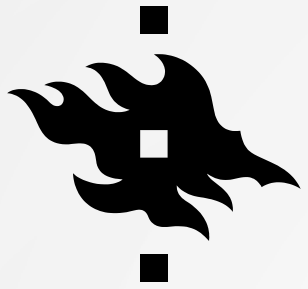




INTRODUCTION TO LLM & PROMPT ENGINEERING

Presented By
Sabina Zaman



REFERENCES

[1] Attention Is All You Need by Google

[2] PROMPT DESIGN AND ENGINEERING: INTRODUCTION AND
ADVANCED METHODS by Xavier Amatriain

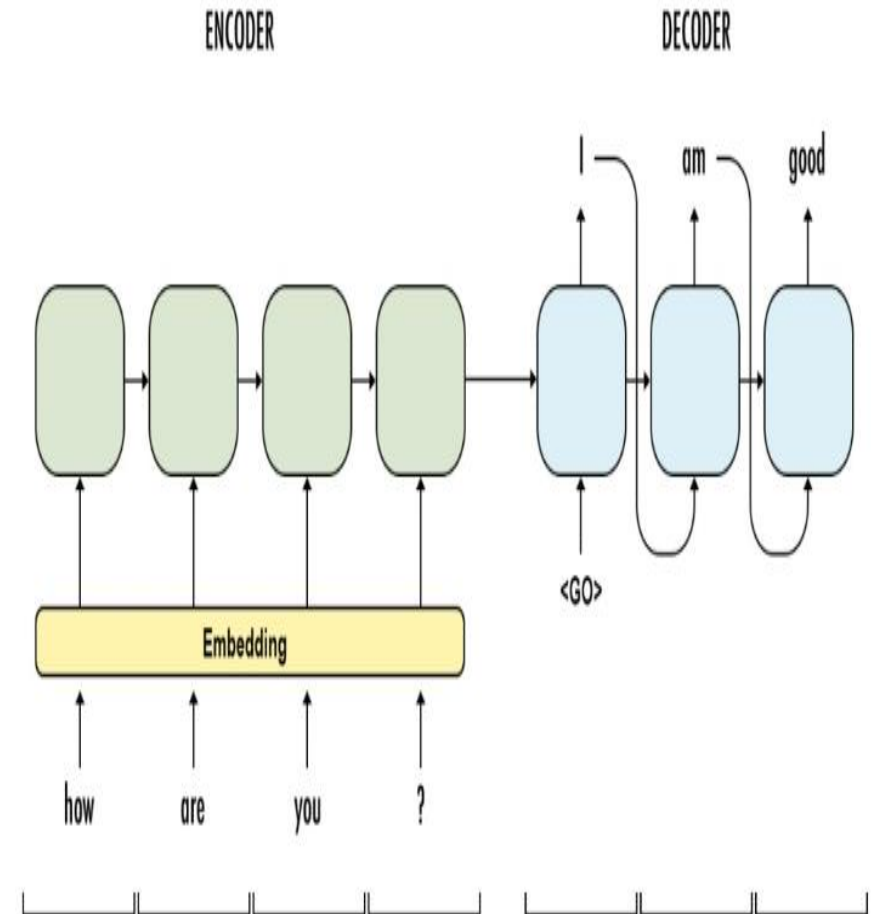
Images to explain the transformer architecture are collected from:

<https://www.youtube.com/watch?v=bCz4OMemCcA>



LSTM FOR NLP

- **Encoder and Decoder:** RNN network
 - Input gate – controls what information to add to the memory.
 - Forget gate – controls what information to discard from the memory.
 - Output gate – controls what information is used to produce the output.
- **Word/Sentence representation:** Matrix representation like TF-IDF, bag-of-words.
- **Word Embedding:** Lower dimensional representation of words.
- **Feed Encoder**
- **Connect Encoder & Decoder**
- **Decoder Teacher Forcing:** Each word in the decoder sentence is to predict the next word.*



*<https://www.ichaoran.com/posts/2019-01-03-build-a-simple-machine-translator-part1/>

