Large Concept Models: Language Modeling in a Sentence Representation Space

The LCM team, Loïc Barrault∗, Paul-Ambroise Duquenne∗, Maha Elbayad∗, Artyom Kozhevnikov∗, Belen Alastruey†, Pierre Andrews†, Mariano Coria†, Guillaume Couairon+†, Marta R. Costa-jussà†, David Dale†, Hady Elsahar†, Kevin Heffernan†, João Maria Janeiro†, Tuan Tran†, Christophe Ropers†, Eduardo Sánchez†, Robin San Roman†, Alexandre Mourachko‡, Safiyyah Saleem‡, Holger Schwenk

FAIR at Meta

Artificial Intelligence

Creating the Future

Dong-A University

Division of Computer Engineering & Artificial Intelligence

References

Large Concept Model

- Github
  - LCM https://github.com/facebookresearch/large_concept_model
  - SONAR https://github.com/facebookresearch/SONAR
- Hugging Face : https://huggingface.co/papers/2412.08821
- Paper : https://arxiv.org/abs/2412.08821
  https://arxiv.org/html/2412.08821v2

arXiv

- H. Ahmad 등, The Future of AI Exploring the Potential of Large Concept Models, https://arxiv.org/html/2501.05487v1

Blog

- Datacamp, https://www.datacamp.com/blog/large-concept-models
- pytorch korea, https://discuss.pytorch.kr/t/lcm-large-concept-models-meta-ai/5744
- AI papers academy, https://aipapersacademy.com/large-concept-models/

Introduction

➢ Abstract

- **Current technology of LLMs** is to process input and generate output **at the token level**.

↔ Sharp contrast to humans who **operate at multiple levels of abstraction**, **well beyond single words**, to **analyze information** and to **generate creative content**.

- We present an attempt at an architecture which **operates on an explicit higher-level semantic representation**, which we name a "**concept**".
- **Concepts** are **language- and modality-agnostic** and represent a **higher level idea or action in a flow**. Hence, we build a "**Large Concept Model**".
- Assume that **a concept corresponds to a sentence**, and **use an existing sentence embedding space**, ***SONAR***, which supports up to **200 languages** in both text and speech modalities.

🚀 **SONAR** stands for **S**entence-level multim**O**dal and la**N**guage-**A**gnostic **R**epresentations

- Large Concept Model is **trained to perform autoregressive sentence prediction in an embedding space**.
- We explore **multiple approaches**, namely **MSE regression**, **variants of diffusion-based generation**, and **models operating in a quantized SONAR space**.
- These explorations are performed using **1.6B parameter models** and **training data** in **the order of 1.3T tokens**.
- We then scale one architecture to a model size of **7B parameters** and training data of about **2.7T tokens**.

- Experimental evaluation on several generative tasks, namely **summarization** and **a new task of summary expansion**.
- Our model exhibits **impressive zero-shot generalization performance** to many languages, outperforming existing LLMs of the same size.

Introduction

☑ LLM Status

- **Impressive performance on a large variety of tasks**, such as providing detailed answers for general knowledge questions, helping in performing long document analysis, or drafting different types of messages, and writing or debugging code.
- **Building an LLM from scratch** requires access to **enormous computational resources** to process ever **larger amounts of data** and **train models**, the size of which now **exceeds four hundred billion parameters**.
- The landscape of available LLMs can be structured into

✓ **Open models** such as **Llama** (The Llama3 team, 2024), **Mistral** (Jiang et al., 2024), **Bloom** (BigScience Workshop, 2023) or **Falcon** (Almazrouei et al., 2023), on the one hand,

✓ **Closed models** such as **Gemini** (Gemini Team Google, 2024), **GPT** (OpenAI, 2024) or **Claude** (Anthropic, 2024), on the other.

- All these models are based on the same underlying architecture: a **transformer-based**, **decoder-only language model**, which is **pretrained to predict the next token**, **given a long context of preceding tokens**.

- Current LLMs **miss a crucial characteristic of human intelligence**: **explicit reasoning and planning at multiple levels of abstraction**.

☑ **The human brain does not operate at the word level only.**

- We usually have a **top-down process** to solve a complex task or compose a long document: First plan at a higher level the overall structure, and then step-by-step, add details at lower levels of abstraction.

☑ **Models with an explicit hierarchical architecture** are better suited to **create coherent long-form output**

Introduction

☑ Human Conception

- 한 연구원이 15분 동안 강연하는 경우

✓ 일반적으로 발표할 모든 단어를 일일이 적어서 연설을 준비하지 않음.

✓ 대신 전달하고자 하는 상위 수준의 아이디어의 흐름을 개략적으로 설명함.

✓ 같은 강연을 여러 번 할 경우 실제로 사용하는 단어는 다를 수 있고, 다른 언어로 강연을 할 수도 있지만, 추상적인 아이디어의 흐름은 동일하게 유지됨

- 특정 주제에 대한 연구 논문이나 에세이를 작성할 때

✓ 일반적으로 전체 문서를 섹션으로 구성하는 개요를 작성하는 것부터 시작하여 반복적으로 내용을 수정함.

✓ 인간은 추상적인 수준에서 긴 문서의 여러 부분 간의 의존성을 감지하고 기억함. 이전 연구 문서 작성 예시를 확장하면, 종속성을 추적한다는 것은 서론에서 언급한 각 실험에 대한 결과를 제공해야 한다는 것을 의미함.

- 정보를 처리하고 분석할 때

✓ 인간은 큰 문서에서 모든 단어를 일일이 고려하는 경우는 거의 없음.

✓ 대신, 긴 문서에서 특정 정보를 찾기 위해 어느 부분을 검색해야 하는지 기억하는 계층적 접근 방식을 사용함.

☑ Our fundamental idea

- Based on any **fixed-size sentence embedding space** for which **an encoder and decoder are available**.
- Particularly, aim to **train a new embedding space** specifically **optimized to our reasoning architecture**.
- We chose an existing and freely available sentence embedding, named **SONAR** (Duquenne et al., 2023b).

✓ SONAR supports **text input and output in 200 languages**, **speech input in 76 languages**, and **speech output in English**.

Introduction

☑ Concepts of Large Concept Model

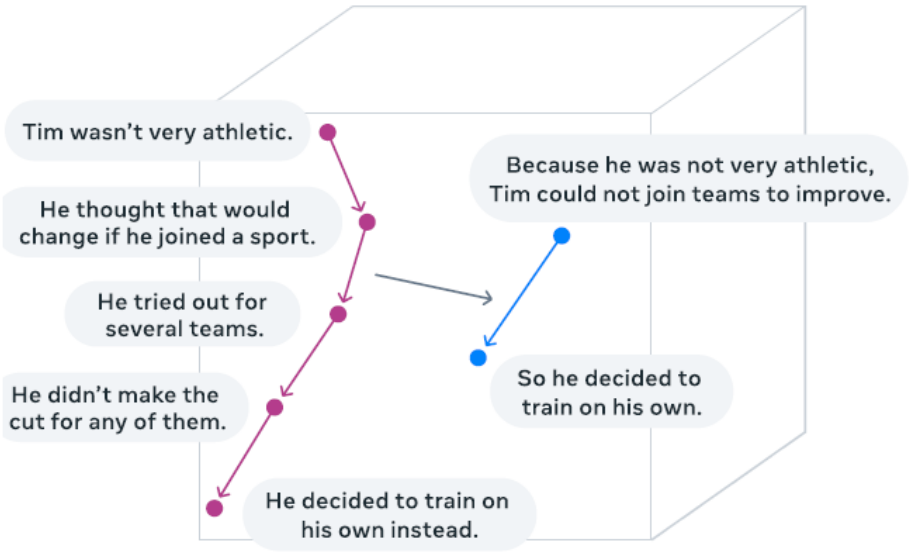
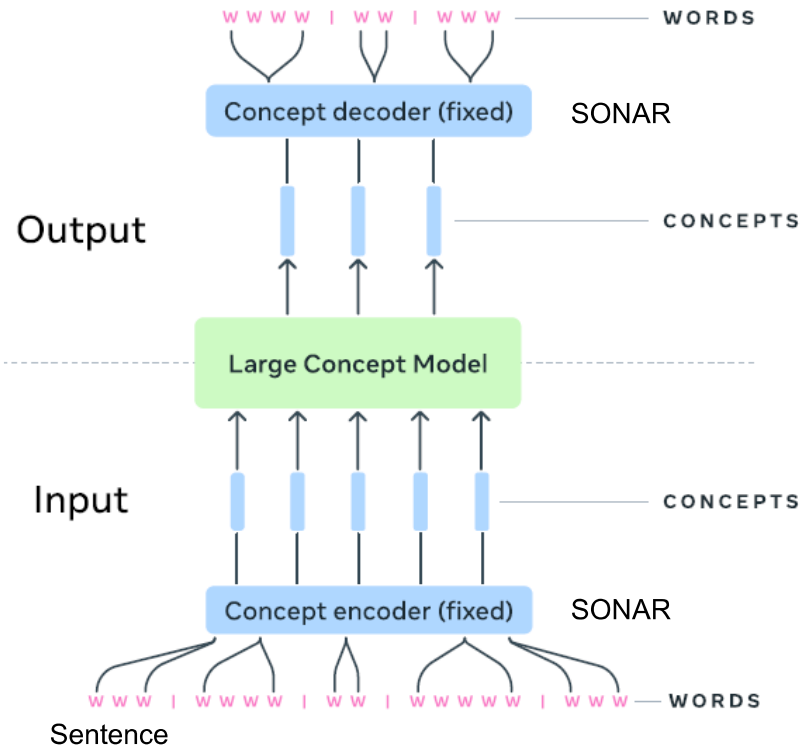


Figure 1. Left: visualization of reasoning in an embedding space of concepts (task of summarization), which is materialized by a function on the embedding space, mapping five concept representations into two.

Introduction

☑ Concepts of Large Concept Model



3) The generated concepts are decoded by **SONAR** into a sequence of subwords. The encoder and decoder are fixed and are not trained.

2) This sequence of concepts is processed by a Large Concept Model (LCM) to generate at the output a new sequence of concepts.

1) The input is first **segmented into sentences**, and each one is **encoded with SONAR to achieve a sequence of concepts**, i.e., **sentence embeddings.**

※ The unchanged sequence of concepts at the output of the LCM can be decoded into other languages or modalities without performing again the whole reasoning process.

※ A particular reasoning operation such as summarization can be performed in a zero-shot setting on input in any language or modality, since it solely operates on concepts.

Figure 1. Right: fundamental architecture of a Large Concept Model (LCM).
⋆: concept encoder and decoder are frozen.

Introduction

☑ Concepts of Large Concept Model

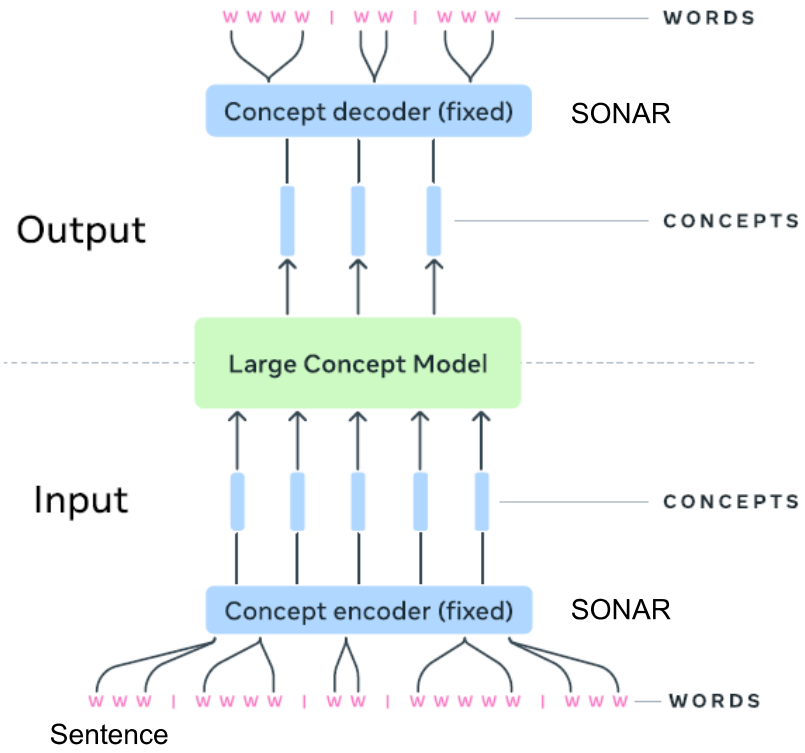


Figure 1. Right: fundamental architecture of a Large Concept Model (LCM). ⋆: concept encoder and decoder are frozen.

