand: int # Number of nodes to expand during each expansion phase

Suppose num_spec_expand = 10
Depth = 0, (root) n = 1
Depth = 1, n=10
Depth = 2, n=100 —-> only 10 out of 100 will be selected to expand
Depth = 3, n=100
Depth = 4, n=100

If num_spec_tokens==1, early exit

If spec_tree_depth==0, but num_spec_tokens>1, or other cases when the num_spec_tokens cannot be filled up, we need to pad them.

E.g.
Target_token_ids: [a1, b1, b2, c1, c2, c3]
Self.input_ids: [a2, b2, b3, c2, c3, c4]

Then the roos will be: a2, b3, c4

First forward pass: (suppose we only select top-2)

a2 —-> a3_1, a3_2
b3 —-> b4_1, b4_2
c4 —-> c5_1, c5_2

Pseudocode:

1. Construct the tree structure for each request in the batch
    Construct the root nodes
2. First forward pass (it happens in the propose(), but we handle the node appending in tree_draft_propose())
    Select top-k highest tokens
    Append them to the corresponding tree
3. Following forward pass until spec_tree_depth

References:
EAGLE: https://arxiv.org/pdf/2401.15077
EAGLE-2: https://arxiv.org/pdf/2406.16858
EAGLE & EAGLE-2 Video: https://www.youtube.com/watch?v=oXRSorx-Llg