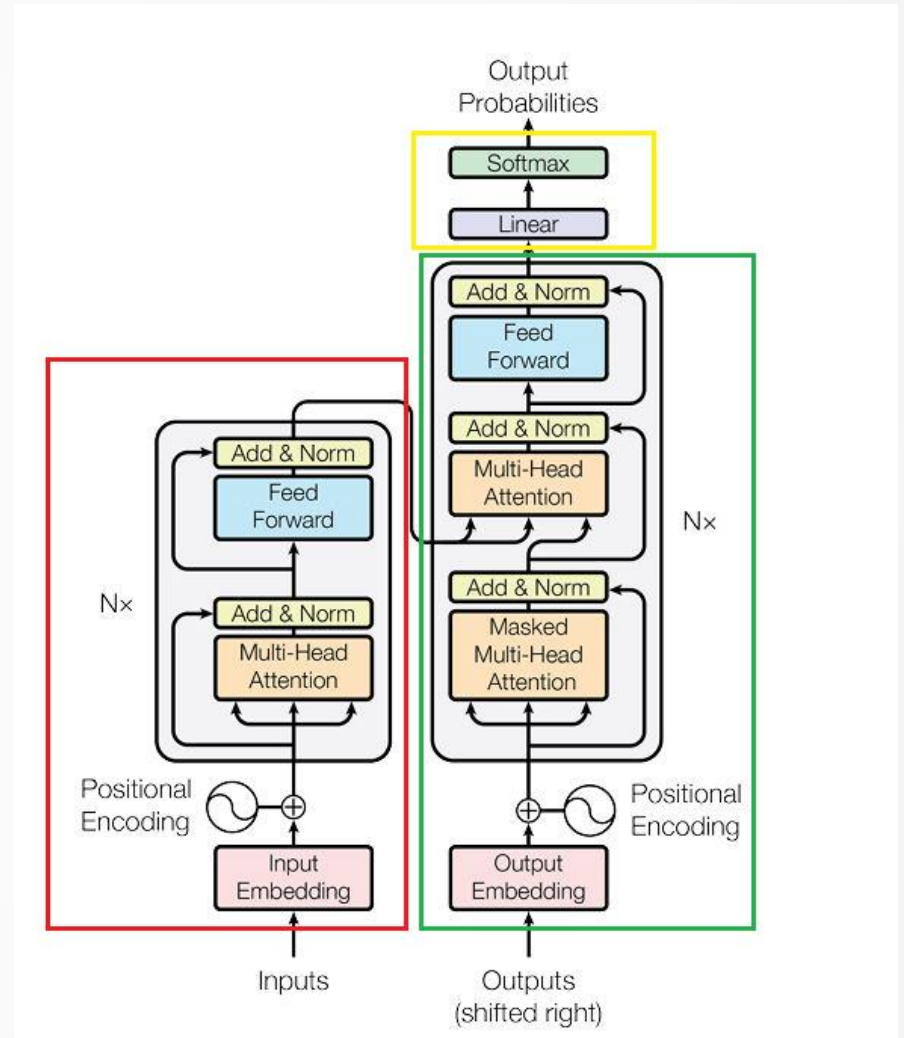


WE NEED ATTENTION!

Limitations of LSTM

- Slow computation for long sequence
- Vanishing or exploding gradients
- Long-range dependency
- Has always the last word and hidden state
- We can't impose **attention** or importance on any word.