🚀 To summarize,

- LCM **neither** has **information on the input language or modality nor generates output** in a particular language or modality.
- We explore multiple architectures to train the LCM, in particular several variants of diffusion.
- We **envision an additional level of abstraction beyond concepts** which could correspond to a short description of a paragraph or small section.
- We report **initial ideas** on **how conditioning and predicting such higher-level representations can improve consistency of output generated by an LCM**.

🚀 LCM & JEPA

- LCM architecture **resembles** the JEPA approach (LeCun, 2022) that aims to **predict the representation of the next observation in an embedding space**.
- **JEPA** places more emphasis on **learning a representation space in a self-supervised way**, the **LCM** focuses on accurate prediction in **the existing embedding space**.

Introduction

☑ The mains characteristics of our generic Large Concept Model approach

➢ **Reasoning at an abstract language- and modality-agnostic level beyond tokens**:

- We model the underlying reasoning process, not its instantiation in a particular language.
- LCM can be trained, i.e. acquire knowledge, on all languages and modalities at once, promising scalability in an unbiased way.

➢ **Explicit hierarchical structure:**

- Better readability of long-form output by a human.
- Facilitates local interactive edits by a user.

➢ **Unparalleled zero-shot generalization:**

- Independently of the language or modality, the LCM is pre-trained and fine-tuned on, it can be applied to any language and modality supported by the SONAR encoders, without the need of additional data or fine-tuning.

➢ **Handling of long context and long-form output**:

- The complexity of a vanilla transformer model increases quadratically with the sequence length.
- This makes handling of large context windows challenging and several techniques have been developed to alleviate this problem, e.g., *sparse attention* (Child et al., 2019) or *LSH attention*(Kitaev et al., 2020).
- Our LCM operates on sequences which are at least an order of magnitude shorter.

➢ **Modularity and extensibility:**

- Unlike multimodal LLMs that can suffer from modality competition (Aghajanyan et al., 2023; Chameleon team, 2024), concept encoders and decoders can be independently developed and optimized without any competition or interference.
- New languages or modalities can be easily added for an existing system.

Main Design Principles

2.1 The SONAR embedding space

➢ The **motivation** is to perform **reasoning at a higher conceptual level than tokens**.

- This requires **an embedding space** which is **highly semantic**.
- We chose **SONAR** (Duquenne et al., 2023b) since it **achieves best performance on several semantic similarity metrics** like xsim or xsim++ (Chen et al., 2023b), and it was successfully **used in large-scale bitext mining for translation** (Seamless Communication et al., 2023b).

➢ **SONAR text embedding**

- **SONAR text embedding space** was **trained as an encoder/decoder architecture**, with **a fixed-size bottleneck** instead of cross-attention (see Figure 2).
- The **criterion** combines a machine translation objective for 200 languages into and out of English, denoising auto-encoding and an explicit MSE loss at the embedding bottleneck layer.
- Once the text embedding space was trained, **a teacher-student approach** was applied to extend the SONAR space to the **speech modality**.

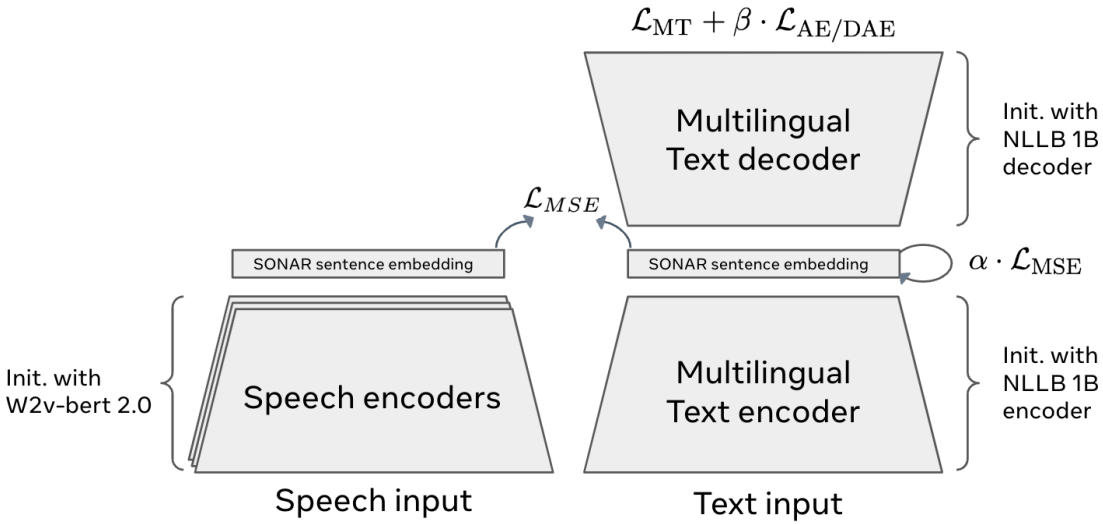


Figure 2 - Encoder/decoder bottleneck architecture to train the SONAR text embeddings (right part of figure). Teacher-student approach to extend SONAR to the speech modality (left part).

❖ P.-A. Duquenne, H. Schwenk, and B. Sagot. SONAR: sentence-level multimodal and language-agnostic representations, 2023b. URL https://arxiv.org/abs/2308.11466.

Main Design Principles

2.1 The SONAR embedding space

- LCM operates directly on SONAR concepts embeddings, hence, it can perform reasoning on all supported languages and modalities.

Table 1 - Comparison of language and modality coverage for several LLMs and our LCM operating on the SONAR embedding space. SONAR has an experimental support for American Sign Language (ASL) which is not used in this paper.

	Text		Speech		Image		Video	
Model	Input	Output	Input	Output	Input	Output	Input	Output
GEMINI	47	47	62	✓	✓	✓	✓	✗
GPT	85	85	✓	✓	✓	✓	?	✗
CLAUDE	37	37	✓	✓	✓	✓	✗	✗
BLOOM	46	46	✗	✗	✓	✓	✗	✗
LLAMA 3-400B	8	8	34	✗	✓	✓	✗	✗
LCM-SONAR	200	200	76	1	✗	✗	(ASL)	✗

Main Design Principles

2.2 Data preparation

➢ We need to convert **raw text datasets** into **a sequence of SONAR embeddings**, each one corresponding to a sentence.

➢ Dealing with **large text corpora** presents **several practical limitations.**

- First, **the precise segmentation of a text into sentences** can be challenging due to the presence of errors, specific formatting issues or any other sources of noise.
  - ✓ Require **robust automatic text segmentation** techniques.
- Second, some sentences (even well formed) can be **very long** and **complex**, which might negatively impact the quality of the encoded SONAR embeddings.

❖ **Sentence segmentation techniques**

- **SpaCy segmenter (SpaCy)** (Honnibal et al., 2020)
  - Well established multilingual NLP toolkit
  - Provide a rule-based approach to sentence segmentation.
- **Segment any Text (SaT)** (Minixhofer et al., 2023; Frohmann et al., 2024)
  - Offers a suite of models and adapters that predict sentence boundaries at the token level.
  - Designed to be resilient to perturbations, particularly avoiding the over-reliance on punctuation and capitalization.
  - The segmentation quality depends on the choice of an "appropriate" split probability threshold.

Main Design Principles

2.2 Data preparation

❖ **Sentence segmentation techniques**

- We customize both methods by incorporating a maximum sentence length cap in characters. Refer ***SpaCy Capped*** and ***SaT Capped***.

- Long sentences are broken down into smaller, logically coherent fragments

✓ Using **a rule-based approach based on punctuation marks for SpaCy**.

✓ Leverage **the provided splitting probability estimates** to identify the next best potential split for **SaT**.

➢ **Measure a segmenter** - Evaluate the quality of the reconstructed sentences with **AutoBLEU**.

- **BLEU score** (Papineni et al., 2002) : Compare the decoded text from a SONAR vector after encoding a segment, to the reference segment.

Lower performance is especially pronounced for sentences exceeding 250 characters, underscoring the limitations of using the segmenters without capping.

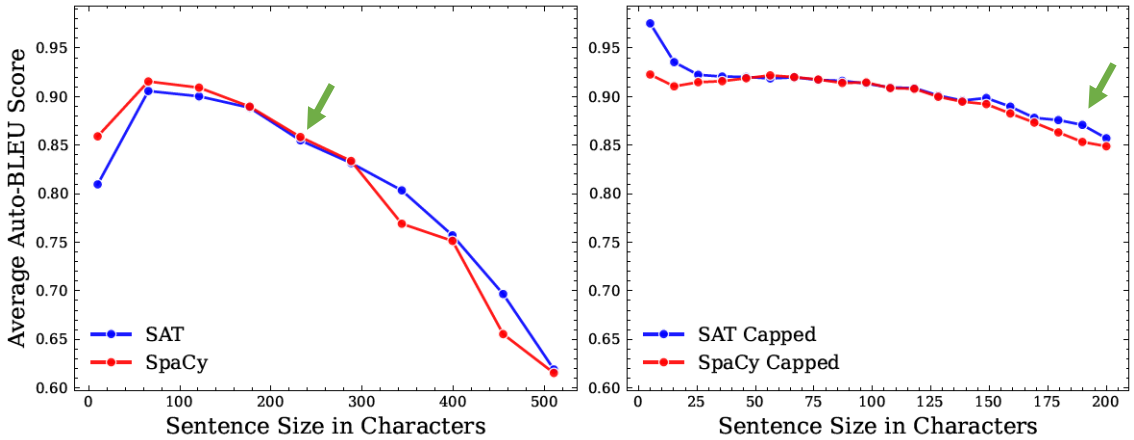


Figure 3 - **Segmenters quality**. Average Auto-BLEU scores for different sentence segmentation methods depending on sentence length, for both out of the box (left) and capped implementations (right).

- Prepare **the LCM training data with SaT Capped**.

Main Design Principles

2.3 Large Concept Model variants

❖ **LCM Design**

- Driven by **the need to conditionally generate a continuous sentence embedding**.
- Obviously contrasts with how current **LLMs** work, i.e., **estimating a probability distribution over a vocabulary of discrete tokens**.

➢ A straightforward way

- To train a transformer model to generate an embedding with the objective of minimizing the MSE loss (see Section 2.3.1).
- However, **a given context may have many plausible, yet semantically different, continuations**.

➢ The model should be able to **learn a conditional probability distribution over the continuous embedding of the next sentence**.

❖ **Motivate from Computer Vision tasks aiming to learn such conditional probability distributions over continuous data (Dhariwal and Nichol, 2021; Rombach et al., 2021).**

1) **Diffusion Models** like **Dall-E 3** (Betker et al., 2023) or **Imagen Video** (Ho et al., 2022)

- To generate an image or video from a text prompt.
- Many different real images may satisfy the same input prompt, hence the model has to **learn a probability distribution over continuous pixel data**.

➢ This **motivates the exploration of diffusion models for sentence embedding generation.**

➢ Sections 2.3.3 and 2.3.4

2) Prevalent take on **continuous data generation** consists of **quantizing** data to ultimately model with discrete units;

➢ Explore LCM modeling with quantization.

➢ Sections 2.3.5

Main Design Principles

2.3.1 Base-LCM

➢ **Baseline architecture for next-concept prediction**

- A standard **decoder-only Transformer** that transduces a sequence of preceding concepts (read sentence embeddings) into a sequence of future ones.

➢ Base-LCM is equipped with a "**PostNet**" and a "**PreNet**".

- The **PreNet** normalizes the input SONAR embeddings and maps them to the model's hidden dimension d_{model}.

$$\text{PreNet}(\mathbf{x}) = \text{normalize}(\mathbf{x})\mathbf{W}_{\text{pre}}^{t} + \mathbf{b}_{\text{pre}}, \quad (1)$$

$$\text{PostNet}(\mathbf{x}) = \text{denormalize}\left(\mathbf{x}\mathbf{W}_{\text{post}}^{t} + \mathbf{b}_{\text{post}}\right), \quad (2)$$

where $\mathbf{W}_{\text{pre}} \in \mathbb{R}^{\text{d}_{\text{model}} \times \text{d}_{\text{SONAR}}}$ and $\mathbf{b}_{\text{pre}} \in \mathbb{R}^{\text{d}_{\text{model}}}$.

