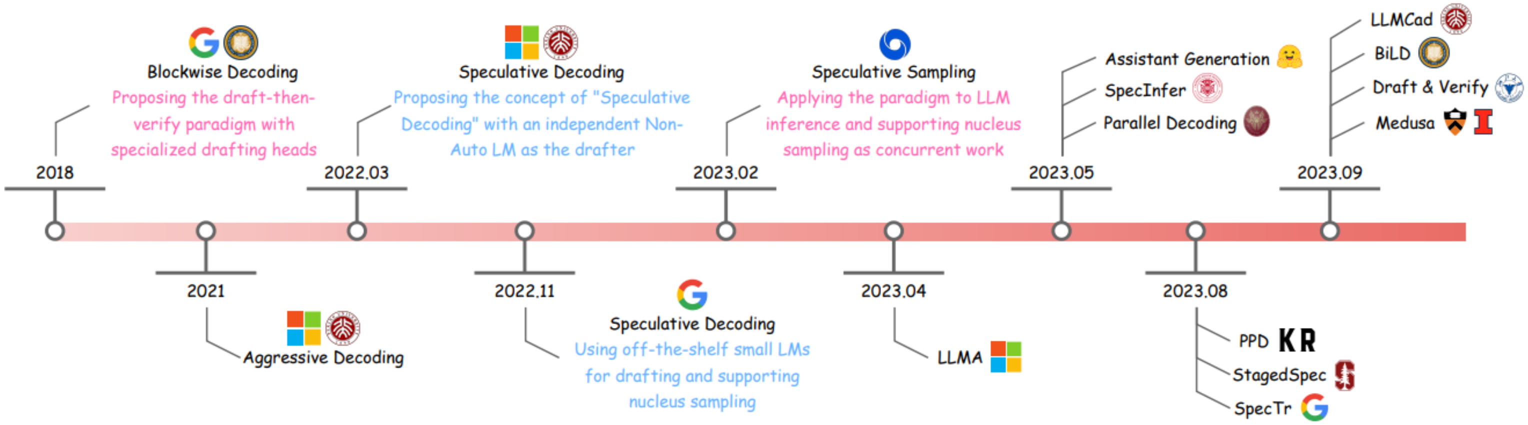
# Speculative Decoding of LLM: CTC-drafter

# Speculative Decoding with CTC-based Draft Model for LLM Inference Acceleration

**Zhuofan Wen**[1,4]**,Shangtong Gui**[1,2,4]**,Yang Feng**[1,3,4*]
[1]Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
[2]State Key Lab of Processors,
Institute of Computing Technology, Chinese Academy of Sciences
[3]Key Laboratory of AI Safety, Chinese Academy of Sciences
[4]University of Chinese Academy of Sciences, Beijing, China
{wenzhuofan24z,guishangtong21s,fengyang}@ict.ac.cn

# Speculative Decoding of LLM

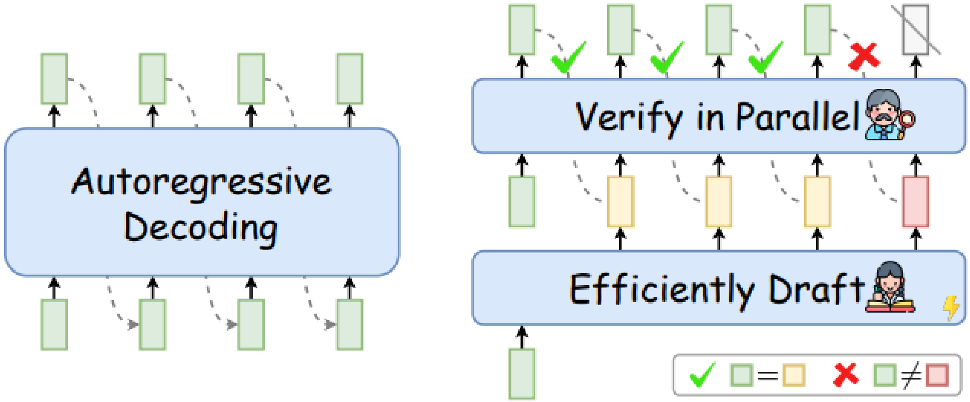
- Large language models(LLM) possess many advantages over traditional language models.
- However, LLM also faces disadvantages such as slower inference speed and higher training difficulty due to the larger number of parameters.
- Improving from the perspective of decoding strategies: speculative decoding.