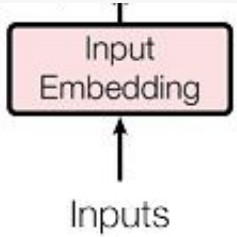
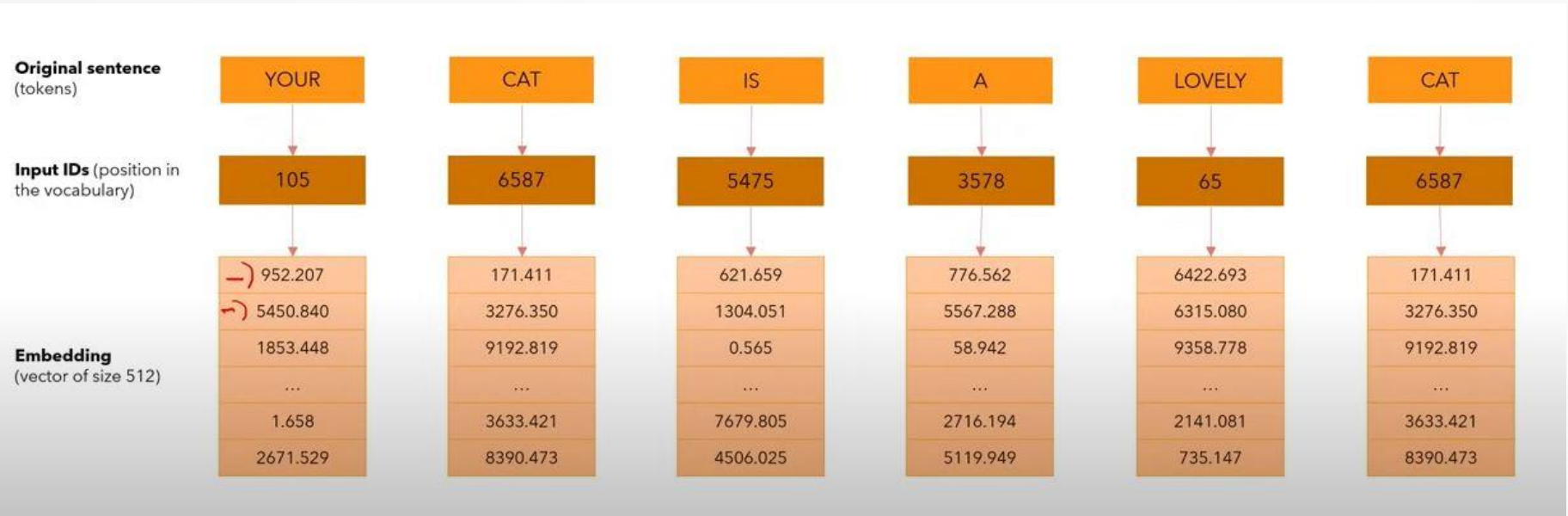
Attention: Where we need to look at!





INPUT EMBEDDING

d_{model} specifies the number of dimensions each token's embedding vector will have.
 $d_{model} = 4$ meaning each token is represented as a vector in a 4-dimensional space

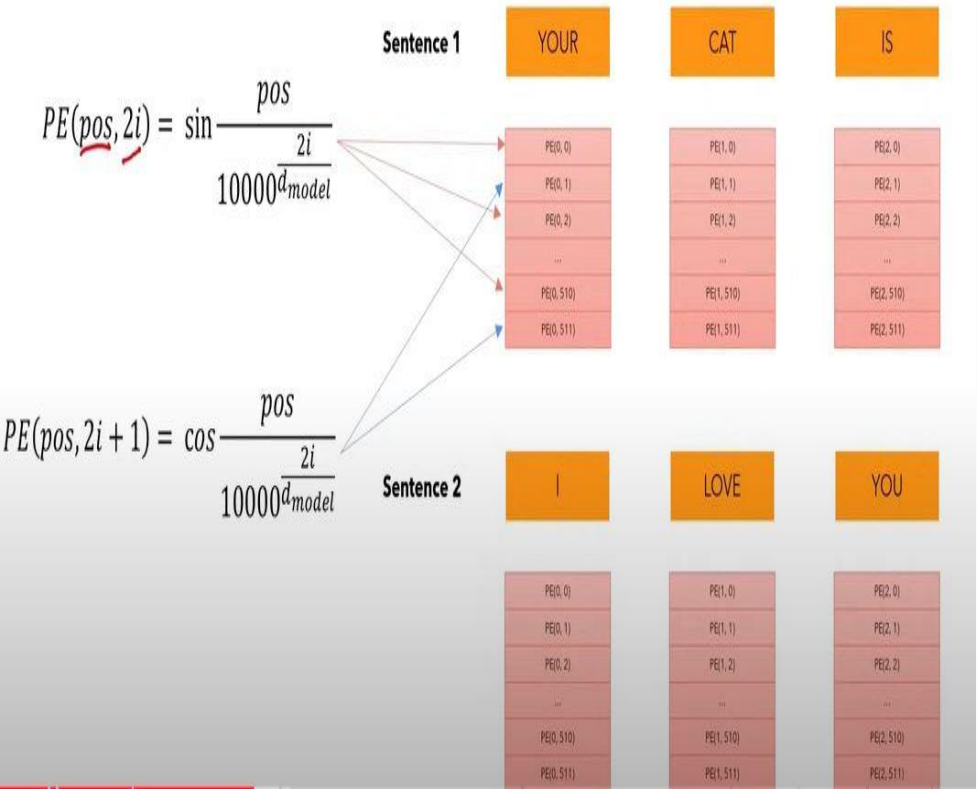
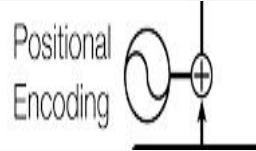