$\mathbf{W}_{\text{post}} \in \mathbb{R}^{\text{d}_{\text{SONAR}} \times \text{d}_{\text{model}}}$, $\mathbf{b}_{\text{post}} \in \mathbb{R}^{\text{d}_{\text{SONAR}}}$

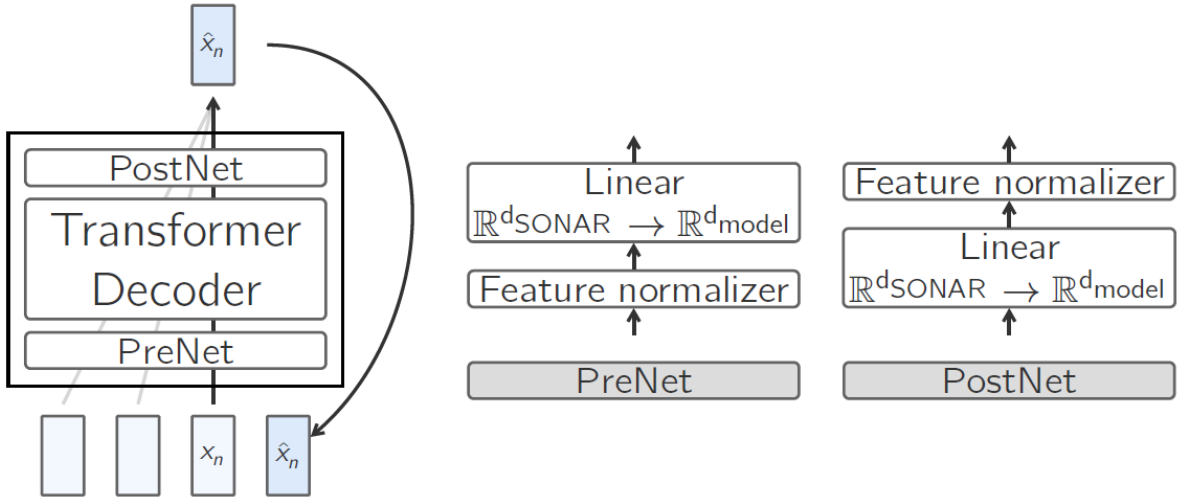


Figure 4 - TheBase-LCM. Illustration of the Base-LCM. At its core is a standard decoder-only Transformer surrounded with a PreNet and a PostNet

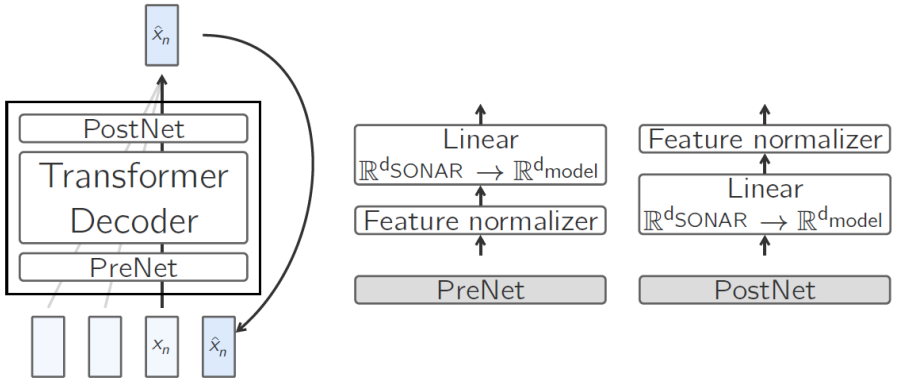
Main Design Principles

2.3.1 Base-LCM

➢ Base-LCM is equipped with a "**PostNet**" and a "**PreNet**".

- To learn the maps "**normalize**" and its inverse "**denormalize**", we fit **a robust scaler** to **a set of randomly sampled SONAR vectors** from **different corpora and domains of text data.**
  - ✓ This scaler removes the median statistics and scales the data according to the interquartile range (IQR).

$$\text{normalize}(\mathbf{x}) = \frac{\mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \quad \text{denormalize}(\mathbf{x}) = \boldsymbol{\mu} + \boldsymbol{\sigma}\mathbf{x}. \qquad (4)$$



➢ **Tain Base-LCM on the semi-supervised task of next concept prediction**

- The model predicts the next concept $\hat{\mathbf{x}}_n$ and its parameters θ are optimized to regress the ground truth next concept ($\mathbf{x}_n$).

$$\hat{\mathbf{x}}_n = f(\mathbf{x}_{<n}; \boldsymbol{\theta}), \quad \text{MSE}(\hat{\mathbf{x}}_n, \mathbf{x}_n) = \|\hat{\mathbf{x}}_n - \mathbf{x}_n\|^2. \qquad (5)$$

- Given a data distribution q of documents (sequences of concepts), the training loss is evaluated as

$$\mathcal{L}_{\text{BASE-LCM}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}\sim q}\left[\sum_{n=1}^{|\mathbf{x}|} \text{MSE}\left(f(\mathbf{x}_{<n}; \boldsymbol{\theta}), \mathbf{x}_n\right)\right]. \qquad (6)$$

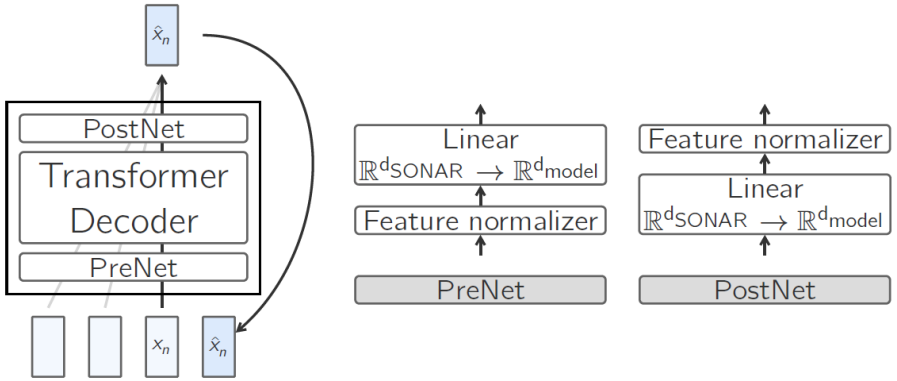
Main Design Principles

2.3.1 Base-LCM

➢ Variable length document

- For the generation of variable length documents at inference time, we suffix training documents with the sentence “End of text.”.
- This special suffix will be encoded with SONAR.

$$\mathbf{x}_{|\mathbf{x}|} = \overrightarrow{\text{eot}} := \text{encode}(\text{"End of text."})$$



➢ Implement two early stopping mechanisms during inference

(1) Measures the similarity of the generated embedding $\hat{\mathbf{x}}_n$ to $\overrightarrow{\text{eot}}$ and stops if the cosine similarity exceeds a threshold s_{eot}.

(2) Compares the newly generated embedding $\hat{\mathbf{x}}_n$ to the previous generation $\hat{\mathbf{x}}_{n-1}$ and stops if their cosine similarity is higher than a threshold s_{prev}. We set both s_{eot} and s_{prev} to 0.9.

2.3.2 Diffusion-based LCM

➢ **Diffusion-based LCMs**

- Generative **latent variable models** that **learn** **a model distribution p_θ** approximating **a data distribution q**.
- Similar to the Base-LCM, we model **the diffusion LCMs** as **auto-regressive models** that **generate concepts in a document**, one at a time.
- **The model distribution** is expressed at each position *n* of the sequence as $p_\theta(\mathbf{x}_n|\mathbf{x}_{<n})$ i.e., the generation of the next concept is conditioned on the preceding context.
- Use a superscript for the denoising/diffusion step ($t \in [0,1]$) and a subscript (n) for indexing the sequence of concepts.
- Simplify for a given n the conditional model distribution $p_\theta(\mathbf{x}_n^0|\mathbf{x}_{<n}^0)$ as $p_\theta(\mathbf{x}_0)$, and the conditional data distribution $q(\mathbf{x}_n^0|\mathbf{x}_{<n}^0)$ as $q(\mathbf{x}_0)$.
- Diffusion models involve two processes: a ***forward noising process*** and a ***reverse denoising process*** (Ho et al., 2020; Song et al., 2020):

Main Design Principles

2.3.2 Diffusion-based LCM

1) Forward process and noise schedule

- The forward process is a ***Gaussian diffusion process*** characterized by the marginal distribution $q(\mathbf{x}^t|\mathbf{x}^0)$, given for every timestep $t \in [0,1]$ as:

$$\mathrm{q}(\mathbf{x}^t \mid \mathbf{x}^0) \coloneqq \mathcal{N}(\alpha_t \mathbf{x}^0, \sigma_t^2 \mathbf{I}). \tag{7}$$

- With the *reparameterization trick*, we can sample from this marginal distribution via:

$$\mathbf{x}^t = \alpha_t \mathbf{x}^0 + \sigma_t \boldsymbol{\epsilon} \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{8}$$

- We use a ***variance-preserving forward process*** (Karras et al., 2022) for which we have:

$$\alpha_t^2 = \mathrm{sigmoid}(\lambda_t), \;\; \sigma_t^2 = \mathrm{sigmoid}(-\lambda_t) = 1 - \mathrm{sigmoid}(\lambda_t), \tag{9}$$
$$\lambda_t = \log\left(\alpha_t^2 / \sigma_t^2\right),$$

λ_t is the log signal-to-noise ratio (log-SNR) for timestep t.

- The **noise schedule** is a strictly monotonically decreasing function f_λ that maps from the timestep $t \in [0,1]$ to a log-SNR level: $\lambda_t = f_\lambda(t)$
- Previous work defines the noise schedule based on a **discrete variance schedule** $(\beta_0, \cdots, \beta_T)$
- This stems from *the formulation of the forward process* as ***a discrete-time Markov chain*** that gradually adds Gaussian noise to the data *according to variance schedule*:

$$\mathrm{q}(\mathbf{x}^{1\ldots\mathrm{T}}|\mathbf{x}^0) \coloneqq \prod_{t=1}^{\mathrm{T}} \mathrm{q}(\mathbf{x}^t|\mathbf{x}^{t-1}),$$
$$\mathrm{q}(\mathbf{x}^t|\mathbf{x}^{t-1}) \coloneqq \mathcal{N}(\mathbf{x}^t; \sqrt{1-\beta_t}\mathbf{x}^{t-1}, \beta_t \mathbf{I}), \tag{10}$$

- From the variance schedule $(\beta_t)_t$, the **noise schedule** can be expressed as:

$$\alpha_t^2 = \prod_{s=1}^{t} (1 - \beta_s). \tag{11}$$

- $\mathbf{x}^t$ is the short for $\mathbf{x}^{t/T}$

Main Design Principles

2.3.2 Diffusion-based LCM

➢ **Noise schedule**

- Following Kingma and Gao (2024), for any given noise schedule, we visualize **the distribution over noise levels $p(\lambda) = -dt/d\lambda$** in order to characterize how much time we are spending at every noise level during training.

- Consider three types of noise schedules

Cosine $$\alpha_t^2 = f(t)/f(0), \text{where } f(t) = \cos^2\left(\frac{t+s}{1+s}\cdot\frac{\pi}{2}\right), \quad (12)$$

$$\text{where } s = 0.008.$$

Quadratic. $$\beta_{t/\mathrm{T}} = \left(\sqrt{\beta_0} + \frac{t}{\mathrm{T}}\cdot\left(\sqrt{\beta_1} - \sqrt{\beta_0}\right)\right)^2. \quad (13)$$

Variances $(\beta_t)_t$ are set to constants increasing quadratically from β_0 to β_1

- In all experiments, we rescale the variance schedule $(\beta_0, \cdots, \beta_T)$ to enforce zero terminal SNR i.e., $\beta_T = 1$.

Sigmoid. $$\alpha_t^2 = f(t)/f(0), \text{ where } f(t) = \text{sigmoid}\left(\delta - \gamma \,\text{logit}(t)\right), \quad (14)$$

$$\text{sigmoid}\, x \mapsto e^x/(e^x+1) \quad \text{logit} : x \mapsto \log(x/(1-x))$$

Sigmoid schedule as a means to study the impact of the SNR distribution on the training of our models

γ controls the scale of the log-SNR distribution $p(\lambda)$ and δ its center (see Figure 5).

2.3.2 Diffusion-based LCM

2) Reverse process and objective function

- The joint distribution of the diffusion model $p_\theta(\mathbf{x}^{0\cdots 1})$ is called the ***reverse process*** and is defined as a Markov chain with learned Gaussian transitions starting at $p(\mathbf{x}^1) = N(\mathbf{0}, \mathbf{I})$.

$$\mathrm{p}_{\boldsymbol{\theta}}(\mathbf{x}^{0:\mathrm{T}}) := \mathrm{p}(\mathbf{x}^{\mathrm{T}}) \prod_{t=1}^{\mathrm{T}} \mathrm{p}_{\boldsymbol{\theta}}(\mathbf{x}^{t-1}|\mathbf{x}^t), \tag{15}$$

$$\mathrm{p}_{\boldsymbol{\theta}}(\mathbf{x}^{t-1}|\mathbf{x}^t) := \mathcal{N}(\mathbf{x}^{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}^t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}^t, t)),$$

- ✓ $\boldsymbol{\mu}_\theta$ and Σ_θ are predicted statistics.
- ✓ Σ_θ is set to the constant $\sigma_t^2\mathbf{I}$ (matching the transitions of the forward process).
- ✓ $\boldsymbol{\mu}_\theta$ can be decomposed into a linear combination of $\mathbf{x}^{t-1}$ and a noise approximation model ϵ_θ.
- ✓ This prediction method is dubbed ϵ-prediction (Ho et al., 2020; Nichol and Dhariwal, 2021; Nichol et al., 2022).

