CS726: Programming Assignment 3

LSA
Asmith Reddy
22B0663

1 Task 0: Introduction to LLM Decoding Techniques

1.1 Greedy Decoding:

At every step, we simply pick the token with the highest probability from the LLM's output distribution. Formally, at the t^{th} step, you obtain the next token as follows:

$$y_t = \arg\max_w p(y \mid y_{1:t-1}, x)$$

where $y_{1:t-1}$ denotes previously generated tokens and x is the input prompt. This process is repeated iteratively until the end-of-sentence (EOS) token is generated.

BLEU	ROUGE-1	ROUGE-2	ROUGE-LCS
0.30	0.35	0.12	0.27

Table 1: Results

1.2 Random Sampling with Temperature Scaling:

Instead of always selecting the most probable token, here we randomly sample from the probability distribution while adjusting its sharpness using a temperature parameter τ. That is, first, we modify the probabilities as follows:

$$P'(w \mid y_{1:t-1}, x) = \frac{P(w \mid y_{1:t-1}, x)^{1/\tau}}{\sum_{w' \in V} P(w' \mid y_{1:t-1}, x)^{1/\tau}}$$

A token is then randomly sampled from P. Like before, keep repeating this process until the EOS token is generated. Here, you must experiment with $\tau \in \{0.5, 0.9\}$ and report your findings.

Temperature	BLEU	ROUGE-1	ROUGE-2	ROUGE-LCS
$\tau = 0.5$	0.28	0.29	0.11	0.238
$\tau = 0.9$	0.19	0.17	0.05	0.147

Table 2: Results

1.3 Top-k Sampling:

Rather than sampling from the entire vocabulary, in Top-k sampling, we restrict our choices to the k most probable tokens. To do this, first, we sort the vocabulary by probability and keep only the top k tokens:

$$V_k = \{w_1, w_2, \dots, w_k\}, \quad \text{where } P(w_i) \geq P(w_{i+1}) \text{ for } i < k$$

The probabilities within V_k are then normalized as follows:

$$P'(w) = \begin{cases} \frac{P(w)}{\sum\limits_{w' \in V_k} P(w')} & \text{if } w \in V_k \\ 0 & \text{otherwise} \end{cases}$$

A token is then randomly sampled from P'. As before, repeat the process until the EOS token is generated. Here, experiment with $k \in \{5, 10\}$ and report your findings.

Top-K	BLEU	ROUGE-1	ROUGE-2	ROUGE-LCS
$k = 5$	0.23	0.22	0.06	0.17
$k = 10$	0.21	0.22	0.05	0.16

Table 3: Results

1.4 Nucleus Sampling:

Instead of picking a fixed number of tokens as in Top-k Sampling, here we dynamically choose the smallest set of tokens whose cumulative probability exceeds a threshold p:

$$V_p = \{w_1, w_2, \dots, w_m\}, \quad \text{such that } \sum_{i=1}^{m} P(w_i) \geq p$$

$$P'(w) = \begin{cases} \frac{P(w)}{\sum\limits_{w' \in V_p} P(w')} & \text{if } w \in V_p \\ 0 & \text{otherwise} \end{cases}$$

Threshold p	BLEU	ROUGE-1	ROUGE-2	ROUGE-LCS
$p = 0.5$	0.27	0.25	0.09	0.19
$p = 0.9$	0.19	0.18	0.04	0.13

Table 4: Results

2 Task 1: Word-Constrained Decoding

Objective: To Improve LLM performance by restricting token selection based on a pre-defined word list provided.

Trie Data Structure:

- Used to store the given word list for efficient lookup.

- Enables quick verification of valid words during decoding.

Decoding Process:

1. Generate logits from the model and apply softmax to get probabilities.
2. Sort tokens by probability in descending order.
3. Iterate over sorted tokens and check if they form a valid word using the Trie.
4. Select the first valid token; if none, pick the most probable one.
5. Append the token to the generated sequence.
6. Stop when the EOS token appears or the maximum length is reached.

BLEU	ROUGE-1	ROUGE-2	ROUGE-LCS
0.35	0.43	0.25	0.37

Table 5: Findings for Task 3

Comparison with Other Methods:

- More constrained and accurate than Greedy Decoding, Top-k Sampling, and Random Sampling.
- Introduces minor computational overhead due to Trie lookups but significantly enhances output quality.

3 Task 2: Staring into Medusa's Heads

3.1 Medusa Architecture

Medusa extends the standard Language Model (LM) head by incorporating multiple speculative decoding heads. These heads predict future tokens beyond the immediate next token, enabling parallel decoding and improving inference speed.

3.2 Decoding Strategies

- **Single Head Decoding:** Uses only the LM head for inference. At each step, the most probable next token is greedily selected from the LM output distribution and fed back as input.
- **Multi Head Decoding:** Utilizes multiple Medusa heads to predict multiple future tokens in one step, reducing sequential dependency.

3.3 Multi Head Decoding Implementation:

Step 1: Compute Probability Distributions

- Obtain probability distributions $\{p_t, p_{t+1}, ..., p_{t+S}\}$ by passing the input sequence through the LLM.
- Here, p_t corresponds to the LM head output, while p_{t+k} corresponds to the k-th Medusa head prediction.

Step 2: Beam Search for Candidate Sequences

- Perform beam search with width W to generate candidate sequences $\{\hat{y}^1_{t+1}, \hat{y}^1_{t+2}, ..., \hat{y}^1_{t+S}\}, ..., \{\hat{y}^W_{t+1},$
- For each candidate, compute scores using log probabilities and retain the top W candidates.

Step 3: Final Token Selection

- Compute scores for all candidate sequences using the LM head and select the highest-scoring sequence.
- The final score is computed as:

$$\text{Score} = \sum_{i=t}^{t+S} \log p_i[\hat{y}_i] \tag{1}$$

- The decoding process repeats until an End-of-Sequence (EOS) token is generated.

3.4 Results:

Multi-Head Decoding	BLEU	ROUGE-1	ROUGE-2	ROUGE-LCS	RTF
H=2, W=2	0.16	0.17	0.032	0.15	0.05
H=2, W=5	0.20	0.22	0.047	0.17	0.11
H=2, W=10	0.26	0.34	0.097	0.27	0.20
H=5, W=2	0.07	0.09	0.008	0.08	0.03
H=5, W=5	0.09	0.11	0.016	0.09	0.11
H=5, W=10	0.10	0.12	0.017	0.10	0.17

Table 6: Multi-Head Decoding

BLEU	ROUGE-1	ROUGE-2	ROUGE-LCS	RTF
0.29	0.39	0.14	0.31	0.05

Table 7: Single-Head Greedy Decoding

Acknowledgments

I would like to express my sincere gratitude to Prof.Sunita Mam for her invaluable teaching and I am also thankful to the teaching assistants for the help I receieved, which greatly helped for this assignment.