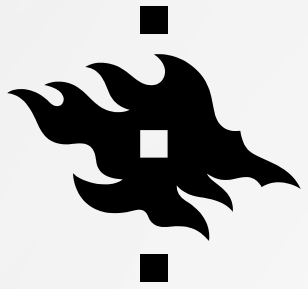
d_{model}

A	B	C
...		
512		



POSITIONAL ENCODING





SOFTMAX FUNCTION

1. **Input Vector:** Start with a vector z of raw scores (logits). For example:

$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

2. **Exponentiation:** Compute the exponential of each element in the vector:

$$e^{z_i} \text{ for each } z_i \in z$$

3. **Sum of Exponentials:** Calculate the sum of the exponentials:

$$S = \sum_j e^{z_j}$$

4. **Normalization:** Divide each exponential by the sum of exponentials to get the probabilities:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{S} \text{ for each } z_i \in z$$

5. **Output:** The result is a vector of probabilities that sum to 1:

$$\text{softmax}(z) = \begin{bmatrix} \text{softmax}(z_1) \\ \text{softmax}(z_2) \\ \text{softmax}(z_3) \end{bmatrix}$$

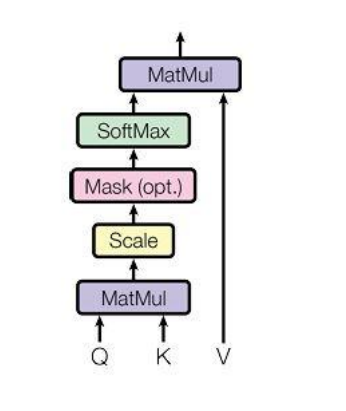
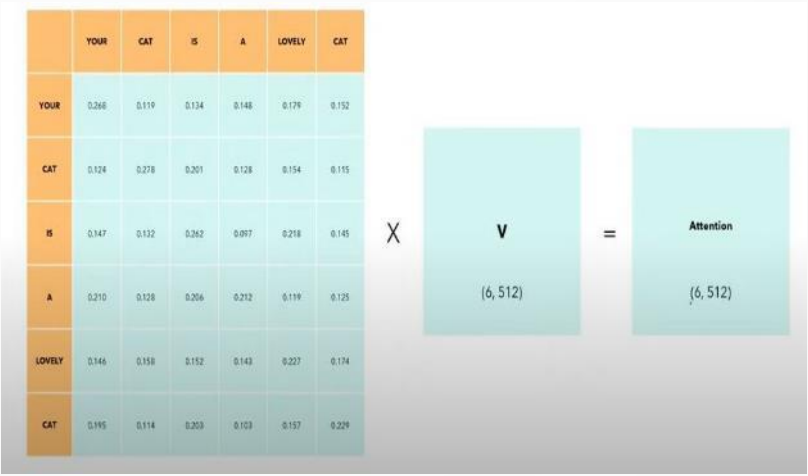
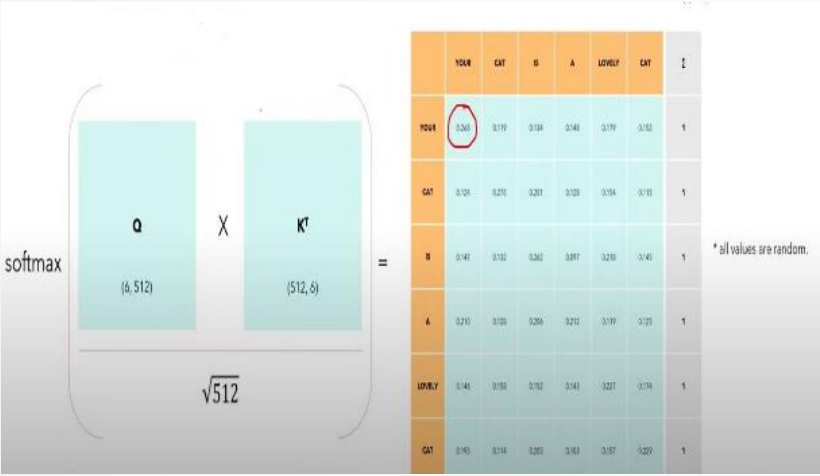


