

1. Inference-Time Compute

When generating outputs with an autoregressive sequence model, we face the question of inference-time compute: how to efficiently allocate computational resources to produce high-quality outputs. Different inference strategies have different computational profiles and can optimize for different objectives, possibly beyond just finding the highest probability sequence. Exhaustively evaluating all possible sequences is computationally intractable since there are $O(M^T)$ possibilities, where M is the vocabulary size and T is the sequence length. In this problem, we explore two major approaches with different compute-quality trade-offs: beam search, which systematically explores promising sequences with deterministic computation, and sampling-based methods, which introduce controlled randomness to generate diverse outputs with variable computation.

Sampling

Sampling is one approach to decoding sequences. We simply sample from the distribution of possible tokens at each timestep using (possibly adapted) probabilities from the model. This allows us to explore a wider range of possible sequences more efficiently. When you sample multiple sequences, you can use additional (possibly external) criteria to select the best sequence.

Method	Purpose	Sampling Adapter
Ancestral sampling	$y \sim p_\theta$	– (Uses original model probabilities without modification)
Temperature sampling	$y \sim q(p_\theta)$	Rescale (Divides logits by temperature parameter to control randomness)
Greedy decoding	$y \leftarrow \max p_\theta$	Argmax (temperature $\rightarrow 0$, selects highest probability token)
Top-k sampling	$y \sim q(p_\theta)$	Truncation (top-k, keeps only k highest probability tokens and renormalizes)
Top-p (nucleus) sampling	$y \sim q(p_\theta)$	Truncation (cumulative prob., keeps smallest set of tokens with cumulative probability at least p)

Table 1: Comparison of different decoding methods and their properties¹

- Suppose you are using language models for a integer answer questions. **How would you use inference-time compute to select the best sequence?** Suppose we don't have access to the internal model states (e.g. logits, suppose we are querying an API) and we cannot directly score model responses. If all we can do is query the model multiple times, how can we use inference-time compute to select the best possible answer?
- Now, suppose you are using language models for a code-generation task. Assume we have access to a syntax checker and can run our output against some test cases to validate our code. **How would you use inference-time compute to select the best sequence?**

¹From [Decoding to Meta-Generation: Inference-time Algorithms for Large Language Models](#)

Beam Search

Beam search is one of the popular approaches for finding the sequence with the highest overall probability according to the model.

It limits our search to only candidate sequences that are the most likely so far. In beam search, we keep track of the k most likely predictions of our model so far. At each timestep, we expand our predictions to all of the possible expansions of these sequences after one token, and then we keep only the top k of the most likely sequences out of these. In the end, we return the most likely sequence out of our final candidate sequences. This is also not guaranteed to be the true most likely sequence, but it is usually better than the result of just greedy decoding.

The beam search procedure can be written as the following pseudocode:

Algorithm 1 Beam Search

```

for each time step  $t$  do
  for each hypothesis  $y_{1:t-1,i}$  that we are tracking do
    find the top  $k$  tokens  $y_{t,i,1}, \dots, y_{t,i,k}$ 
  end for
  sort the resulting  $k^2$  length  $t$  sequences by their total log-probability
  store the top  $k$ 
  advance each hypothesis to time  $t + 1$ 
end for

```

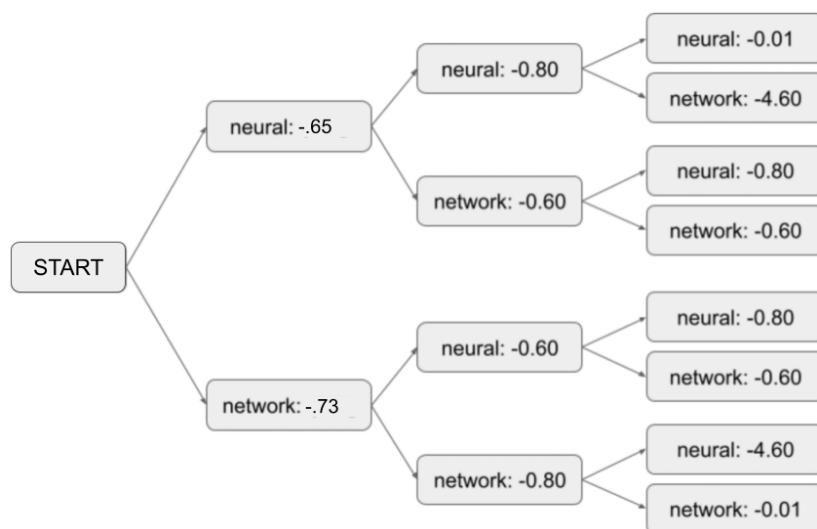


Figure 1: The numbers shown are the decoder’s log probability prediction of the current token given previous tokens.

