Assignment 3 Report
CS-726: Advanced Machine Learning

Deeptanshu Malu Deevyanshu Malu Neel Rambhia

1 Task 0

Decoding Strategy	**BLEU**	**ROUGE-1**	**ROUGE-2**	**ROUGE-LCS**
Greedy	**0.3097**	**0.3538**	**0.1297**	**0.2704**
Random (temp=0.5)	0.2856	0.2929	0.1113	0.2387
Random (temp=0.9)	0.1996	0.1791	0.0550	0.1477
Top-k (k=5)	0.2388	0.2230	0.0607	0.1715
Top-k (k=10)	0.2427	0.2491	0.0799	0.2004
Nucleus Sampling (p=0.5)	0.2706	0.2554	0.0905	0.1973
Nucleus Sampling (p=0.9)	0.1957	0.1883	0.0469	0.1329

Table 1: Evaluation metrics for different decoding strategies.

Greedy decoding performs best across all metrics. On adding more randomness (higher temperature or larger sampling pools), the performance gets worse. Among sampling methods, Random with temperature=0.5 and Nucleus Sampling with p=0.5 work better than others. This tells us that some randomness can help, but too much hurts performance.

2 Task 1

The model used is `meta-llama/Llama-2-7b-chat-hf`.

We implemented constrained decoding using a trie data structure to limit the model's output to a predefined vocabulary. The algorithm selects the highest probability token that is valid according to the trie at each step.

We tested two trie traversal strategies: resetting the trie when reaching an end node (is_end) and resetting when a node has no children. The second strategy performed slightly better, as shown in the results table below.

Strategy	**BLEU**	**ROUGE-1**	**ROUGE-2**	**ROUGE-LCS**
Regular greedy	0.2781	0.3351	0.1230	0.2640
Reset when is_end	0.5111	0.5270	0.3189	**0.4694**
Reset when no children	**0.5116**	**0.5307**	**0.3205**	0.4675

Table 2: Evaluation metrics for constrained decoding strategies.

Algorithm 1 Constrained Decoding using Trie

Build trie from target vocabulary (word list)
Add EOS token to trie
$node \leftarrow trie.root$
for $i = 1$ to max_output_length **do**
    Compute logits for the last token in the sequence
    Convert logits to probabilities using softmax
    Sort probabilities in descending order
    for each token t in sorted order **do**
        if t is a child of current $node$ **then**
            Select t as the next token
            $node \leftarrow node.children[t]$
            if $node$ has no children **then** ▷ Can be replaced with $node.is\_end$ condition
                $node \leftarrow trie.root$ ▷ Reset trie state
            end if
            break
        end if
    end for
    Append selected token to the output sequence
    if selected token is EOS **then**
        break
    end if
end for
return generated sequence

Contributions

- **Deeptanshu Malu**:
    1.
- **Deevyanshu Malu**:
    1.
- **Neel Rambhia**:
    1.

Acknowledgements

-