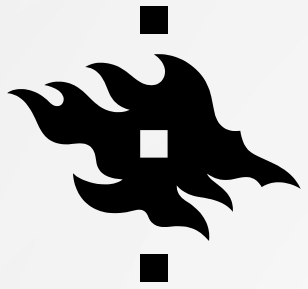
SELF ATTENTION

- Q -> We want to know about the input.
- K -> The input content that can be matched against the Queries.
- V -> The actual information that we want to retrieve
- W -> Weighted vectors we calculated on the training phase

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$

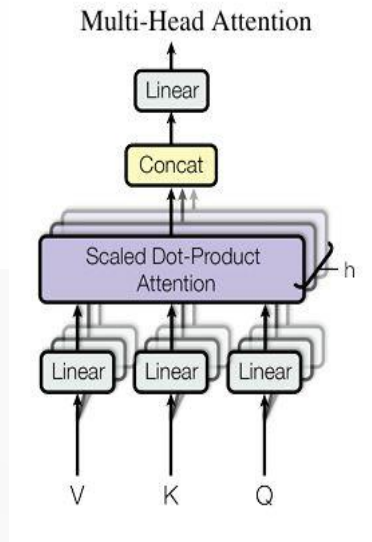
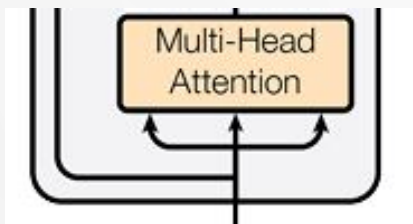
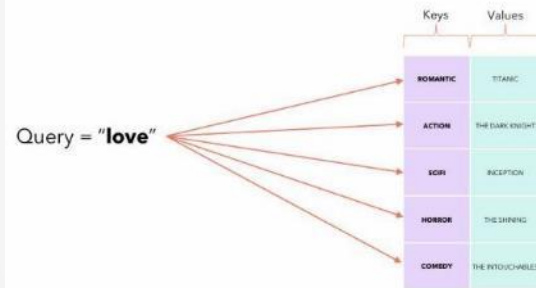