The development chart of speculative decoding from： Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding

# Basic Paradigm of Speculative Decoding

- Compared to autoregressive decoding, speculative decoding utilizes auxiliary models to achieve multi-token decoding in one inference step.




- **The autoregressive decoding process(left):**

  Only one token is decoded per step

- **The speculative decoding process(right) :**

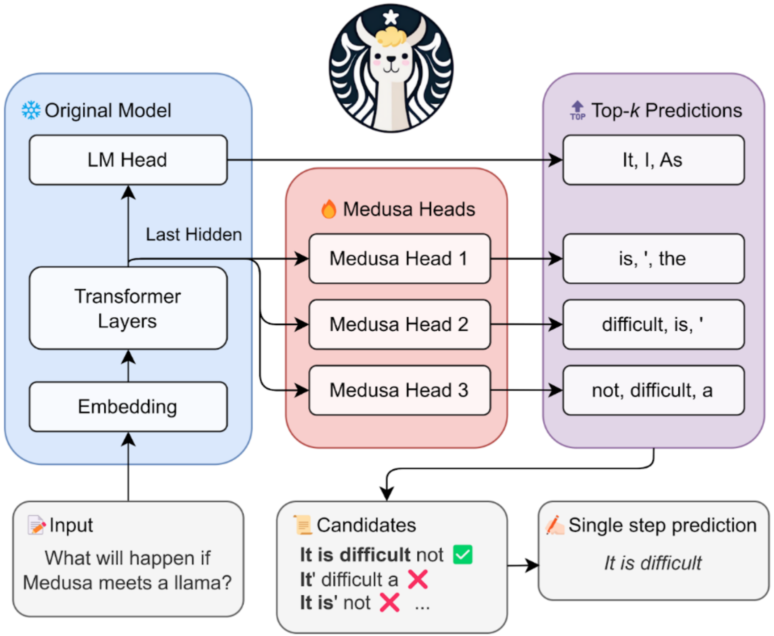
  Smaller model is used to predict subsequent tokens in advance (Efficiently Draft). Draft tokens are sent to the LLM for verification (Verify in Parallel).

  Multiple tokens are decoded per step

Figure from: Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding

# Medusa

- Existing speculative decoding works: take Medusa as an example




- **Draft**:

  Several independent linear layers as auxiliary model(Medusa head)

- **Verify**:

  Organizing candidate tokens as tree structure(token tree verification)

**Analysis**:

- Insufficient representation ability with linear layers.
- Prediction of candidate tokens at different positions is independent, without considering contextual information.
- Fixed method to combine candidate sequences lacks generality across different generation tasks.

Figure from: Medusa: Simple framework for accelerating llm generation with multiple decoding heads.

# CTC-drafter： More accurate candidate generation based on CTC


## CTC inference：

- Auxiliary model provides probability distribution at each position
- Combine and generate multiple candidate sequences.
- Remove blank ε and duplicate tokens(CTC blank-collapse) → final sequence.