- We adopt $\mathbf{x}^0$-prediction i.e., we predict the noiseless state and optimize the simple reconstruction loss

$$\mathcal{L}(\boldsymbol{\theta}) := \mathbb{E}_{t\sim\mathcal{U}(0,1)}\left[\omega(t)\mathcal{L}(t,\boldsymbol{\theta})\right], \tag{16}$$

$$\mathcal{L}(t,\boldsymbol{\theta}) := \mathbb{E}_{\mathbf{x}^0,\boldsymbol{\epsilon}}\left[\left\|\mathbf{x}^0 - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\alpha_t\mathbf{x}^0 + \sigma_t\boldsymbol{\epsilon}, t)\right\|_2^2\right].$$

- Weighting strategies for the reconstruction loss; We default to the simple reconstruction loss (ω(t) = 1, ∀t) and we experiment with a clamped-SNR weighting strategy:

$$\omega(t) = \max(\min(\exp(\lambda_t), \lambda_{\max}), \lambda_{\min}),\ \lambda_t = \log(\alpha_t^2/\sigma_t^2), \tag{17}$$

Which is a generalization of Salimans and Ho (2022)'s truncated-SNR weighting and Hang et al. (2023)'s min-SNR strategy where the SNR is clamped between $\lambda_{\min}$ and $\lambda_{\max}$.

2.3.2 Diffusion-based LCM

2) Reverse process and objective function

- Additionally, we consider a weighting strategy that factors in the quality of the sample $\mathbf{x}^0$.
- We use as **sample weight** a scalar $\omega(\mathbf{x}^0) \in [0,1]$ correlated with the **sample's fragility score** i.e., **how easy it is to reconstruct a noised sample** (see Section 2.5.2).
- Fragile samples will be assigned a smaller weight and thus contribute less to the objective function.

$$\mathcal{L}_{\text{fragility}}(\boldsymbol{\theta}) := \mathbb{E}_{t\sim\mathcal{U}(0,1),\mathbf{x}^0,\boldsymbol{\epsilon}}\left[\omega(\mathbf{x}^0)\ \left\|\mathbf{x}^0 - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\alpha_t\mathbf{x}^0 + \sigma_t\boldsymbol{\epsilon}, t)\right\|_2^2\right], \quad (18)$$

$$\omega(\mathbf{x}^0) = \text{sigmoid}(a\ \ \text{fragility}(\mathbf{x}^0) + b), \quad (19)$$

a < 0 and b are hyper-parameters to tune

➢ Classifier-free diffusion guidance for the LCM

- Classifier-free diffusion guidance (Ho and Salimans, 2022) consists of **jointly training a conditional and an unconditional diffusion model.**
- The resulting **conditional and unconditional score estimates** are **combined** at inference time to achieve **a trade-off between sample quality and diversity.**

$$\nabla_x \log_\gamma p(x|y) = (1-\gamma)\nabla_x \log p(x) + \gamma\nabla_x \log p(x|y), \quad (20)$$

- y is the conditioning variable, in our case the sequence of preceding embeddings $(\mathbf{x}_1, \cdots, \mathbf{x}_{n-1})$ when denoising $\mathbf{x}_n$.
- γ controls the contribution of the conditional score;
- For $\gamma = 0$, this is equivalent to an unconditional model, and for $\gamma = 1$, it is a fully conditional model.
- In practice for vision models, γ is set to a value greater than 1, thus amplifying the signal from the conditioning model.

Main Design Principles

2.3.2 Diffusion-based LCM

➢ **Inference**

- At inference time, the reverse process is applied.
- $\mathbf{x}^T$ is obtained by sampling a random noise from $p(\mathbf{x}^T) = N(\mathbf{0}, \mathbf{I})$, and is then iteratively denoised by taking steps in the direction of the score function (i.e., the direction in which the log-likelihood increases fastest).
- Additional noise is added during the process in order to avoid falling down into modes of the distribution.
- Practically, we start from $\mathbf{x}^T \sim N(0, \sigma_{\text{init}}^2 \mathbf{I})$ and find that the quality of the sampled output is sensitive to the initial noise scale σ_{init}.
- During inference, we employ ***the classifier-free guidance rescaling technique*** of Lin et al. (2024) proven to *alleviate the image over-exposure problem* as the terminal SNR approaches zero.
- Denote with g_{scale} and $g_{rescale}$ the guidance scale and guidance rescale factors used at inference.

Sample steps

- Although we train the model on a large number of discretized timesteps, e.g., *T=100*, we only generate with a smaller number of steps, e.g., S=40, at inference via *accelerated generation processes* (Song et al., 2020).
- We select the sample steps following *the trailing method* of Lu et al. (2022) as it is found to be more efficient for **smaller steps S** (Lin et al., 2024).
- We generate along the sampled steps

  $(\tau_1, \cdots, \tau_S)$ = round(flip(arange(T, 0,−T/S))).

Main Design Principles

2.3.2 Diffusion-based LCM

- **Two variants of diffusion LCM: One-Tower and Two-Tower.**

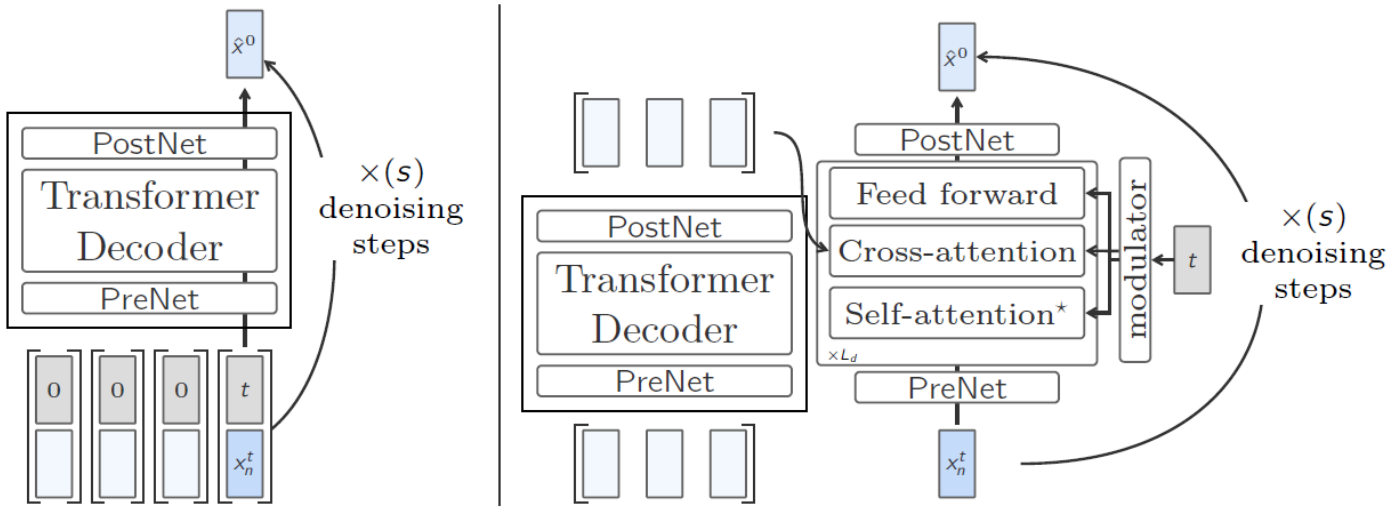


Figure 6 - Inference with diffusion-based LCMs. In the left-hand side, an illustration of the One-Tower LCM and on the right-hand side an illustration of the Two-Tower LCM.

Main Design Principles

2.3.3 One-Tower Diffusion LCM

- **A single transformer backbone** whose task is to **predict the clean next sentence embedding $\mathbf{x}_n^0$ given a noisy input $\mathbf{x}_n^t$, conditioned on previous clean sentence embeddings $\mathbf{x}_{<n}^0$.**

- **Each input embedding** is concatenated with **the corresponding diffusion timestep embedding. The learned position embeddings** are added to the input vectors prior to being fed to LCM. The backbone utilizes **a causal multi-head self-attention.**

- For efficient training, the model is trained to predict each and every sentence in a document at once.
- As depicted in Figure 7, during the diffusion process, the model attends to the clean sentences in the context using causal multi-head attention layers.
- **The input** is specially prepared by **interleaving the noisy (blue) and clean (light blue) sentence embeddings**, and **the attention mask** is prepared accordingly to **only attend to the clean sentence embeddings (gray arrows).**

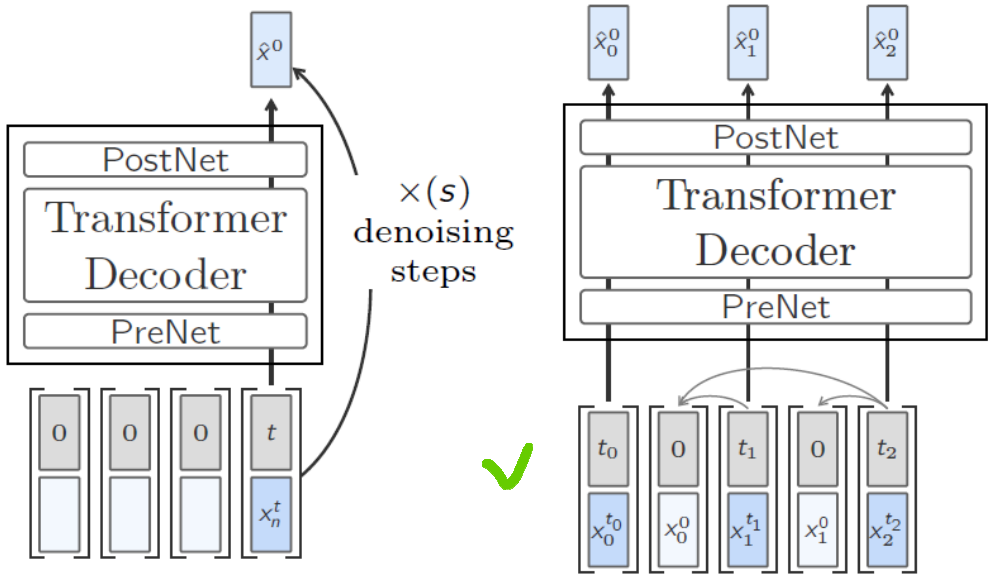


Figure 7 - **Training of One-Tower diffusion LCM**. Interleaving the clean and noisy embeddings and sampling different diffusion timesteps allows for efficient training.

- During training, self-attention can be dropped with a certain probability for unconditional training. This enables *classifier-free guidance* at inference time

Main Design Principles

2.3.4 Two-Tower Diffusion LCM

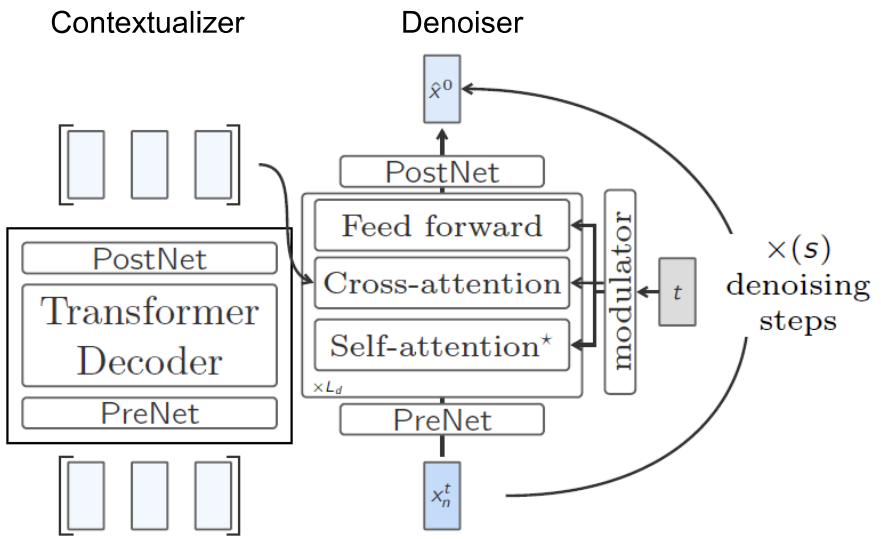
- Separates **the encoding of the preceding context (*contextualizer*)** from **the diffusion of the next embedding (*denoiser*)**

➢ **Contextualizer**

- Takes as input the context vectors $\mathbf{x}_{<n}$ and encodes them causally i.e., we apply a decoder-only Transformer with causal self-attention.

➢ **Denoiser**

- The outputs of the *contextualizer* are fed to a *denoiser*, which predicts the clean next sentence embedding $\mathbf{x}_n^0$ by iteratively denoising the latent $\mathbf{x}_n^1 \sim N(\mathbf{0}, \mathbf{I})$.
- The denoiser consists of a stack of Transformer blocks with *cross-attention block* to attend over the encoded context.
- Both the *denoiser* and the *contextualizer* share the same Transformer hidden dimension d_{model}.



