RAG Road Map

Overview

> **Abstract**
>
> RAG: improve LLM text generation capability by combining its current with external knowledge.

> **Question**
>
> **LLM model often come with problem of losing context** when processing large document. Retrieval-Augmented Generation (RAG) was invented to tackle this problem by only retrieving most relevant contents as inputs for LLM instead of the whole documents.
>
> Thus RAG allow us to bypass LLM context window probelm.

> **Seealso**
>
> LLMs are like cars, if it stands in the middle of the deep forest we can point at it and laugh at how it's stupid and how it's better to just walk through the forest.
>
> RAG and tools (as in tool-calling for llms) are the infrastructure comparable to roads. Many people don't realize that once the "car" gets on the proper "road", it is all of sudden very efficient at what it does.
>
> We don't faster cars (e.g. GPT-5), infrastructure is all we need right now. e.g. DeepSeek R1 architecture, o1 architecture, etc...

Basically **RAG = Embedding Model (Encoder) + LLM**, to comsume contents RAG, I personally encourage you to learn the *basic of RAG from the top-down then understand the math from the ground-up*, this way you both understand why do we need RAG and what RAG comprise of both intuively (top-down) & logically (ground-up).

- ? In short, here what you need to learn. Note: `==` mean `require`.
  - **RAG == Embedding Model (Encoder) + LLM**. (This mean *RAG reuiqre you to learn*)
  - **Embedding Model (Encoder) + LLM == Sentence Transformer**
  - **Sentence Transformer == Deep Learning + Basic NLP (Tokenization + Embedding)**
- ! Why do I have to learn all these stuff Thanh ? Can't I just learn RAG from Top Down, the answer is YES but NO. :)) **You surely can understand RAG but that not the point, the point is HOW DO YOU USE RAG cause RAG a architecture, this mean you modify components inside RAG, therefor by learning Sentence Transformer you automatically learn RAG as well, in my opinion ;))**

Note:

- Top-Down: allow you to understand concept intuition, what it does, why we need it.
- Bottom-Up: allow you to understand how this concept is created.
- **Highly Recommend content is Highly Recommended ;))**

Highly Recommend Youtuber for understanding LLM : Umar Jamil

Prequisites for RAG (điều kiện, kiến thức tiên quyết để học RAG)

- MIT Intro to Deep Learning **(Highly Recommend) - Teach Deep Learning from the Top Down**, which **include**:
  - Gradient Descent
  - Activation Function
  - Neural Network
    - Neural Network Bottom Up (this course give you the best mathematical foundation of Neural Network, will take some time to finish)
  - RNN
    - RNN Fool Proof
    - RNN Bottom Up (hard to eat, but not impossible)
  - LSTM
    - RNN Intro + LSTM explain in detail
  - Attention Mechanism in a nutshell
  - Transformer in a nutshell
- Basic of NLP: Tokenization + Embedding (no link, i'm too lazy)
- Attention Mechanism (required before learning Transformer)
  - Attention Mechanism Top Down by Starquest
- **Transformer** (BOSS FIGHT)
- ? **Transformer along with Attention is quite tricky to understand**, so I'm **recommed you to learn them from 2 different viewpoints**. Both of these video teach Transformer from Top-Down to Bottom-Up.
  - Transformer by Umar Jamil
    - ? Explain Attention Mechanism + Transformer (Encoder only)
    - $ The best Transformer explaination I've seen. No further ado, HIGHLY RECOMMEND.
  - Transformer by Startquest (Encoder + Decoder)
    - ? Explain linearly, give detail & clear example but the example become quite messy as you go on.

    - $ Excellently Explain how Positional Encoding work. He definitly the GOAT.
- BERT by the Umar Jamil
    - $ The best BERT explaination I've seen

History of Sentence Transformer

- ? Explain the **evolution of Language Model** from **Tradition Method** using Statistic (TF-IDF) **to Modern Deep Learning method** like RNN to LSTM to B-LSTM (Bidirectional LSTM) to Transformer to BERT to SBERT (Sentence BERT).
    - Sentence Transformers Explain

RAG Explaind Intuitively & In detail

- Retrieval Augmented Generation (RAG) Explained: Embedding, Sentence BERT, Vector Database (HNSW)

Coding from ScratchTurtorial

- Self-Attention with Pytorch
- RAG Coding from Scratch (Coding Turtorial)