## CTC loss function：

- Given input $X$, calculate the probability of a specific final sequence $Y$:

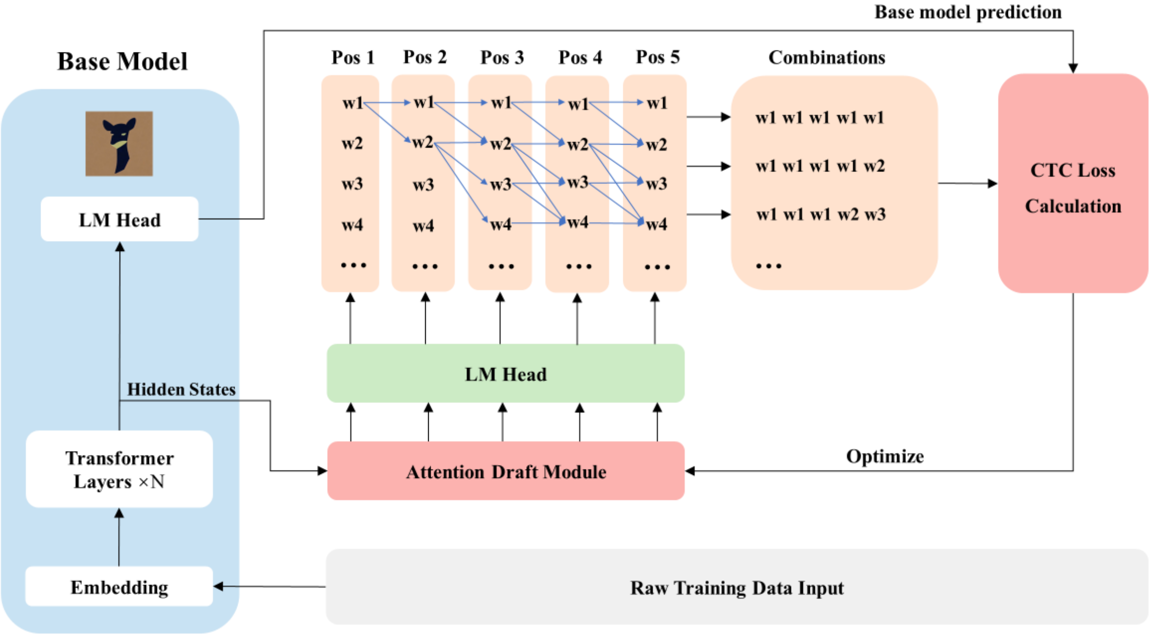
  Sum probabilities of all paths $A$ that produces sequence $Y$ after CTC-collapse：

  $P(Y|X) = \sum_{A \in A_{X,Y}} P(A)$
- The product of each tokens' probabilities on the path: $P(A) = \prod_{t=1}^{T} p_t(a_t|X)$ ,

  $A = (a_1, a_2, a_3 \ldots \ldots \ldots a_t, a_{t+1} \ldots \ldots a_n)$
- The final training objective： $max \sum_{(X,Y) \in D} P(Y|X)$

Figure from： Hannun, "Sequence Modeling with CTC", Distill, 2017

# CTC-drafter： More accurate candidate generation based on CTC




**Training**

- **Attention Draft Module**

  Replace linear layers with transformer layers.