Main Design Principles

2.3.4 Two-Tower Diffusion LCM

➢ **Denoiser**

- Each block of each Transformer layer in the denoiser (including the cross-attention layer) is modulated with *adaptive layer norm* (**AdaLN**, Perez et al. (2018); Peebles and Xie (2023)).
- The **AdaLN** modulator of Two-Tower regresses *channel-wise scale* (γ), *shift* (β) and *residual gates* (α) from the embedding of the current diffusion timestep t.

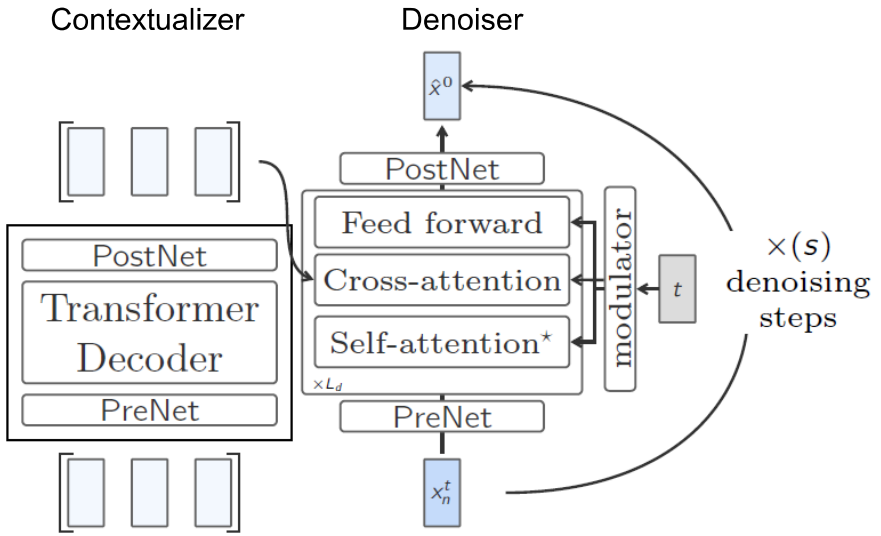
$$[\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\alpha}] = \mathrm{SiLU}(\mathrm{embed}(t))\mathbf{W}^t + \mathbf{b}, \quad (21)$$

$$\mathbf{y} = \mathbf{x} + \boldsymbol{\alpha}\ \mathrm{Block}((1+\boldsymbol{\gamma})\,\mathbf{x} + \boldsymbol{\beta}), \quad (22)$$

SiLU : Sigmoid Linear Unit function

$$\mathrm{silu}(x) = x * \sigma(x), \text{where } \sigma(x) \text{ is the logistic sigmoid.}$$

- We initialize each residual block in a Transformer layer (“Block”) with the identity function via initializing **W** and **b** in Eq. (21) to zero.



- **Diffusion timestep** *t* is embedded using a 256-dimensional frequency embedding followed by a two-layer MLP with SiLU as activation function. “*embed*” maps to the denoiser’s hidden dimension d_{model}.
- **The self-attention layers** in the *denoiser* do only attend to the current position i.e., we do not attend to the preceding noised context. *The self attention layers were kept for consistency with a standard Transformer block and for the possible extension of denoising multiple vectors at once*.

Main Design Principles

2.3.4 Two-Tower Diffusion LCM

➢ **Two-Tower training**

- At training time, Two-Tower's parameters are optimized for **the next-sentence prediction task** on *unsupervised sequences of embeddings*.
- The causal embeddings from the *contextualizer* are shifted by one position in the *denoiser* and **a causal mask** is used in its cross-attention layers.
- A **zero vector** is prepended to the context vectors to enable the prediction of the first position in the sequence (see Figure 8).
- To train the model both conditionally and unconditionally in preparation for inference with classifier-free guidance scaling, we **drop random rows from the cross-attention mask with a rate of p_{cfg}** and **denoise the corresponding positions with only the zero vector as context.**

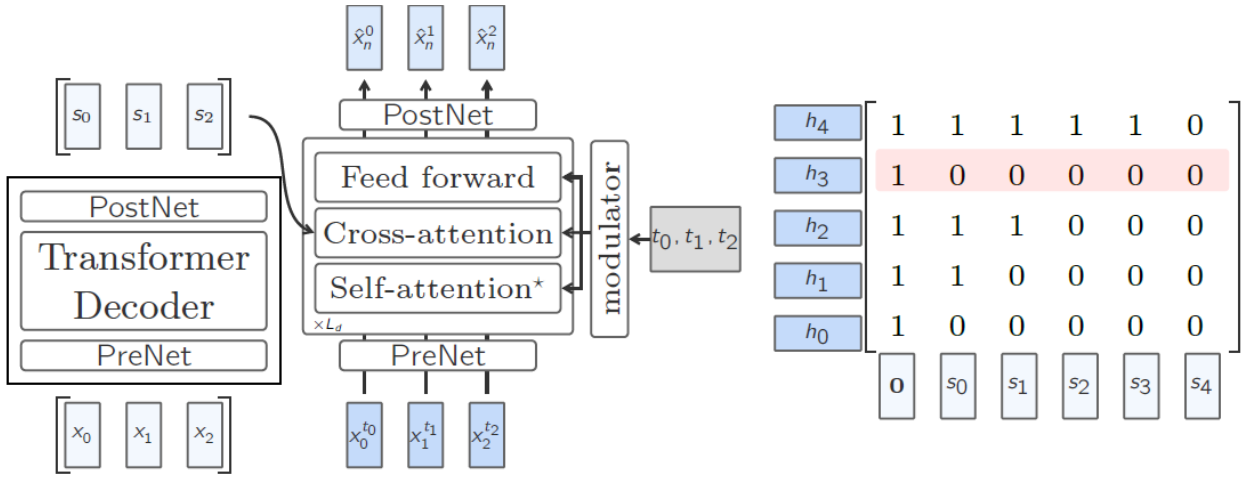


Figure 8 - **Training Two-Tower diffusion LCM.**

On the left panel, a Two-Tower forward pass in training time in order to denoise multiple embeddings in parallel.

On the right side panel a visualization of the denoiser's cross-attention masks with the red highlighted row signaling a sample dropped to train the denoiser unconditionally. $(h_1, \cdots, h_4)$ denotes the sequence of intermediate representations in the denoiser right before the cross-attention layer.

Main Design Principles

2.3.5 Quantized LCM

- Two major approaches to **deal with continuous data (image, speech) generation**
  - ✓ **Diffusion modeling**
  - ✓ **Learning quantization of the data** before modeling on top of these discrete units.
- Text modality remains discrete
  - ✓ Despite dealing with continuous representations in the SONAR space, **all possible text sentences** are **a cloud of points rather than a real continuous distribution in the SONAR space**.
- These considerations **motivate the exploration of quantization of SONAR representations** and then modeling on these discrete units to address the next sentence prediction task.
- Use of **temperature**, **top-p** or **top-k sampling**, to control the level of ***randomness*** and ***diversity*** in the sampling of the next sentence representation.
- We learn **residual quantizers for the SONAR space**, and then build a ***Quantized Large Concept Model*** based on these discrete units.
- We tried to come up with an architecture as close as the diffusion LCM models, to be able to compare approaches

Main Design Principles

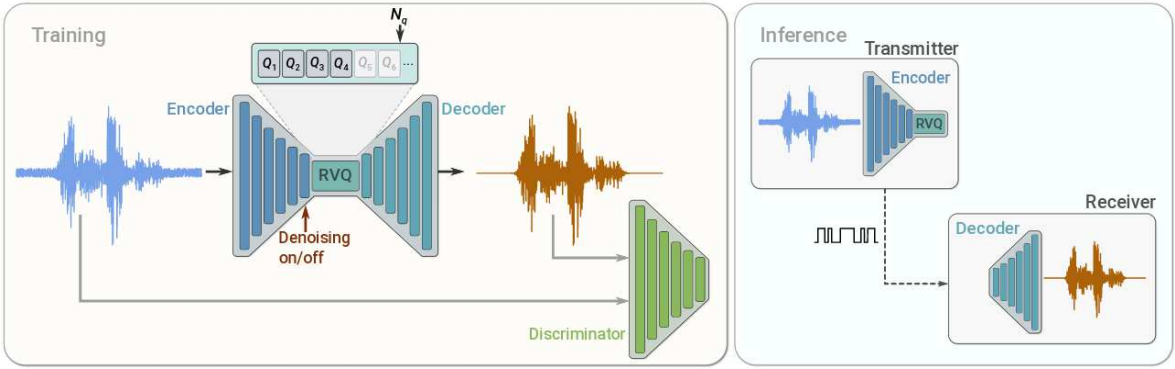
2.3.5 Quantized LCM

➢ **Quantization of SONAR space**

- We use **Residual Vector Quantization** (**RVQ**; Zeghidour et al.(2021)) as **a coarse-to-fine quantization technique to discretize SONAR representations.**
  - ✓ Vector quantization maps continuous input embeddings to the nearest entry in a learnt codebook.
  - ✓ RVQ **iteratively quantize residual errors from previous quantizations** using additional **codebook** for each iteration.

- We use **FAISS implementation** (Douze et al., 2024) which performs **iterative k-means clustering of residuals.**
- We use the **Improved Residual Vector Quantization (IRVQ)** method from Liu et al. (2015), with ***a beam size of 1*** for memory efficiency.
- We trained the RVQ codebooks on 15 million English sentences extracted from Common Crawl using $n_{codebooks}$ = 64 number of quantizers with $n_{units-per-codebook}$ = 8192 units per codebook.

Zeghidour et al., SoundStream: An End-to-End Neural Audio Codec, 2021



- Duration T를 가지고 f_s에서 sample 된 single channel recording $x \in \mathbb{R}^T$가 있다고 하자
  - SoundStream은 다음의 3가지 component로 구성됨
    1. **Encoder** : x를 embedding sequence에 mapping 하는 역할
    2. **Residual Vecotr Quantizer** : 각 embedding을 finite codebook set의 합으로 대체하여 representation을 target bit 수로 compressing 하는 역할
    3. **Decoder** : quantized embedding으로부터 lossy reconstruction $\hat{x} \in \mathbb{R}^T$를 생성하는 역할
  - 이때 SoundStream은 adversarial loss와 reconstruction loss를 결합한 loss를 통해 discriminator와 함께 end-to-end training 됨
  - 추가적으로 conditioning signal을 사용하여 denoising을 encoder-side/decoder-side에 적용하는 것을 결정할 수 있음

Main Design Principles

2.3.5 Quantized LCM

➢ **Quantization of SONAR space**

- One property of RVQ is that **the cumulative sum of centroid embeddings of the first codebooks** are **an intermediate coarse approximation of input SONAR vectors**.

- We can report **the evolution of auto-encoding BLEU scores** with **the increasing number of codebooks** used to quantize SONAR embeddings, before using the SONAR text decoder to decode quantized embeddings.

- **Auto-encoding BLEU consistently improves as the number of codebooks increases**, reaching around **70% of the auto-encoding BLEU score** achieved with **continuous SONAR embeddings**, when using **all 64 codebooks**.

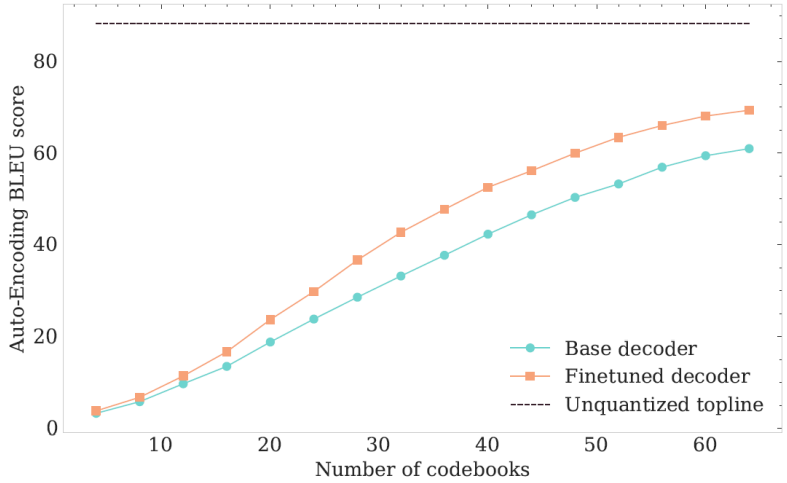


Figure 9 - Auto-encoding BLEU scores on FLORES devtest set, encoding sentences with SONAR encoder, quantizing with a varying number of codebooks, dequantizing and decoding with SONAR decoder.

