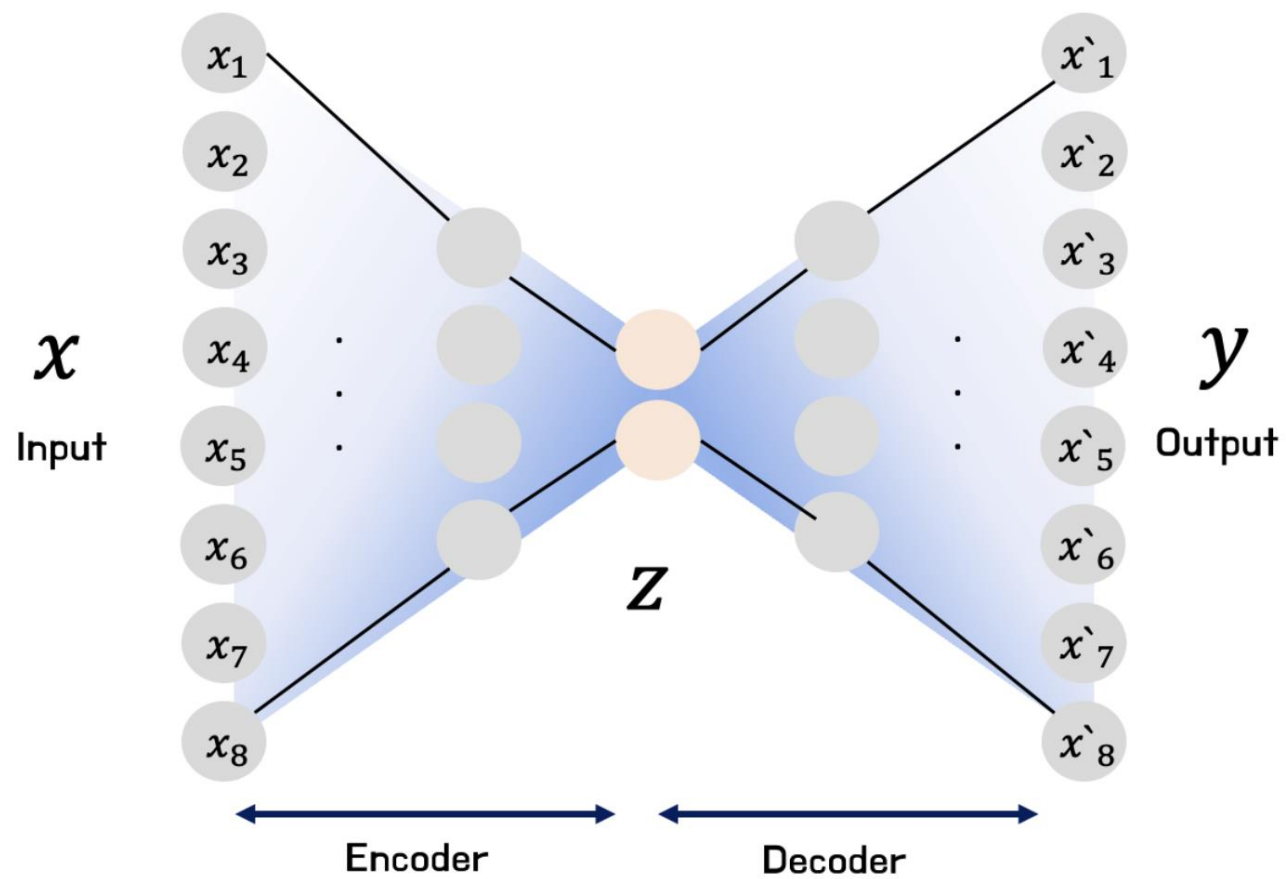


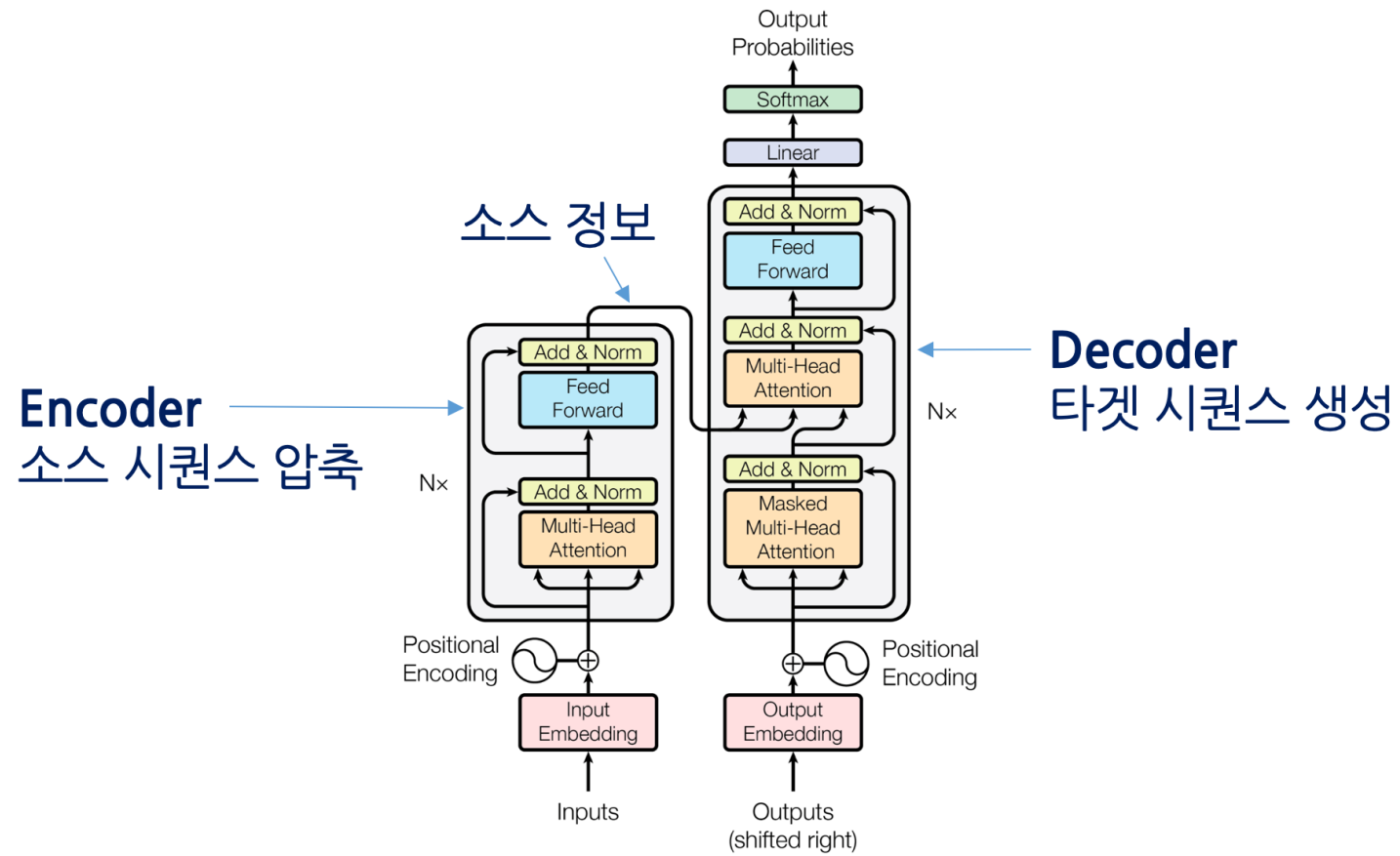
AI 11주차 LLM

텍스트 생성

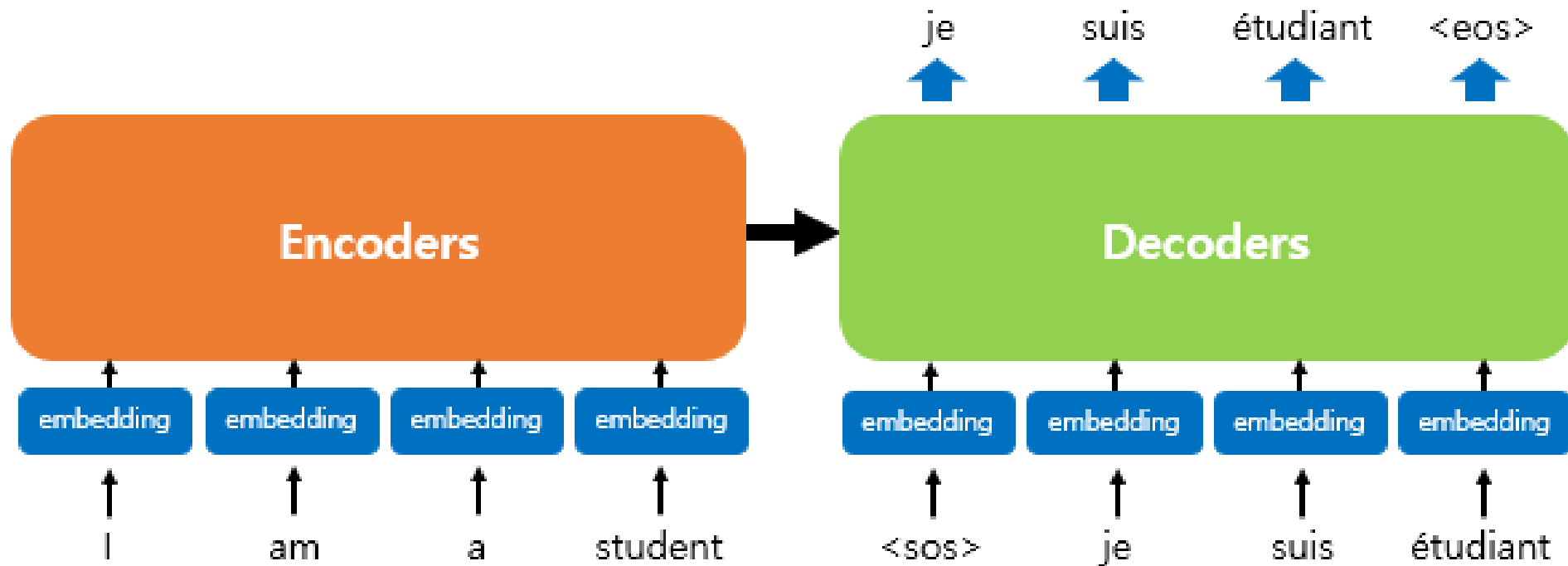
오토인코더



트랜스포머

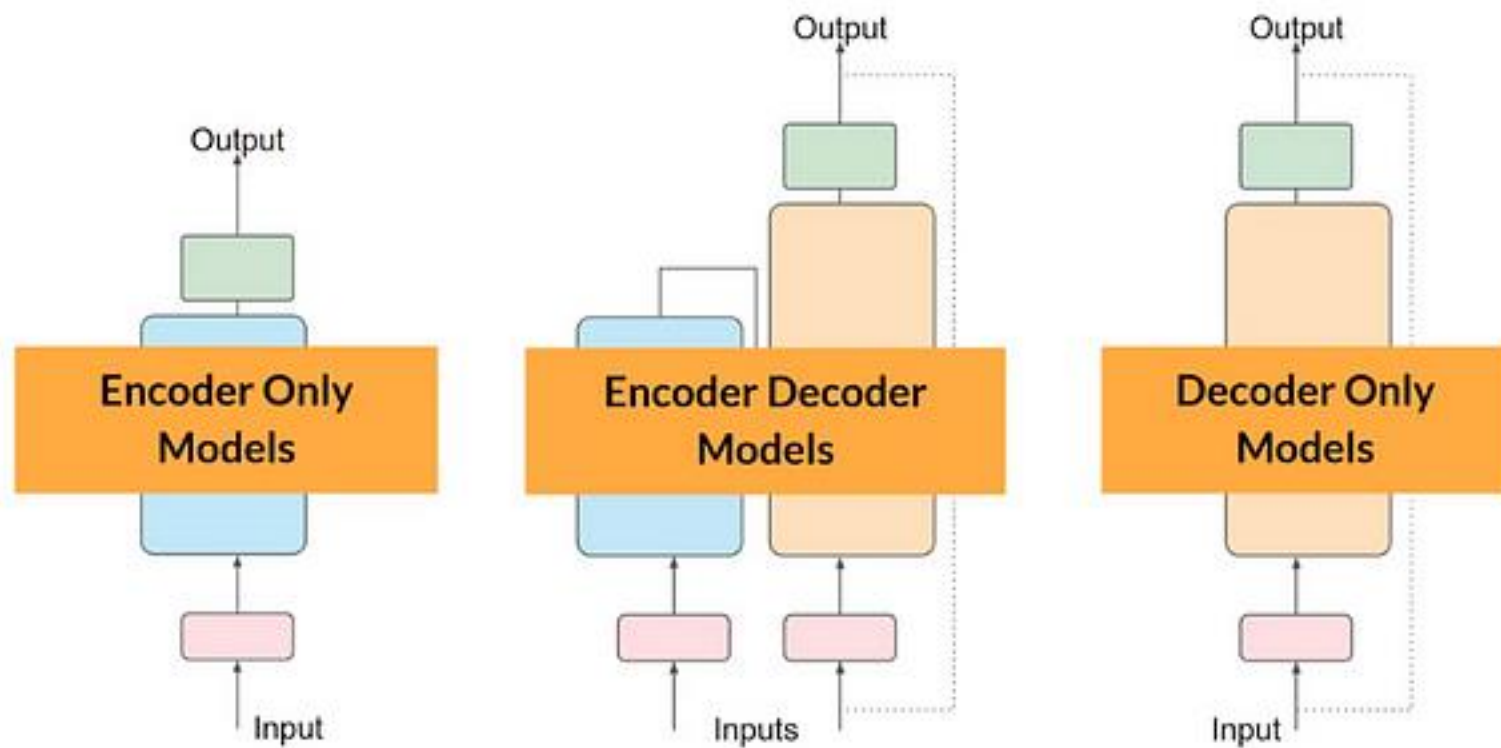


트랜스포머 핵심 - 인코더 디코더



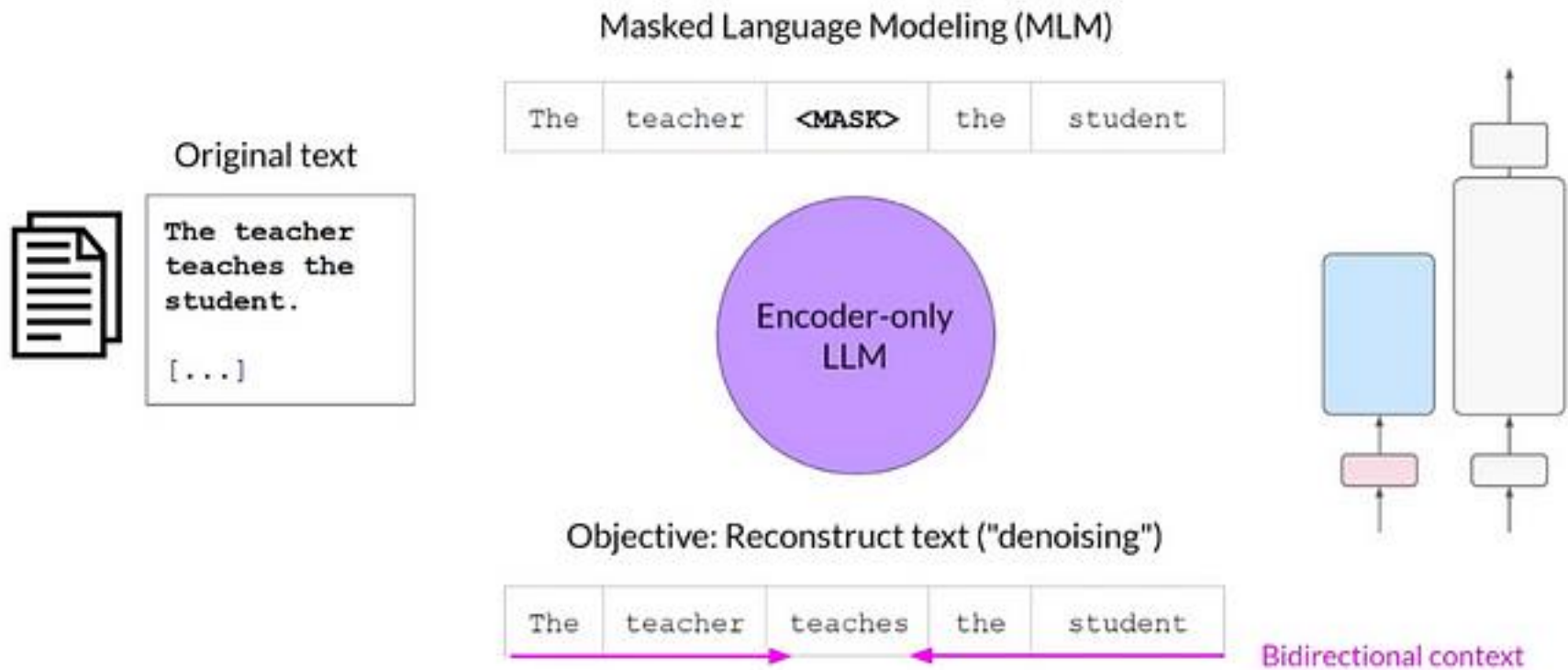
트랜스포머 종류

Transformers