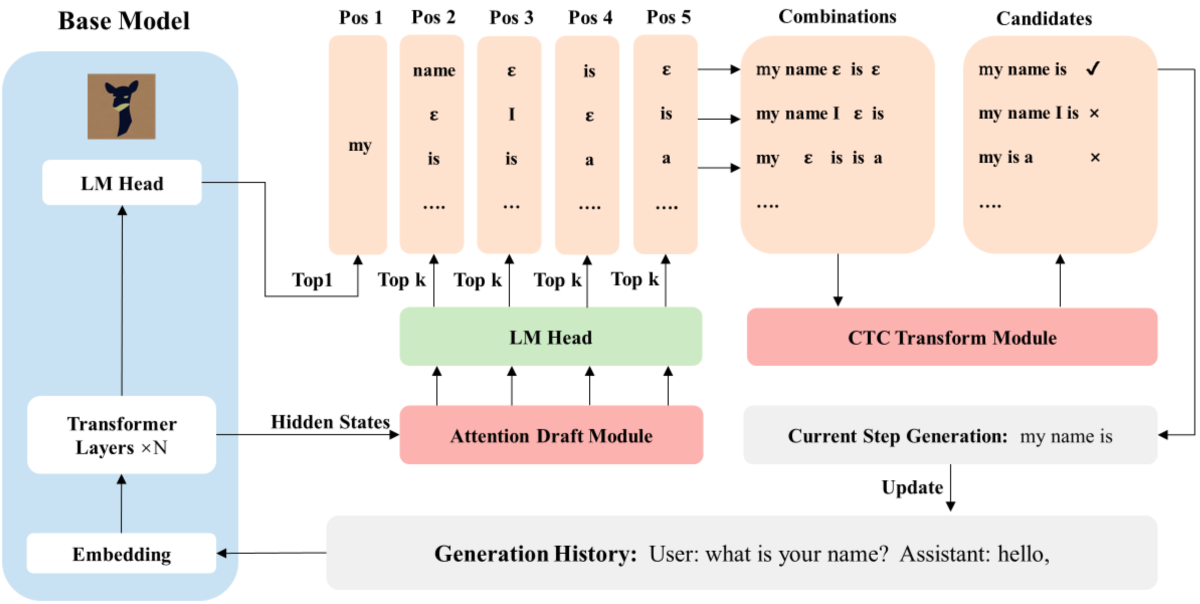
  → better align with the base model.

- **CTC Loss**：

  Replace Cross-Entropy Loss with CTC Loss

  → Traverse all possible sequences and enhance contextual information based on dynamic programing.

# CTC-drafter： More accurate candidate generation based on CTC




**Inference**

- **Combinations**:

  ① LLM generate the first token. Select the top k tokens of highest probability for following positions.

  ② Organize these tokens as tree structure described in [1].

- **CTC Transform Module**:

  ① Apply the blank-collapse as described in [2] and obtain Candidates.

  ② Sent to LLM for validation.


[1] Miao X, Oliaro G, Zhang Z, et al. SpecInfer: Acceerating Generative Large Language Model Serving with Tree-based Speculative Inference and Verification.

[2] Graves A, Fernández S, Gomez F, et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks

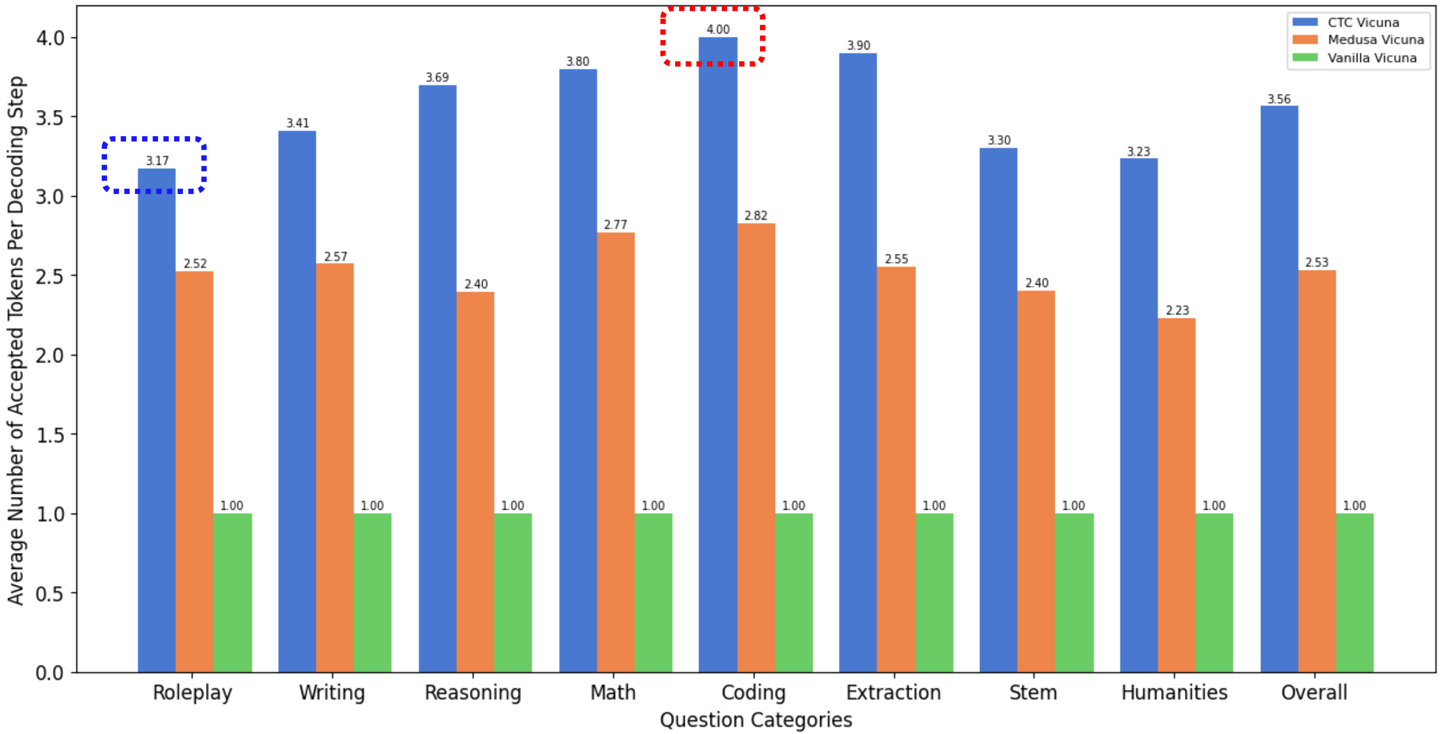
# CTC-drafter： More accurate candidate generation based on CTC

| Speculation Method | Vicuna-7B | | Vicuna-13B | | Vicuna-33B | |
|---|---|---|---|---|---|---|
| | $\gamma$ | $\beta$ | $\gamma$ | $\beta$ | $\gamma$ | $\beta$ |
| MT-bench | | | | | | |
| Vanilla(Chiang et al. 2023) | 1.00× | 1.00 | 1.00× | 1.00 | 1.00× | 1.00 |
| Medusa(Cai et al. 2023) | 2.13× | 2.58 | 1.97× | 2.60 | 1.93× | 2.55 |
| Hydra(Ankner et al. 2024) | 2.36× | 3.04 | 2.17× | 3.06 | 2.15× | 2.95 |
| CTC-drafter | **2.78×** | **3.56** | **2.52×** | **3.51** | **2.20×** | **3.53** |
| GSM8K | | | | | | |
| Vanilla(Chiang et al. 2023) | 1.00× | 1.00 | 1.00× | 1.00 | 1.00× | 1.00 |
| Medusa(Cai et al. 2023) | 2.33× | 2.78 | 2.21× | 2.68 | 2.10× | 2.46 |
| CTC-drafter | **2.43×** | **3.53** | **2.66×** | **3.53** | **2.16×** | **3.40** |

- Choose Vicuna-7B, 13B, and 33B as the LLMs to be accelerated.
- $\gamma$: The average speedup ratio relative to vanilla method.
- $\beta$: The average number of accepted tokens per decoding step.
- MT-bench, GSM8K： Evaluation benchmark datasets.