BLEU(Bilingual Evaluation Understudy) : 기계 번역 결과와 사람이 직접 번역한 결과가 얼마나 유사한지 비교하여 번역에 대한 성능을 측정하는 방법. 측정 기준은 n-gram에 기반

자세한 설명은 다음 참조 : https://wikidocs.net/31695

$$BLEU = BP \times exp(\sum_{n=1}^{N} w_n \log p_n)$$

Brevity Penalty $$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

짧은 문장 길이에 대한 penalty

Main Design Principles

2.3.5 Quantized LCM

➢ **Finetuning the SONAR decoder on quantized representations**

- We fine-tuned SONAR decoder on quantized representations **to adjust it for the space created by the quantizers on 1.2M English sentences**.

- To make the **decoder** more **robust against residual representations from *intermediate codebooks***, we randomly select a codebook number $k \in [\frac{2}{3} \cdot n_{codebooks}, n_{codebooks}]$ during fine-tuning, with probability p = 0.3, and use the quantized representation with codebooks up to k.

➢ **Quant-LCM architecture**

- We aim at ***coarse-to-fine generation of SONAR embeddings conditioned on left-context sentences***.

- *An iterative generation of SONAR embeddings based on **intermediate quantized representation**.*

  ✓ The Quant-LCM model starts with *the intermediate representation* as ***a vector filled with zeros***.

  ✓ We iteratively add to this *intermediate representation* **the predicted residual centroid embeddings**.

  ✓ In that way, the predicted SONAR embeddings are iteratively refined based on **the growing cumulative sum of centroid embeddings of first codebooks**, until all codebooks have been seen.

- We used the One-Tower architecture for Quant-LCM experiments even though it could be trained with Two-Tower architecture too.

- **Noisy input representations** are replaced with **intermediate quantized representations** and **diffusion timestep embeddings as input** are replaced by **codebook index embeddings**.

Main Design Principles

2.3.5 Quantized LCM

➢ **Discrete targets**

- Following previous work on modeling discrete units from residual quantizers (Wang et al., 2023; Rubenstein et al., 2023; Lee et al., 2022), a **Quant-LCM** can be **trained to predict the unit from the next codebook**, parameterized with a softmax output layer.

- For parameter efficiency,

✓ We do not use $n_{codebooks} \cdot n_{units-per-codebook}$ indices as discrete targets which would imply $n_{codebooks} \cdot n_{units-per-codebook}$ output dimensions,

✓ But only $n_{units-per-codebook}$ output dimensions while inputting the information of the codebook index to the model.

- **At training time**, similarly to diffusion LCM training,

✓ We randomly sample codebook index k between 1 and $n_{codebooks}$, and compute the cumulative sum of centroid embeddings of the first $k-1$ codebooks as input.

✓ We use the unit from codebook k of the target embedding as target index for cross entropy loss computation.

- **At inference time**,

✓ We iteratively predict the unit from the next codebook, get the corresponding centroid embedding and add it to the current intermediate representation as additional predicted residual embedding.

Main Design Principles

2.3.5 Quantized LCM

➢ **Discrete targets**

- We also enable ***classifier-free guidance on logits*** at inference time (Gafni et al., 2022) by randomly dropping left-context conditioning during training (in Section 2.3.3).
- This modeling approach with discrete targets is dubbed **Quant-LCM-d** in the following sections.
- The **improved SONAR decoder for quantized representations** is used to bridge the compression gap coming from SONAR quantization when using **Quant-LCM-d**.

➢ **Continuous targets**

- Explored a modeling approach that **predicts continuous target SONAR vectors** based on **left-context sentences** and **intermediate quantized representation of the target vector**, minimizing the Mean Squared Error between prediction and target embeddings.
- **At inference time**,

✓ We can either iteratively add the closest centroid embedding based on the predicted residual $\hat{\boldsymbol{r}}$ or sample a centroid $\boldsymbol{c_i}$ from the following distribution:

$$\mathrm{p}(\mathbf{c}_i|\hat{\mathbf{r}}) = \frac{e^{-\beta \cdot \|\mathbf{c}_i - \hat{\mathbf{r}}\|_2}}{\sum_k e^{-\beta \cdot \|\mathbf{c}_k - \hat{\mathbf{r}}\|_2}}, \tag{23}$$

β is a temperature hyper-parameter

- This modeling approach with continuous targets is denoted with **Quant-LCM-c**

Main Design Principles

2.4 Ablation Study

❖ We compare all the variants of LCMs introduced above, namely, Base-LCM, One-Tower, Two-Tower and Quant-LCM.

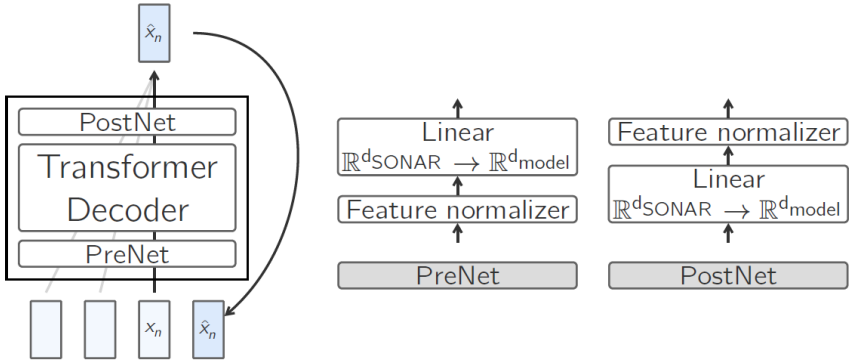
2.4.1 Experimental setup

- Pre-train our models on the **Fineweb-edu** dataset (Lozhkov et al., 2024).
- All models are configured to have approximately **1.6B** trainable parameters and are pre-trained on **Meta's Research Super Cluster** (**RSC**, Lee and Sengupta (2022)) for **250k optimization steps spanning 32 A100 GPUs** with **a total batch size of 229k concepts**

➢ **Models architectures**

1) **Base-LCM**

- Has **32 layers** and a model dimension d_{model} **= 2048** with **16 attention heads**.
- Uses **rotary position embeddings** (RoPE, Su et al. (2024)), applies **pre-normalization** using **RMSNorm** (Zhang and Sennrich, 2019), uses the **SwiGLU** activation function (Shazeer, 2020) and is trained with a dropout rate of p=0.1.



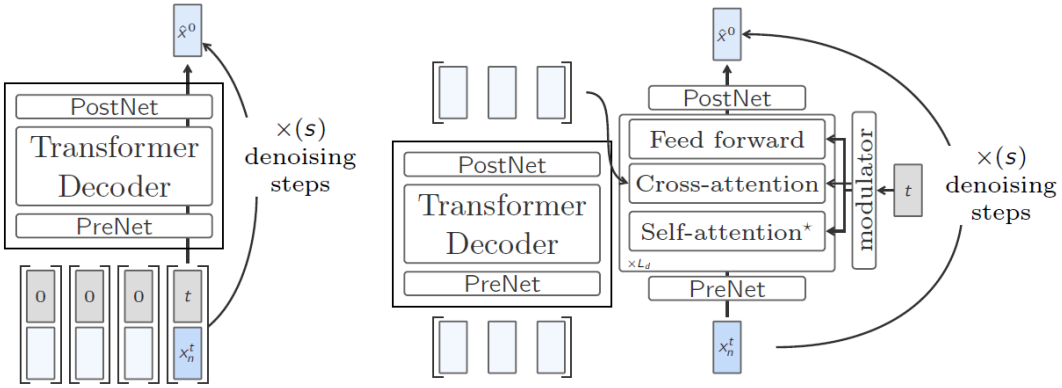
Main Design Principles

2.4.1 Experimental setup

➢ **Models architectures**

2) **One-Tower diffusion LCM**

- **32 transformer blocks**, each made of a self-attention layer with **32 attention heads** and followed by **a feed-forward neural network** with **inner size 8192**.
- d_{model} **= 2048** and uses **learned position embeddings**.
- The **noise scheduler** is set with **T=100 diffusion timesteps**.
- During training, self-attention is dropped with p = 0.15 for unconditional training, enabling classifier-free guidance at inference time.



3) **Two-Tower diffusion LCM**

- **5 layers in *contextualizer*** and **13 layers in *denoiser***.
- Similar to the Base-LCM, it has **16 attention heads**, d_{model} **= 2048** , and uses **SwiGLU** activations and **RMSNorm** in both *contextualizer* and *denoiser*.
- ***Contextualizer*** uses **RoPE** for embedding positions whereas ***Denoiser*** is **without positional embeddings**.
- We use by default *the cosine noise schedule* with T=100 and train with a dropout rate of p=0.1.
- For training the model *unconditionally*, we use a **cross-attention mask dropout of rate 0.15** (see Section 2.3.4).
- The pre-training documents are wrapped at 128 sentences.
- Unless otherwise mentioned we decode with S=40 sample steps with a guidance scale g_{sacle} = 3, a guidance rescaling factor of $g_{resacle}$ = 0.7, an initial noise scale σ_{init} = 0.6 and epsilon-scaling with λ_{eps} = 1.00045.

2.4.1 Experimental setup

➢ **Models architectures**

4) **Quant-LCM**

- Exactly the same architecture as the One-Tower diffusion LCM, except for **Quant-LCM-d** which differs only by its output dimension $n_{units-per-codebook}$ = 8192 for softmax computation.
- For single sentence prediction tasks,
  - ✓ t_{opk} = 1, g_{sacle} = 2 for Quant-LCM-d
  - ✓ t_{opk} = 1, g_{sacle} = 3 for Quant-LCM-c
- For multi-sentence generation tasks
  - ✓ temperature 1, t_{opk} = 3, g_{sacle} = 1 Quant-LCM-d
  - ✓ temperature 0.005, t_{opk} = 5, g_{sacle} = 1.5 for Quant-LCM-c,
  - ✓ As higher guidance or lower temperature setups led to repeated generated sentences.

Main Design Principles

2.4.1 Experimental setup

➢ **Pre-training evaluation**

- **Pre-trained token-level language models** are typically evaluated with ***perplexity***: a measure of how well each next token is predicted given a *teacher-forced* (i.e., ground truth) prefix of the document.

- Each pre-trained model is initially evaluated on the quality of its predicted next sentence $\hat{\mathbf{x}}_n$ given a ground truth context $\mathbf{x}_{<n}$.
- Practically, for a given document $\mathbf{x}_{1:n}$, we run the LCM inference in teacher-forcing mode and evaluate the following metrics:

❖ **L2 distance** (l_2) : Euclidean distance in the **SONAR space** between the predicted embedding $\hat{\mathbf{x}}_n$ and the ground truth continuation $\mathbf{x}_n$

$$\ell_2 := \|\hat{\mathbf{x}}_n - \mathbf{x}_n\|^2$$

❖ **Round-trip L2 distance** (l_{2-r}) : Euclidean distance in the **SONAR space** between the re-encoded sentence generated from the predicted embedding and the ground truth continuation $\mathbf{x}_n$

$$\ell_{2\text{-r}} := \|\,\text{encode}(\text{decode}(\hat{\mathbf{x}}_n)) - \mathbf{x}_n\|^2.$$

✓ Since an LCM can predict an embedding outside of the distribution of real embeddings, the SONAR decoder might shift these embeddings to the nearest *plausible* embeddings subspace.

✓ $\boldsymbol{l_{2-r}}$ **metric** is introduced to capture the shift in embeddings after decoding them into text then re-embedding them again in the SONAR space.

Main Design Principles

2.4.1 Experimental setup

➢ **Pre-training evaluation**

❖ **Contrastive accuracy (CA)** : The ratio of embeddings in a batch that are further away from the predicted embedding $\hat{\mathbf{x}}_n$ than the ground truth $\mathbf{x}_n$

- This metric assigns higher penalty for large l_2 values in the regions with high density of sentence embeddings.

❖ **Paraphrasing (PAR)** : The maximum cosine similarity (CS) between the generated embedding $\hat{\mathbf{x}}_n$ and the context embeddings $\mathbf{x}_{<n}$, normalized by the score of the ground truth sentence.

$$\mathrm{PAR} = \max_{m<n} \mathrm{CS}(\hat{\mathbf{x}}_n, \mathbf{x}_m) / \max_{m<n} \mathrm{CS}(\mathbf{x}_n, \mathbf{x}_m)$$

✓ The goal of this metric is to capture if the model is simply copying or paraphrasing a sentence from the context more (> 1) or less (< 1) than the reference sentence.

❖ **Mutual information (MI) :** This metric of **text coherence** evaluates the ***mutual information*** between the next predicted sentence $\hat{\mathbf{s}}_n = decode(\hat{\mathbf{x}}_n)$ and the previous k=10 ground truth sentences by computing the difference between the unconditional perplexity of $\hat{s}_n$ and its perplexity conditioned on the prompt:

$$\mathrm{MI} = \frac{1}{|\hat{\mathbf{s}}_n|} \left(\log \mathrm{p}_{\mathrm{LM}}(\hat{s}_n) - \log \mathrm{p}_{\mathrm{LM}}(\hat{\mathbf{s}}_n | \mathbf{s}_{n-k:n-1})\right)$$

✓ We estimate the *perplexity* with a small language model, GPT-2 (Radford et al., 2019).

