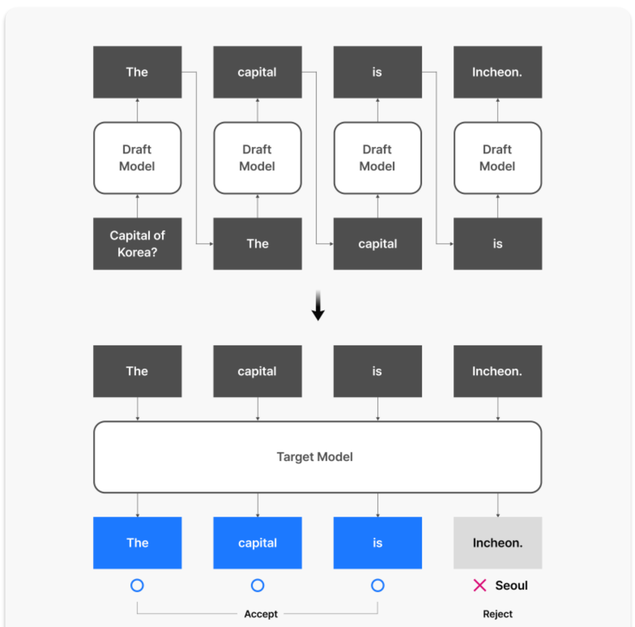
Ref:https://www.youtube.com/watch?v=VkWlLSTdHs8
https://pytorch.org/blog/hitchhikers-guide-speculative-decoding/
https://blog.naver.com/n_cloudplatform/223985359735

Speculative decoding is an optimization technique for inference that makes educated guesses about future tokens while generating the current token, all within a single forward pass. It incorporates a verification mechanism to ensure the correctness of these speculated tokens, thereby guaranteeing that the overall output of speculative decoding is identical to that of vanilla decoding. Optimizing the cost of inference of large language models (LLMs) is arguably one of the most critical factors in reducing the cost of generative AI and increasing its adoption. Towards this goal, various inference optimization techniques are available, including custom kernels, dynamic batching of input requests, and quantization of large models.

Core Idea:

Speculative Decoding, instead of generating tokens one by one as in the conventional method,
involves a **smaller** model predicting multiple tokens in advance,
and a larger model verifying them in parallel
to finalize the output.
This process can significantly boost generation speed.
When the smaller model (Draft Model) quickly proposes a draft,
the larger model (Target Model) checks it all at once:
if correct, it adopts it; if incorrect, it generates it directly.

The reason this method is effective is that,
because the next token in context is highly predictable,
the results from the smaller model are indeed adopted at a high rate
+
since the larger model can verify multiple tokens in parallel at once,
it can utilize hardware resources much more efficiently
than when generating tokens one by one.

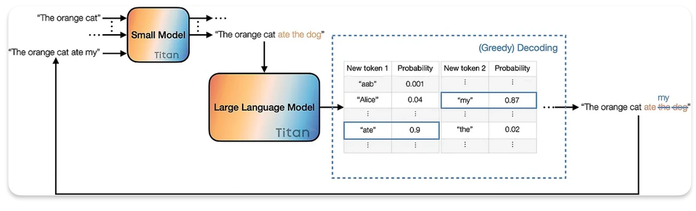


Process :

1 The Draft Model rapidly predicts multiple tokens based on the input.
2 The Target Model verifies these predicted values in parallel, all at once.
3 If the verified tokens match, they are adopted; if there's a mismatch, the Target Model directly generates the tokens.

The reason this method is effective is that,
because the next token in context is highly predictable,
the results from the smaller model are indeed adopted at a high rate

"Speculative execution refers to a process where an operation is performed concurrently with the verification of whether that operation is actually necessary. A prime example of speculative execution is the well-known branch prediction."



Advantage:

- **It leverages predictability:** As you mentioned, language is inherently predictable, allowing the smaller, faster model to make good guesses most of the time.
- **It exploits parallelization:** Modern hardware is optimized for parallel processing, and Speculative Decoding capitalizes on this by allowing the larger model to verify multiple tokens simultaneously, rather than sequentially.
- **It maintains quality:** Crucially, it achieves speed gains without sacrificing the quality of the output, as the final decision rests with the more accurate (and often larger) target model.
- **It's practical:** The ability to integrate it relatively easily with existing model architectures (given shared tokenizers and training setups) makes it a very appealing optimization strategy.

Disadvantages:

- **Implementation Complexity:** Managing and optimizing two models (the draft model and the verifier model) can make the implementation more complex.
- **Draft Model Selection:** The size and quality of the draft model influence overall performance. If it's too small, prediction accuracy decreases; if it's too large, the speed advantage is reduced

In Speculative Decoding, the key factor determining performance is the **choice of the smaller model** responsible for prediction.
If the model is too simple, its accuracy will be low, leading to frequent failures in verification.Conversely, if it's too large, the speed difference will diminish, potentially reducing efficiency.

Ultimately, what matters is how frequently the predicted results are actually adopted as the final output.
If predictions are accurate, they quickly pass the verification stage, improving speed. However, inaccurate predictions require re-computation, increasing the overall processing time.

Therefore, it's not simply about choosing a "faster model,"
but rather about finding a balance between prediction utilization rate and computational speed.