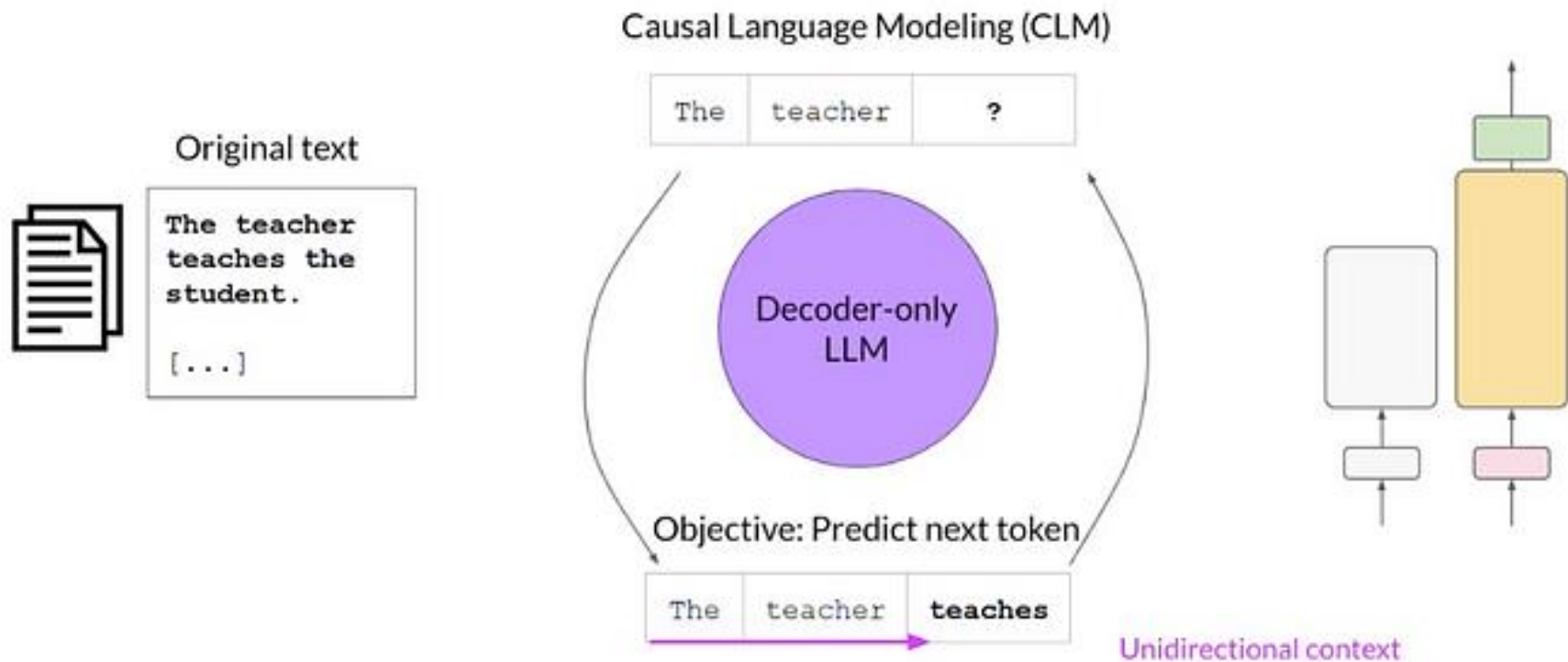
인코딩만 사용

Autoencoding models



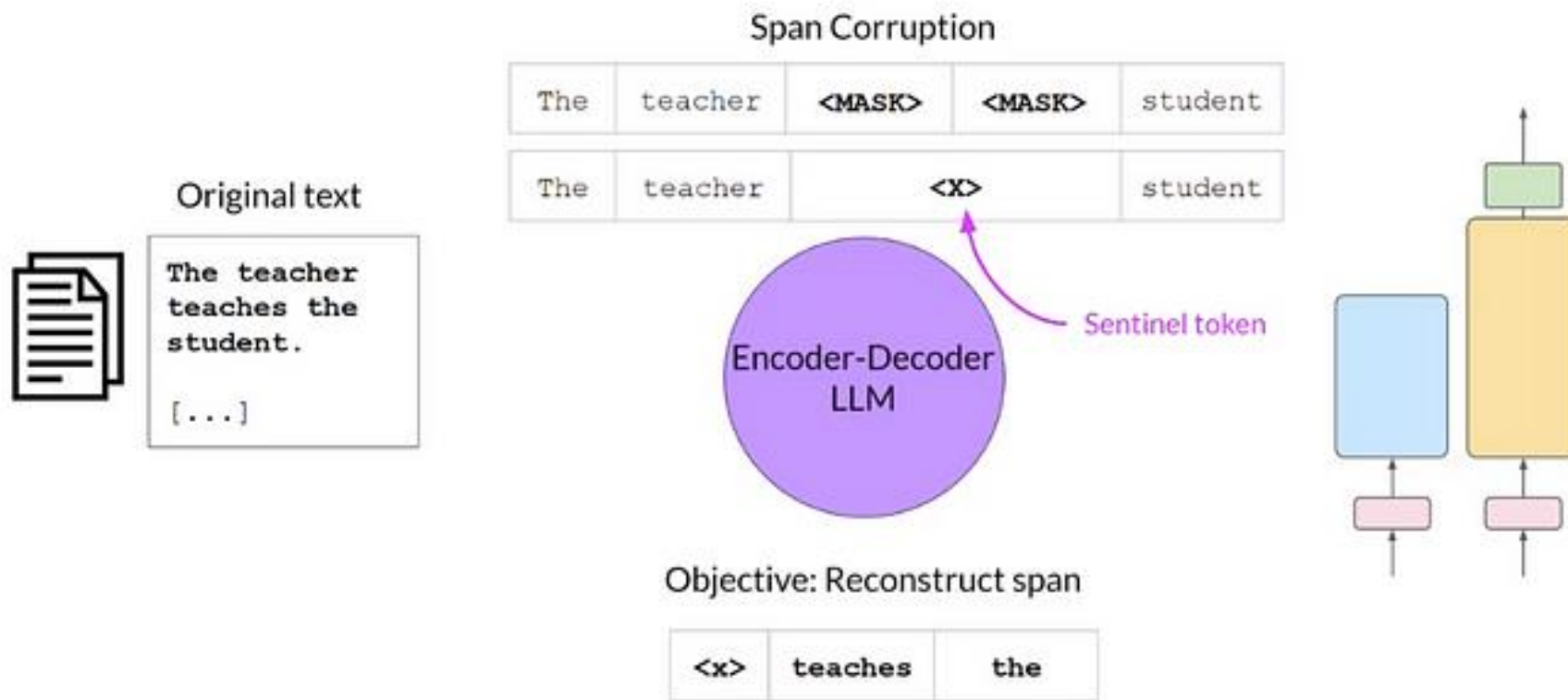
디코더만 사용

Autoregressive models



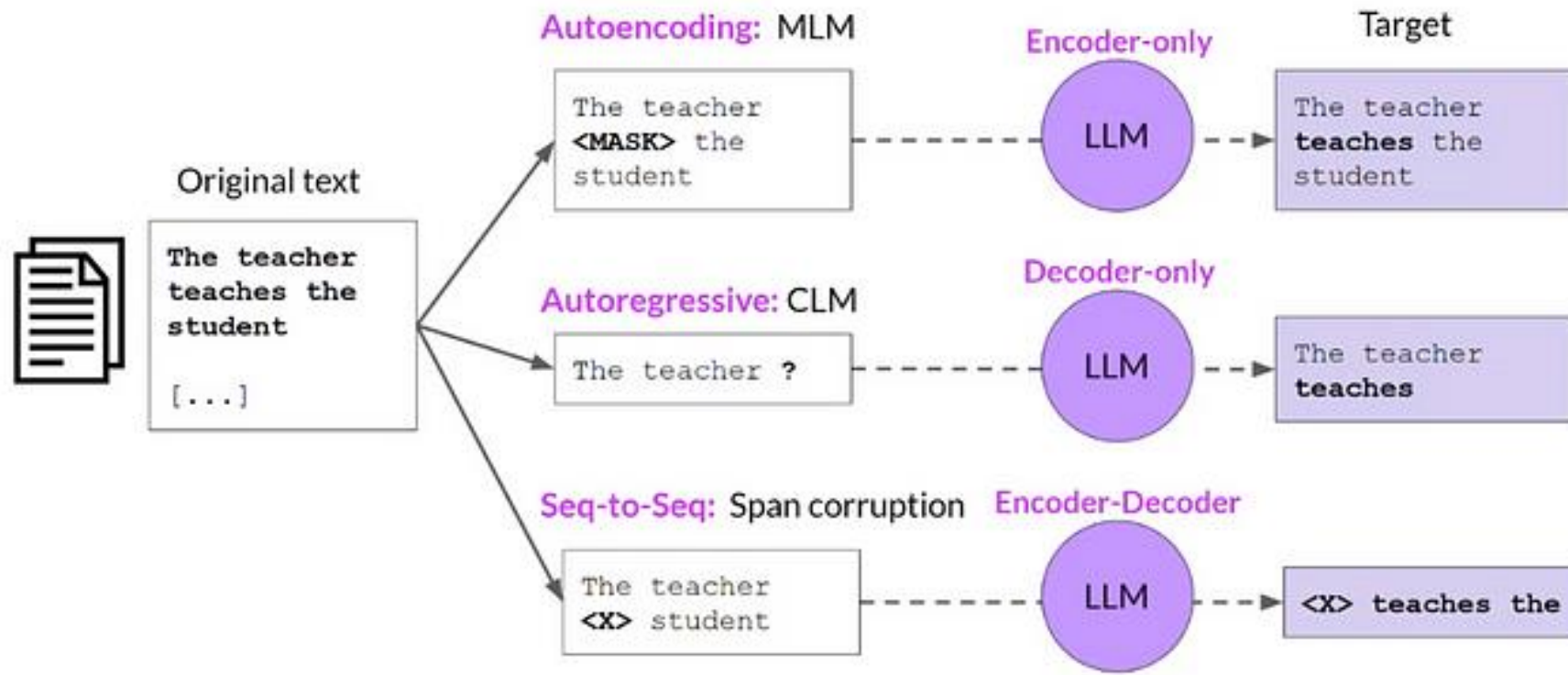
인코딩 & 디코딩 같이 사용

Sequence-to-sequence models



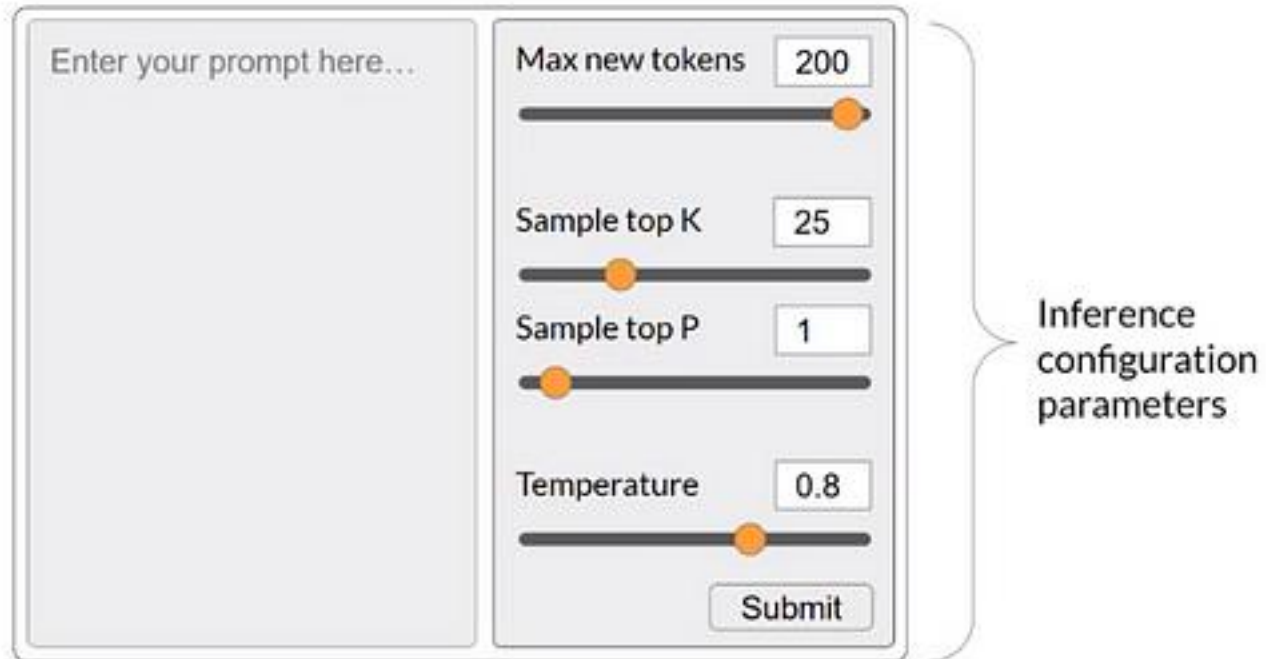
용도에 따른 트랜스포머