✓ We prepend a newline symbol to $\hat{\mathbf{s}}_n$, so that a probability could be assigned to its first token, and we compute the average mutual information per-token by normalizing it with $|\hat{\mathbf{s}}_n|$, the length of $\hat{\mathbf{s}}_n$ in tokens.

✓ When averaging MI over a dataset,$|\hat{\mathbf{s}}_n|$ are used as weights

Main Design Principles

2.4.1 Experimental setup

➢ **Pre-training evaluation data**

- Perform the pre-training evaluation on sampled subsets from four corpora covering different domains: **ROC-stories** (Mostafazadeh et al., 2016), **C4** (Raffel et al.,2019), **Wikipedia-en** (English Wikipedia dump) and **Gutenberg**.
- We sample two distinct subsets (***dev*** and ***test***) from each corpus, we use the ***dev*** split for **tuning inference hyper-parameters** and report the results on the ***test*** splits.

Dataset	#Docs	#Llama2 Tokens			#Sentences			Total sentences
		Q1	Q2	Q3	Q1	Q2	Q3	
ROC-stories (dev)	2000	48	57	64	5	5	5	10K
ROC-stories (test)	1871	50	56	62	5	5	5	9.4K
C4 (dev)	1000	136	288	577	6	12	24	20.6K
C4 (test)	1000	133	282	599	6	11	25	21.9K
Wikipedia-en (dev)	1000	146	332	736	5	10	23	21.1K
Wikipedia-en (test)	1000	147	312	673	5	9	21	19.1K
Gutenberg (dev)	55	10297	15934	22259	328	530	687	216.2K
Gutenberg (test)	61	10752	15204	23414	325	457	735	562.2K

Table 2 - **Statistics of the pre-training evaluation datasets**. For each subset we report the number of documents, the total number of sentences and document lengths quartiles in sentences and in Llama2 tokens for reference.

Main Design Principles

2.4.1 Experimental setup

➢ **Pre-training evaluation data**

- **BASE-LCM**

✓ Substantially lower l_2 score; since Base-LCM effectively optimizes l_2 score during training. Yet, l_{2-r} score is not improved compared to other models

✓ When many plausible next sentence continuations are possible, Base-LCM generates their average in SONAR space (instead of sampling one of plausible modes) which may not correspond to any relevant point in SONAR space.

✓ So, highlighted by the poor Base-LCM performance in term of CA and MI scores.

- No significant difference in **CA scores** between diffusion LCMs and Quant-LCM variants
- **MI scores** are consistently higher for **diffusion-based models** compared to Quant-LCM.

- To tackle the next sentence prediction task in the SONAR space, **diffusion-based methods** give clearly better results compared to all other models.

Model	ROC-STORIES					C4				
	ℓ_2	$\ell_{2\text{-r}}$	PAR	CA	MI	ℓ_2	$\ell_{2\text{-r}}$	PAR	CA	MI
BASE-LCM	0.177	0.237	1.847	72.4%	0.062	0.204	0.261	1.964	69.1%	-0.105
ONE-TOWER	0.236	0.236	1.939	80.2%	0.977	0.279	0.273	2.239	77.1%	1.110
TWO-TOWER	0.233	0.231	2.088	80.6%	1.137	0.265	0.261	2.265	75.4%	1.134
QUANT-LCM-C	0.236	0.237	1.683	76.0%	0.610	0.279	0.283	2.013	77.2%	0.715
QUANT-LCM-D	0.240	0.246	1.871	81.1%	0.682	0.270	0.282	1.808	75.0%	0.359

Model	WIKIPEDIA-EN					GUTENBERG				
	ℓ_2	$\ell_{2\text{-r}}$	PAR	CA	MI	ℓ_2	$\ell_{2\text{-r}}$	PAR	CA	MI
BASE-LCM	0.229	0.283	1.770	69.6%	0.071	0.207	0.264	1.780	67.8%	-0.184
ONE-TOWER	0.324	0.311	2.087	80.9%	1.202	0.284	0.281	2.051	75.1%	0.725
TWO-TOWER	0.307	0.297	2.079	78.8%	1.307	0.267	0.267	2.077	73.0%	0.684
QUANT-LCM-C	0.306	0.317	1.842	79.5%	0.744	0.269	0.281	1.774	72.1%	0.419
QUANT-LCM-D	0.295	0.311	1.592	76.0%	0.323	0.276	0.290	1.599	72.0%	0.153

Table 3 - **Comparing architectures**. Pre-training evaluation results on the four select corpora. For each subset, we report l_2 (L2 distance in SONAR space), l_{2-r} (round-trip L2 distance after decoding and re-encoding the generated embeddings), PAR (similarity to preceding embeddings) and CA (contrastive accuracy)

Main Design Principles

2.4.1 Experimental setup

➢ **Instruction-tuning evaluation**

- The **pre-trained models** are in**struction-tuned on the stories subset of Cosmopedia** (Ben Allal et al., 2024).
- Aim with this finetuning to evaluate the models to **follow instructions and generate consistent stories**.

- We trained a **small Llama** (Touvron et al., 2023) on the same training data (Fineweb-edu) and finetuned it on **Cosmopedia**.
  - ✓ 24 transformer layers, each with 16 attention heads and a model dimension of 2048 for a total of 1.4B parameters.
  - ✓ Referred to as **smaLlama**.

❖ 전체적으로 Diffusion 기반 LCM(특히 Two-Tower) 이 문맥적 일관성, 생성 품질, 그리고 다양한 노이즈 처리 능력에서 가장 뛰어난 성능을 발휘함을 명확히 보여줌. Quant-LCM은 메모리 효율성과 실용성 면에서 여전히 유리하지만, 성능 면에서는 Diffusion 기반 모델에 비해 다소 저조함
https://discuss.pytorch.kr/t/lcm-large-concept-models-meta-ai/5744

- Evaluation metrics

✓ **ROUGE-L (R-L)** : ROUGE-L (F-measure) (Lin, 2004) between the generated and reference stories

✓ **Coherence (Coherence)** : Computed with a bidirectional transformer model fine-tuned by Jwalapuram et al. (2022) to assign higher scores to positive "natural" documents than to negative examples with permuted sentences. We normalize it with a sigmoid

§ Scores of "*certainly incoherent*" documents close to 0 and those of "*certainly coherent*" documents close to 1

Model	R-L ↑	Coherence ↑
BASE-LCM	23.69	0.482
ONE-TOWER	33.40	0.968
TWO-TOWER	33.64	0.938
QUANT-LCM-C	30.87	0.847
QUANT-LCM-D	28.01	0.704
SMALLAMA	34.88	0.984

Table 4 - **Comparing architectures**. Instruction-tuning evaluation results. For each model we score the generated stories on the held-out test prompts and report R-L (ROUGE-L) scores.

Main Design Principles

2.4.2 Importance of the diffusion inference hyper-parameters

➢ **Effect of different inference hyper-parameters on the quality of the generated text.**

- Generate outputs for the **C4** test split with **the Two-Tower LCM mode**, while varying the following hyper-parameters: the guidance scale g_{scale}, the initial noise scale σ_{init}, and the number of inference sample steps S.

- As increase g_{scale}, the mutual information (MI) between the prefix and the generated suffix increases, and so does paraphrasing as we are paying more attention to the conditioning context variables.

- The ℓ2 distance from the ground truth continuation increases as the model prefers to stay close to the prefix.

- Regarding σ_{init}, values between 0.5 and 0.7 achieve the best MI score. The generated individual sentences are usually longer with a higher σ_{init}. The ℓ2 distance on the other hand does not reflect this trend.

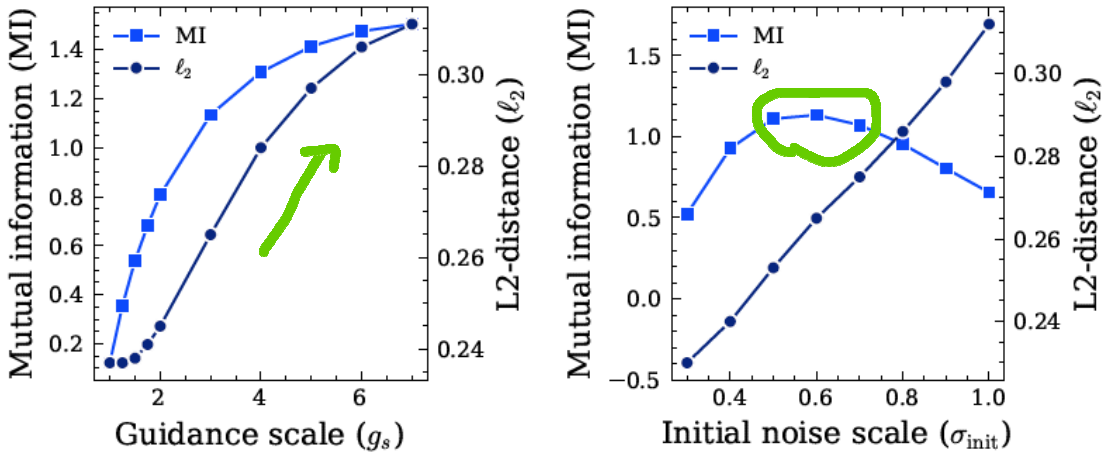


Figure 10 - **Importance of inference hyper-parameters**.

The first panel shows the quality of the generated output measured with MI and ℓ2 as we vary the guidance scale g_{scale} with fixed σ_{init} = 0.6, S = 40.

The second panel varies the initial noise scale σ_{init} with fixed g_{scale} = 3, S = 40.

2.4.2 Importance of the diffusion inference hyper-parameters

➢ **Effect of different inference hyper-parameters on the quality of the generated text.**

- With more inference steps, improve the prefix-suffix mutual information,
- But there is diminishing returns to increasing the inference cost with little qualitative improvement.

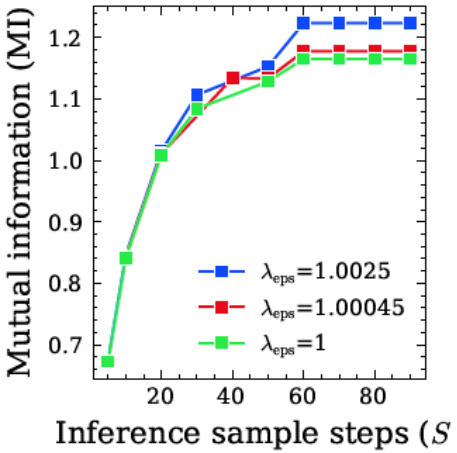


Figure 10 - **Importance of inference hyper-parameters**.

The third panel varies the inference steps S while holding g_{scale} = 1.5, σ_{init} = 0.6 fixed. We consider 3 values for λ_{eps} to see the impact of epsilon-scaling in the regime of large inference steps.

Main Design Principles

2.4.3 Studying the noise schedules

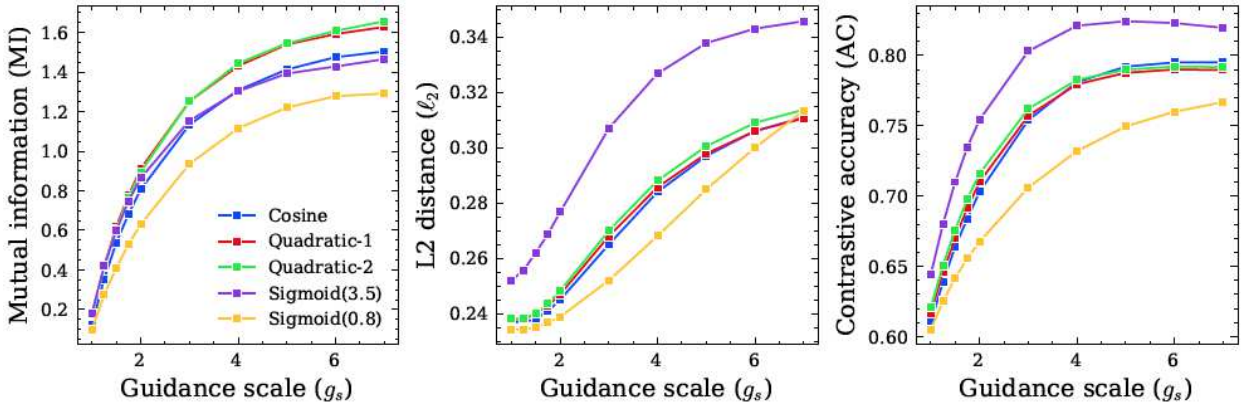
- **Compare different Two-Tower diffusion LCMs trained with different noise schedules**

- **Cosine** : Our default cosine noise schedule.
- **Quadratic** : The first quadratic schedule (Quadratic-1) has β_0 = 0.001 and β_T = 0.0012, whereas the second quadratic schedule (Quadratic-2) has β_0 = 0.02 and β_T = 0.022.
- **Sigmoid** : 4 sigmoid schedules with $(\alpha, \beta) \in \{(1.5,-1), (1.5,-2), (0.8,-1), (3.5, 0)\}$.