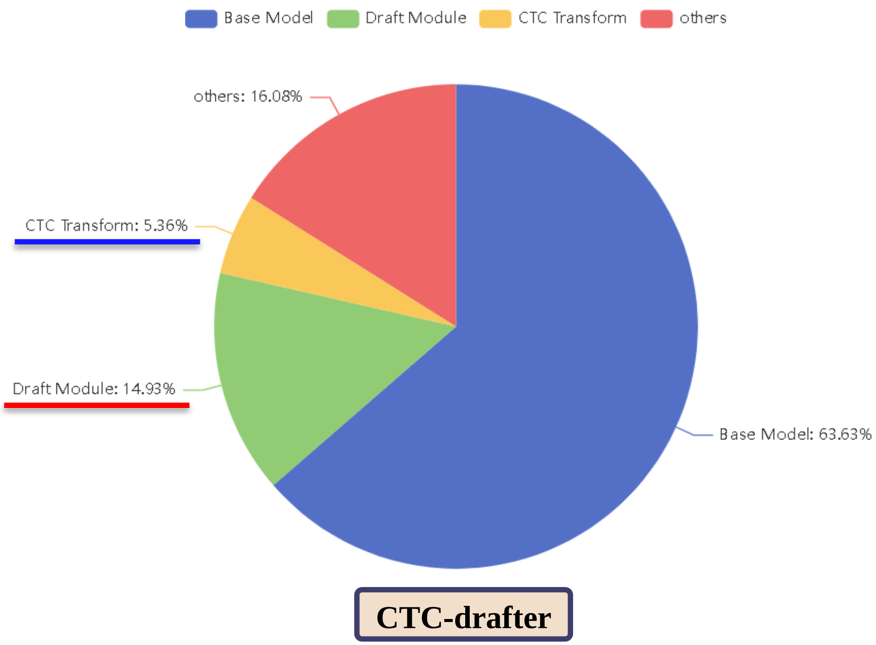
**CTC-drafter achieves higher speedup ratios ($\gamma$) by generating higher quality candidate tokens ($\beta$)**

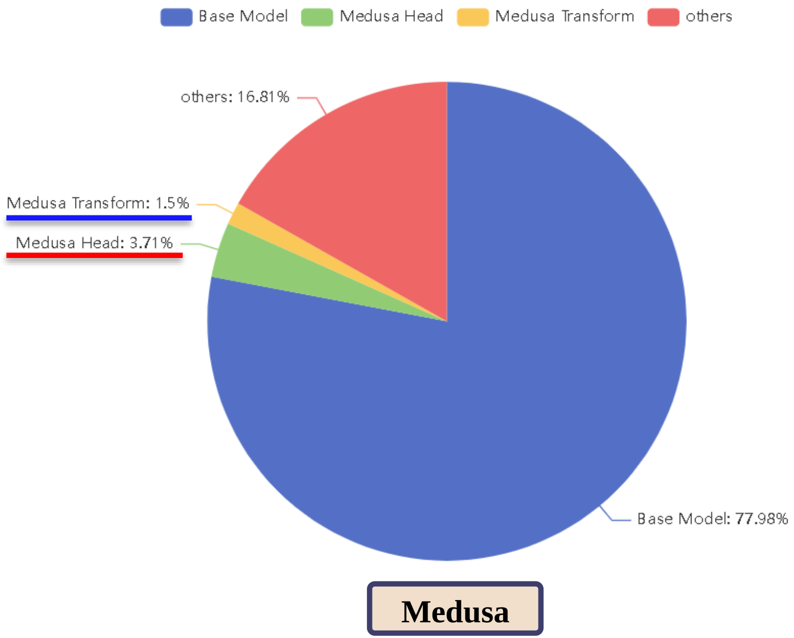
# CTC-drafter: More accurate candidate generation based on CTC




- **Extraction:**
  **CTC-Vicuna perform the best on coding category.**
- **Roleplay:**
  **CTC-Vicuna performs relatively poorly on role-playing category.**
  **Lack of questions on this category in the training dataset.**

# CTC-drafter: More accurate candidate generation based on CTC




- More complex auxiliary model structures are introduced, inevitably introducing additional draft latency.
- **Reducing the overall decoding steps of LLM → Still more significant inference acceleration.**