- Suppose we have an algorithm which can evaluate our responses and generate a score after each token.
How would you use inference-time compute to select the best sequence?
- If we don’t have an external scoring algorithm, **how can we generate scores from the model itself?**
- We are running the beam search to decode a sequence of length 3 using a beam search with $k = 2$. Consider predictions of a decoder in Figure 1, where each node in the tree represents the next token **log probability** prediction of one step of the decoder conditioned on previous tokens. The vocabulary consists of two words: “neural” and “network”.
 - At timestep 1, which sequences are beam search storing?**

- ii. At timestep 2, which sequences are beam search storing?
 - iii. At timestep 3, which sequences are beam search storing?
 - iv. Does beam search return the overall most-likely sequence in this example? Explain why or why not.
 - v. What is the runtime complexity of generating a length- T sequence with beam size k with an RNN? Answer in terms of T and k and M .
- (d) Again, consider a code generation task. Assume we have a syntax checker which can incrementally check our output for syntax errors (suppose the checker can distinguish between things which are definitely errors vs. things that could be correct if completed). **How could we modify beam search to take advantage of the code checker?**

2. Finetuning Pretrained NLP Models

In this problem, we will compare finetuning strategies for three popular architectures for NLP.

- (a) **BERT** - encoder-only model
- (b) **GPT** - decoder-only model
- (c) **T5** - encoder-decoder model.

In contrast to BERT or GPT, T5 is trained using a different masking objective. Spans of multiple words are masked out from the encoder. The decoder must predict the tokens for each span, as shown in Figure 3.

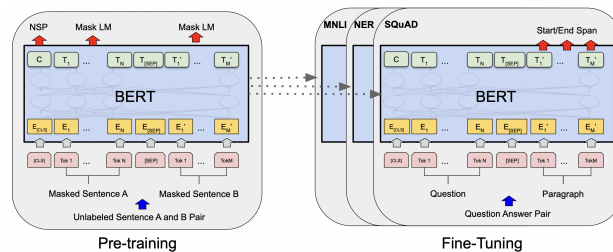


Figure 2: Overall pre-training and fine-tuning procedures for BERT.

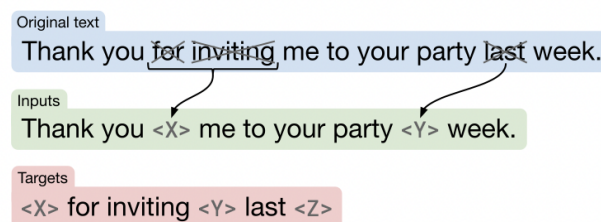


Figure 3: T5 Training procedure

Consider the MNLI (Multi-Genre Natural Language Inference Corpus) task. It provides a passage and a hypothesis, and you must state whether the hypothesis is an entailment, contradiction, or neutral.

EXAMPLE:

Passage: At the other end of Pennsylvania Avenue, people began to line up for a White House tour.

Hypothesis: People formed a line at the end of Pennsylvania Avenue.

Classification: entailment

- (i) With each of the 3 models, state whether it is possible to use the model for this task with no finetuning or additional parameters. If so, state how.
- (ii) With each of the 3 models, state how you would use the model for this task if you were able to add additional parameters and/or finetune existing parameters.
- (iii) How can we adapt soft-prompting to work with a T5-style model?

Contributors:

- CS 182 Staff from previous semesters.
- Olivia Watkins.
- Kumar Krishna Agrawal.
- Dhruv Shah.
- Naman Jain.
- Anant Sahai.
- Kevin Li.
- CS 182/282A Staff from previous semesters.