Table 5 - **Comparing noise schedules**. Results of the pre-training evaluation on two corpora, C4 and Wikipedia-en.

Model	C4					Wikipedia-en				
	ℓ_2	$\ell_{2\text{-r}}$	PAR	CA	MI	ℓ_2	$\ell_{2\text{-r}}$	PAR	CA	MI
Cosine	0.265	0.261	2.265	75.4%	1.134	0.307	0.297	2.079	78.8%	1.307
Quadratic-1	0.268	0.264	2.341	75.7%	**1.252**	0.309	0.300	2.202	79.1%	**1.409**
Quadratic-2	0.270	0.265	2.320	76.2%	**1.252**	0.312	0.303	2.185	79.7%	1.399
Sigmoid(1.5, -1)	0.257	0.259	2.226	74%	1.083	0.298	0.292	2.110	77%	1.271
Sigmoid(1.5, -2)	0.277	0.267	2.291	77.2%	1.173	0.321	0.303	2.179	80.3%	1.308
Sigmoid(0.8, -1)	0.252	0.255	2.053	70.6%	0.936	0.285	0.283	1.883	71.7%	1.127
Sigmoid(3.5, 0)	0.307	0.265	2.347	**80.3%**	1.154	0.347	0.303	2.187	**83.7%**	1.288

Figure 11 - **Comparing noise schedules**. The prefix-suffix mutual information (MI), the ℓ2 distance and contrastive accuracy (CA) scores of evaluated C4 documents while varying the guidance scale (g_{scale}) under different schedules



Scaling the model to 7B

➢ **Large 7B Two-Tower diffusion LCM**

- Scale Two-Tower diffusion LCM given its smaller memory footprint, particularly when processing long contexts with a shallower contextualizer tower
- Has 5 layers in its *contextualizer* and 14 layers in its *denoiser*.
- Extended to d_{model}=4096.
- Each self-attention layer has 32 attention heads.
- All other parameters are kept the same as for the 1.6B Two-Tower model.

● Pre-trained Model : Two-Tower-7B

- Pre-trained on a dataset of 2.3B documents, representing 2.7T tokens and 142.4B concepts/sentences
- On Meta's RSC for 124k optimization steps spanning 256 A100 GPUs with a total batch size of 1M concepts
- Extend the context length of this model to cover 2048 concepts instead of the 128 concepts in the ablation experiments
- Use AdamW optimizer with (β_1, β_2) = (0.9, 0.95), ϵ = 1e-5 and a weight decay of 0.1.
- Use a cosine learning rate schedule, with warm-up of 10,000 steps up to LR = 3e-4.
- Clip gradients at a maximum norm of g = 10.

● Finetuned Model : Two-Tower-7B-IT

- Finetune on publicly available instruction tuning datasets following Chung et al. (2024).
- Each sample consists of a prompt and an answer and we back-propagate on answer sentences only.
- Each answer sequence is suffixed with the phrase "End of response." to teach the model when to stop generating.
- Finetuning data totals 389M sentences, of which 53M are answers (i.e., targets).
- Use a cosine learning rate schedule with an initial rate of LR = 3e-5 and finetune the model for 7 epochs with a batch size of 262K sentences (prompts and answers combined).

Scaling the model to 7B

3.1 Evaluation Tasks and Data

3.1.1 Metrics

- Longform text generation is main challenge of LCM
- Evaluate them with multiple automatic metrics

Task	Area	Metric	Description	Reference
Summarization	Target similarity	R-L	ROUGE-L	Lin (2004)
	Source similarity	OVL-3	N-grams overlap (N=3)	
	Grammaticality	REP-4	Portion of duplicated N-grams (N=4)	Welleck et al. (2019)
	Fluency	CoLA	Sentence fluency classifier score	Krishna et al. (2020)
	Attribution	SH-4	Seahorse-Large-Q4 score	Clark et al. (2023)
	Semantic coverage	SH-5	Seahorse-Large-Q5 coverage score	Clark et al. (2023)
Summary Expansion	Grammaticality	REP-4	(see above)	Welleck et al. (2019)
	Fluency	CoLA	(see above)	Krishna et al. (2020)

Table 8 - Summary of automatic metrics used in different tasks in Section 3.1. Order mostly follows paper's narrative.

Scaling the model to 7B

3.1 Evaluation Tasks and Data

3.1.2 Summarization

➢ **Task and datasets**

- A summarization task : Generating a much shorter corresponding document that includes the essential information contained in the long document and the same logical structure linking the various pieces of essential information.
- Summarization techniques can range from more **extractive** to more **abstractive**. We focuses more on **abstractive summarization**, as such type of summarization cannot be performed without some form of ***understanding*** and ***reasoning***.

❖ CNN DailyMail 데이터셋은 CNN의 기사를 기반으로 기사의 본문과 함께 기사의 주요 요약(Highlights)을 제공하는 데이터셋으로, 본문을 기반으로 추출적 요약(Extractive Summarization) 성능을 측정함.

❖ XSum 데이터셋은 BBC 뉴스 기사 데이터를 기반으로 본문과 함께 단일 문장으로 구성된 요약(Headline)을 제공하는 데이터셋으로, 본문 내용을 요약하는 추상적 요약(Abstractive Summarization) 성능을 측정함

- We use the **CNN DailyMail** (Hermann et al., 2015) and **XSum** (Narayan et al., 2018) datasets and report results on the challenging **LCFO corpus** which takes long documents as input, approx. 5k words (Costa-jussà et al., 2024).
- The task is to provide abstractive summaries with lengths representing 20%, 10%, and 5% of the input document

		#Llama2 Tokens			#Sentences		
Dataset	#Docs	Q1	Q2	Q3	Q1	Q2	Q3
CNN DailyMail	11.5k	605/61	892/78	1266/97	10/3	14/4	21/4
XSum	11.3k	273/25	445/30	735/35	25/1	30/1	35/1
LCFO.5%	249	6559/341	7214/378	7916/418	209/12	295/15	527/18
LCFO.10%	249	6559/654	7214/718	7916/796	209/22	295/27	527/32
LCFO.20%	249	6559/1276	7214/1403	7916/1524	209/41	295/48	527/59

Table 9 - Statistics of the test split of evaluation benchmarks. For each subset we report the number of documents and statistics of document and summary length in terms of sentences and Llama2 tokens. Each table cell shows “document/summary” length quartiles.

Scaling the model to 7B

3.1 Evaluation Tasks and Data

3.1.2 Summarization

➢ **Summarization results**

- For encoder-decoder transformer models, we use T5 (Raffel et al., 2020).
- For decoder-only LLMs, we choose Gemma-7B, Llama-3.1-8B and Mistral-7B-v0.3.
- Chose the published instruction-tuned models to compare with the LCM with the same training regime, and have a similar size (7B)

- 실험 결과 LCM은 기존 LLM보다 추상적 요약 생성 능력(낮은 OVL-3, REP-4)에서 강점을 보임.
- 문법적 유창성(CoLA) 및 원문 의미 충실성(SH-4, SH-5)에서 LLM보다 낮은 성능을 보였지만, 이는 LCM이 보다 독창적이고 추상적인 요약을 생성하는 데 초점을 맞추고 있음을 반영함

Table 9 - Statistics of the test split of evaluation benchmarks. For each subset we report the number of documents and statistics of document and summary length in terms of sentences and Llama2 tokens. Each table cell shows “document/summary” length quartiles.

Model	Paradigm	CNN DailyMail					
		R-L(↑)	OVL-3 (↑)	REP-4 (↓)	CoLA (↑)	SH-4 (↑)	SH-5 (↑)
Ground truth	—	100.00	0.170	0.684	0.850	0.683	0.586
T5-3B	SFT	**37.56**	0.174	0.854	0.946	0.773	0.503
Gemma-7B-IT	IFT	31.14	0.245	1.032	0.963	0.740	0.560
Mistral-7B-v0.3-IT	IFT	36.06	0.200	0.780	0.972	**0.780**	0.676
Llama-3.1-8B-IT	IFT	34.97	**0.248**	0.928	**0.973**	0.763	**0.692**
Two-Tower-7B-IT	IFT	**36.47**	0.177	**0.757**	0.767	0.723	0.459

Model	Paradigm	XSum					
		R-L(↑)	OVL-3 (↑)	REP-4 (↓)	CoLA (↑)	SH-4 (↑)	SH-5 (↑)
Ground truth	—	100.00	0.108	0.399	0.987	0.352	0.418
T5-3B	—	17.11	0.221	0.671	0.939	0.680	0.450
Gemma-7B-IT	IFT	18.20	0.177	0.620	0.769	0.546	0.446
Mistral-7B-v0.3-IT	IFT	21.22	0.162	0.480	0.922	0.633	0.621
Llama-3.1-8B-IT	IFT	20.35	**0.186**	0.501	**0.941**	**0.687**	**0.658**
Two-Tower-7B-IT	IFT	**23.71**	0.106	**0.464**	0.683	0.358	0.284

Scaling the model to 7B

3.1 Evaluation Tasks and Data

3.1.2 Summarization

➢ **Long-context summarization results**

- Long-context summarization (LCFO.5%, LCFO.10% and LCFO.20%)
- LCFO 데이터셋의 평균 문서 길이는 약 5,000단어로, 상당히 긴 텍스트를 포함하고 있음. 요약문은 입력 문서 길이의 20%, 10%, 5%로 제한된 길이를 가지도록 요구하고 있어, 긴 문서를 기반으로 얼마나 추상적인 요약(Abstractive Summarization)을 잘 생성하는지를 측정함
- Two-Tower-7B-IT 모델이 요청된 요약 길이에 가장 적합한 WR(Word Ratio) 점수를 기록하여, 긴 문서 요약에서도 안정적인 성능을 보임.
- 그 외 LLaMA-3.1-8B 모델도 R-L에서 높은 점수를 기록했지만, 이는 사전 학습 데이터와의 중복 가능성 때문으로 보임.
- 또한 Two-Tower 모델은 REP-4 점수가 가장 낮아 반복이 적은 요약을 생성함. 이는 다른 모델보다 깔끔한 요약을 생성하는 데 탁월함을 보여줌.

Table 11 - Performance on the long-context summarization task of LCFO. WR is the word count ratio between the generated text and the source document

Method	WR	LCFO.5%					
		R-L(↑)	OVL-3 (↑)	REP-4 (↓)	CoLA (↑)	SH-4 (↑)	SH-5 (↑)
Gemma-7B-IT	0.107	25.21	0.151	4.711	0.688	0.357	0.174
Mistral-7B-v0.3-IT	0.512	21.36	**0.532**	5.997	0.854	**0.656**	0.296
Llama-3.1-8B-IT	0.076	**37.67**	0.190	2.767	**0.931**	0.488	**0.314**
Two-Tower-7B-IT	0.060	26.88	0.162	**2.473**	0.796	0.628	0.196
		LCFO.10%					
		R-L(↑)	OVL-3 (↑)	REP-4 (↓)	CoLA (↑)	SH-4 (↑)	SH-5 (↑)
Gemma-7B-IT	0.150	29.25	0.164	6.427	0.667	0.377	0.194
Mistral-7B-v0.3-IT	0.549	25.00	**0.537**	6.289	0.848	**0.660**	0.306
Llama-3.1-8B-IT	0.128	**42.85**	0.243	3.804	**0.907**	0.486	**0.310**
Two-Tower-7B-IT	0.089	29.38	0.202	**3.00**	0.791	0.623	0.183
		LCFO.20%					
		R-L(↑)	OVL-3 (↑)	REP-4 (↓)	CoLA (↑)	SH-4 (↑)	SH-5 (↑)
Gemma-7B-IT	0.257	33.32	0.201	9.188	0.603	0.425	0.239
Mistral-7B-v0.3-IT	0.493	28.82	**0.527**	5.806	0.858	**0.658**	0.293
Llama-3.1-8B-IT	0.179	**46.92**	0.272	4.783	**0.888**	0.485	**0.315**
Two-Tower-7B-IT	0.140	31.74	0.253	**3.664**	0.779	0.613	0.187

LImitations

➢ **Key Limitations of the Large Concept Model**

1. Bias Toward Short Sentences
2. Limited Granularity Without Token-Level Refinement
3. Narrow Scope of Evaluation
4. Dependence on Pre-Trained SONAR Embeddings
5. Challenges with Long-Form Content
6. Multilingual Generalization Constraints

☞ Limitations of Large Concept Models (LCMs): Analysis and Recommendations for Improvement

https://dr-arsanjani.medium.com/limitations-of-large-concept-models-critical-analysis-and-recommendations-for-improvement-64318f9b428e

➢ **Read the paper**