Model architectures and pre-training objectives



LLM Configuration

Generative configuration - inference parameters



The image shows a user interface for configuring an LLM. On the left is a large text area with the placeholder "Enter your prompt here...". To the right of this area are four sliders, each with a numerical input box to its right. The sliders are for "Max new tokens" (set to 200), "Sample top K" (set to 25), "Sample top P" (set to 1), and "Temperature" (set to 0.8). Below these sliders is a "Submit" button. A large curly bracket on the right side of the sliders is labeled "Inference configuration parameters".

Parameter	Value
Max new tokens	200
Sample top K	25
Sample top P	1
Temperature	0.8

Submit

Inference configuration parameters

"Max new tokens"

- 모델이 생성하는 토큰의 수에 제한을 둡니다. 그러나 다른 중단 조건으로 인해 완성의 실제 길이는 다를 수 있습니다.

"Greedy Decoding"

- 다음 단어 예측을 위한 가장 간단한 방법으로, 가장 높은 확률을 가진 단어를 선택합니다. 그러나 이는 반복된 단어나 시퀀스를 결과로 가져올 수 있습니다.

"Random Sampling"

- - 확률 분포를 기반으로 단어를 무작위로 선택하여 다양성을 도입하며, 단어의 반복 가능성을 줄입니다.

"Top-K"

- 가장 높은 확률을 가진 k개의 토큰 중에서 선택하여 옵션을 제한하며, 높은 무작위성을 촉진하면서도 매우 불가능한 완성을 방지합니다.

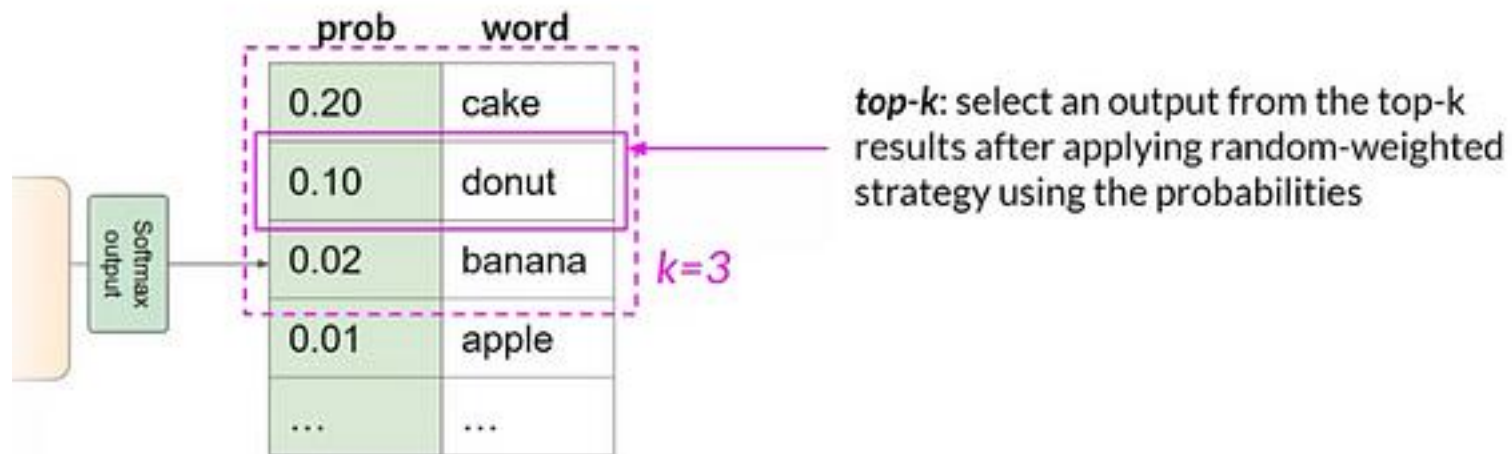
"Top-P"

- 누적 확률이 지정된 임계값을 초과하지 않는 예측에 무작위 샘플링을 제한하여 합리적인 출력을 보장합니다.