MULTI-HEAD ATTENTION

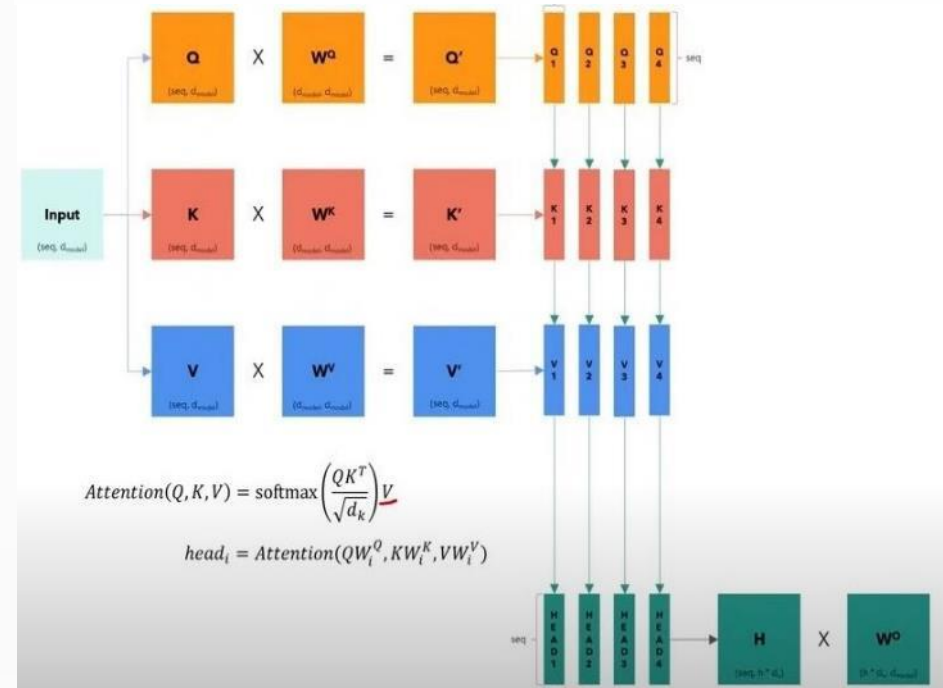
Each attention head focuses on different aspects.

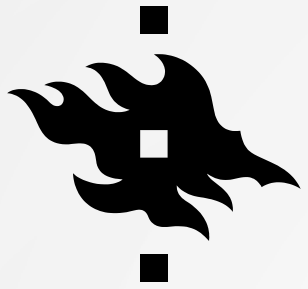
- One head might focus on syntactic relationships
- One might capture semantic relationships.



$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$MultiHead(Q, K, V) = \text{Concat}(\text{head}_1 \dots \text{head}_h)W^O$$
$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

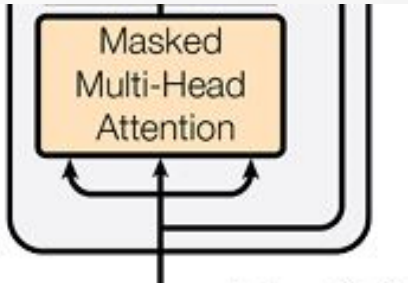




MASKED MULTI-HEAD ATTENTION

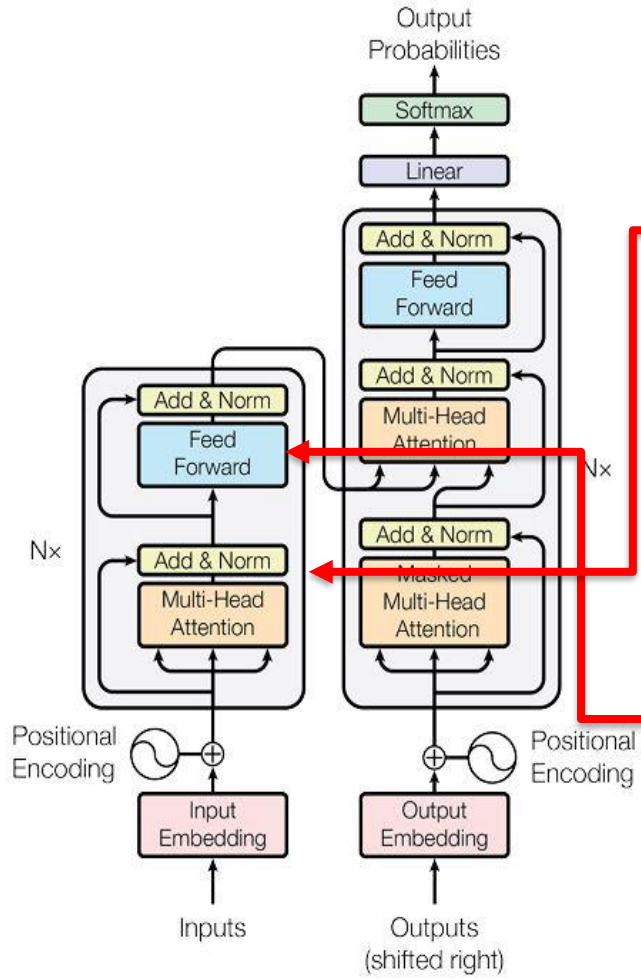
The output will have no relation with the future word/ words.