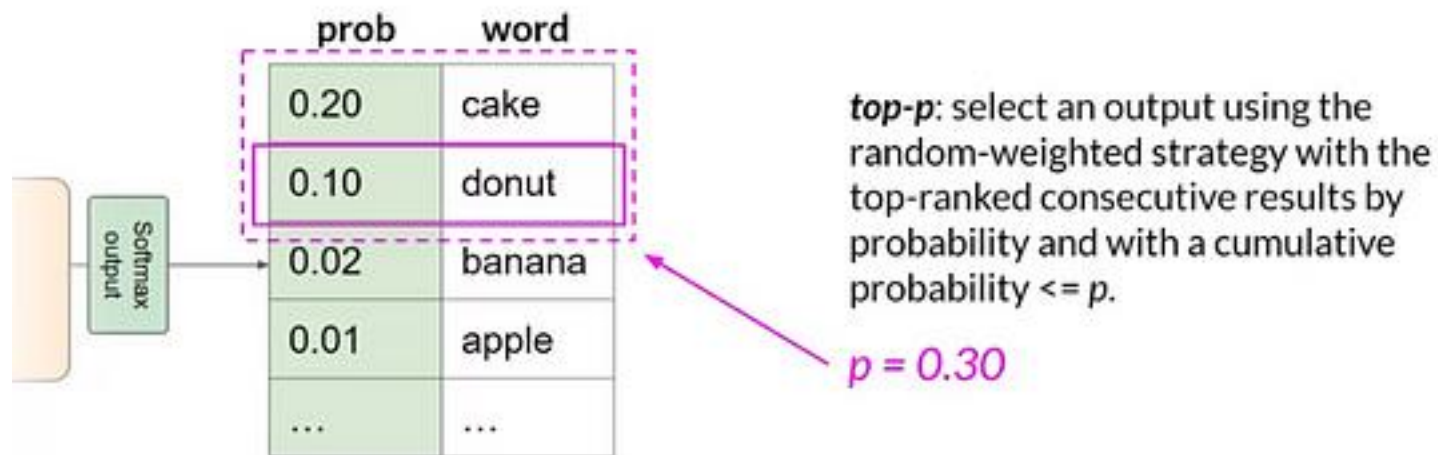
"Temperature"

- - 확률 분포의 형태에 영향을 미칩니다. 높은 온도 값은 무작위성을 증가시키며, 낮은 값은 확률을 더 작은 단어 집합에 집중시킵니다.

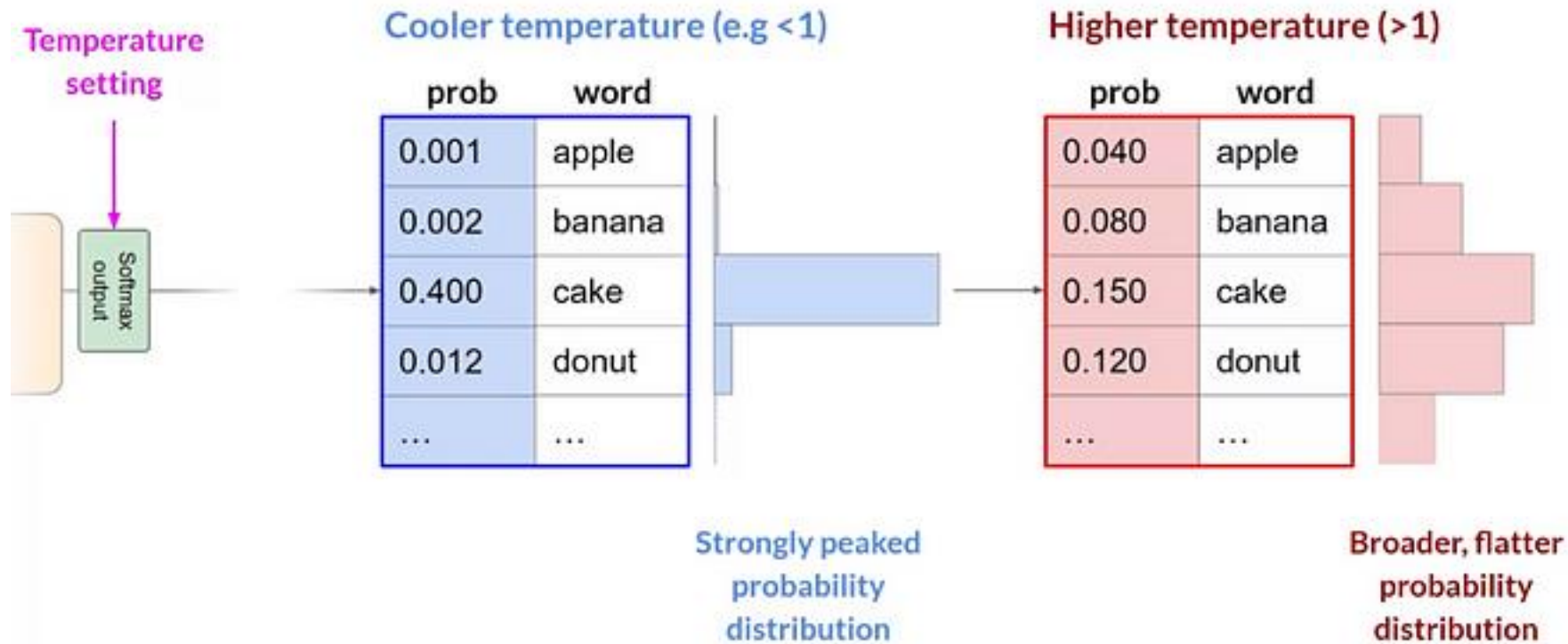
Generative config - top-k sampling



Generative config - top-p sampling



Generative config - temperature

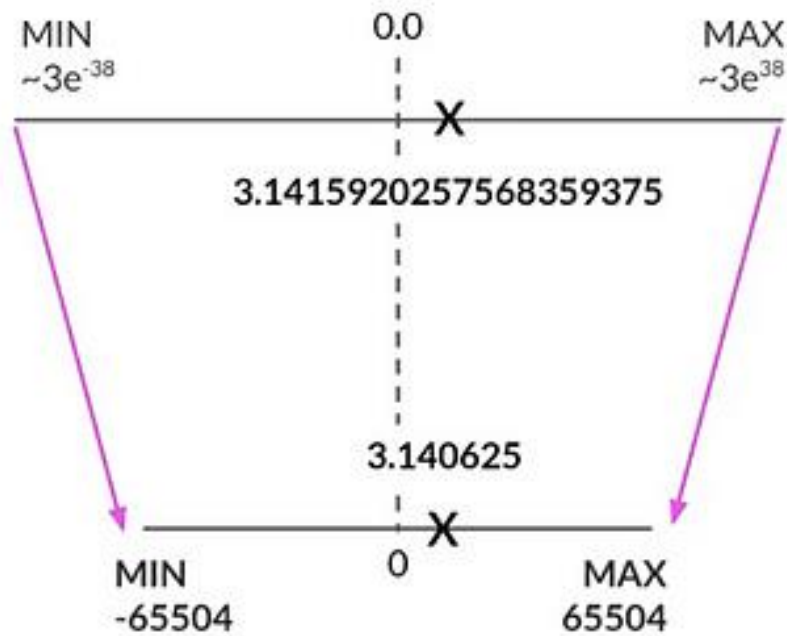


Additional GPU RAM needed to train 1B parameters

	Bytes per parameter
Model Parameters (Weights)	4 bytes per parameter
Adam optimizer (2 states)	+8 bytes per parameter
Gradients	+4 bytes per parameter
Activations and temp memory (variable size)	+8 bytes per parameter (high-end estimate)
TOTAL	=4 bytes per parameter +20 extra bytes per parameter

~20 extra bytes
per parameter

Quantization: FP16



Let's store Pi: 3.141592

FP32 4 bytes memory

0	10000000	10010010000111111011000
<hr/>		
Sign 1 bit	Exponent 8 bits	Fraction 23 bits

FP16 2 bytes memory

0	10000	1001001000
<hr/>		
Sign 1 bit	Exponent 5 bits	Fraction 10 bits

LLaDA Model

Large Language Diffusion Models

Shen Nie^{1*†} Fengqi Zhu^{1*†} Zebin You^{1†} Xiaolu Zhang^{2‡} Jingyang Ou¹ Jun Hu^{2‡} Jun Zhou²
Yankai Lin^{1‡} Ji-Rong Wen¹ Chongxuan Li^{1‡¶}

Abstract

Autoregressive models (ARMs) are widely regarded as the cornerstone of large language models (LLMs). We challenge this notion by introducing **LLaDA**, a diffusion model trained from scratch under the pre-training and supervised fine-tuning (SFT) paradigm. LLaDA models distributions through a forward data masking process and a reverse process, parameterized by a vanilla Transformer to predict masked tokens. By optimizing a likelihood bound, it provides a principled generative approach for probabilistic inference. Across extensive benchmarks, LLaDA demonstrates strong *scalability*, outperforming our self-constructed ARM baselines. Remarkably, LLaDA 8B is competitive with strong LLMs like LLaMA3 8B in *in-context learning* and, after SFT, exhibits impressive *instruction-following* abilities in case studies such as multi-turn dialogue. Moreover, LLaDA addresses the reversal curse, surpassing GPT-4o in a reversal poem completion task. Our findings establish diffusion models as a viable and promising alternative to ARMs, challenging the assumption that key LLM capabilities discussed above are inherently tied to ARMs. Project page and codes: <https://ml-gsai.github.io/LLaDA-demo/>.

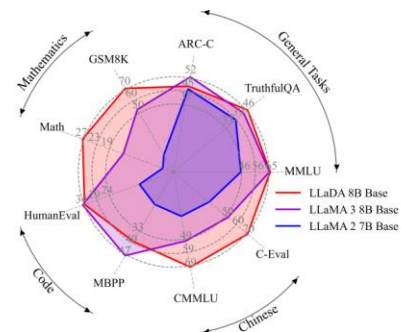


Figure 1. **Zero/Few-Shot Benchmarks.** We scale LLaDA to an unprecedented size of 8B parameters from scratch, achieving competitive performance with strong LLMs (Dubey et al., 2024).

distribution $p_{\text{data}}(\cdot)$ by optimizing a model distribution $p_{\theta}(\cdot)$ through maximum likelihood estimation, or equivalently KL divergence minimization between the two distributions:

$$\max_{\theta} \mathbb{E}_{p_{\text{data}}(x)} \log p_{\theta}(x) \Leftrightarrow \min_{\theta} \text{KL}(p_{\text{data}}(x) || p_{\theta}(x)). \quad (1)$$

Generative modeling principles

The predominant approach relies on the *autoregressive* modeling (ARM)—commonly referred to as the *next-token prediction* paradigm—to define the model distribution:

What is 2 + 2 ? <MASK> <MASK> <MASK> <MASK> <MASK> ...

Prediction

What is 2 + 2 ? The result is 5 <EOS>

Remasking

What is 2 + 2 ? The <MASK> <MASK> <MASK> <EOS>

Prediction

What is 2 + 2 ? The answer is 4 <EOS>

Remasking

What is 2 + 2 ? The answer <MASK> 4 <EOS>

Prediction

What is 2 + 2 ? The answer is 4 <EOS>

High-confidence tokens
Low-confidence tokens

LLaDA PREVIEW