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229





FEED FORWARD NW ADD & NOM



$$\text{Output} = \text{Layer Output} + \text{Layer Input}$$

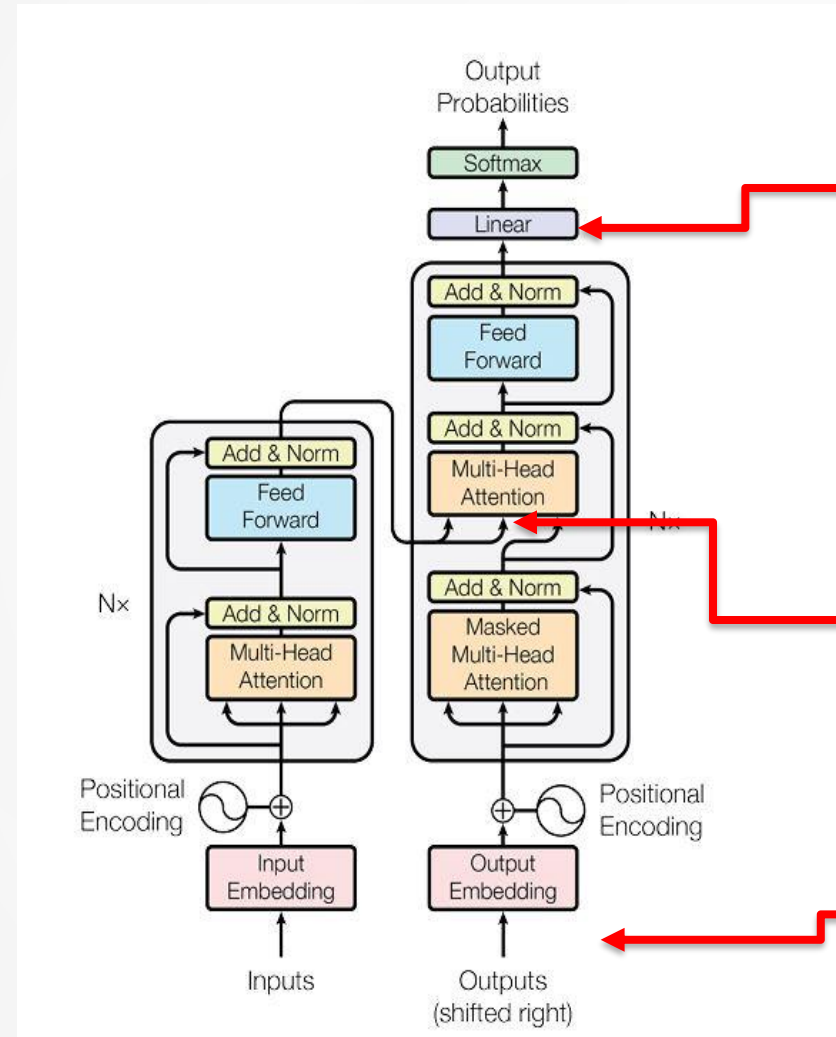
$$\text{LayerNorm}(x) = \frac{x - \mu}{\sigma + \epsilon}$$

Non-linear transformation to each token's embedding, enhancing the model's ability to capture complex patterns independently for each token

- First Linear Transformation
- Non-linear Activation
- Second Linear Transformation



OUTPUT ENCODING



Translates the high-dimensional representation into a probability distribution over all possible tokens in the vocabulary

- Keys (K): The decoder will use input representations to compare and decide which input parts are most relevant for generating the next word.
- Values (V): These hold the actual content of the input representation that the decoder can use to help generate the next word

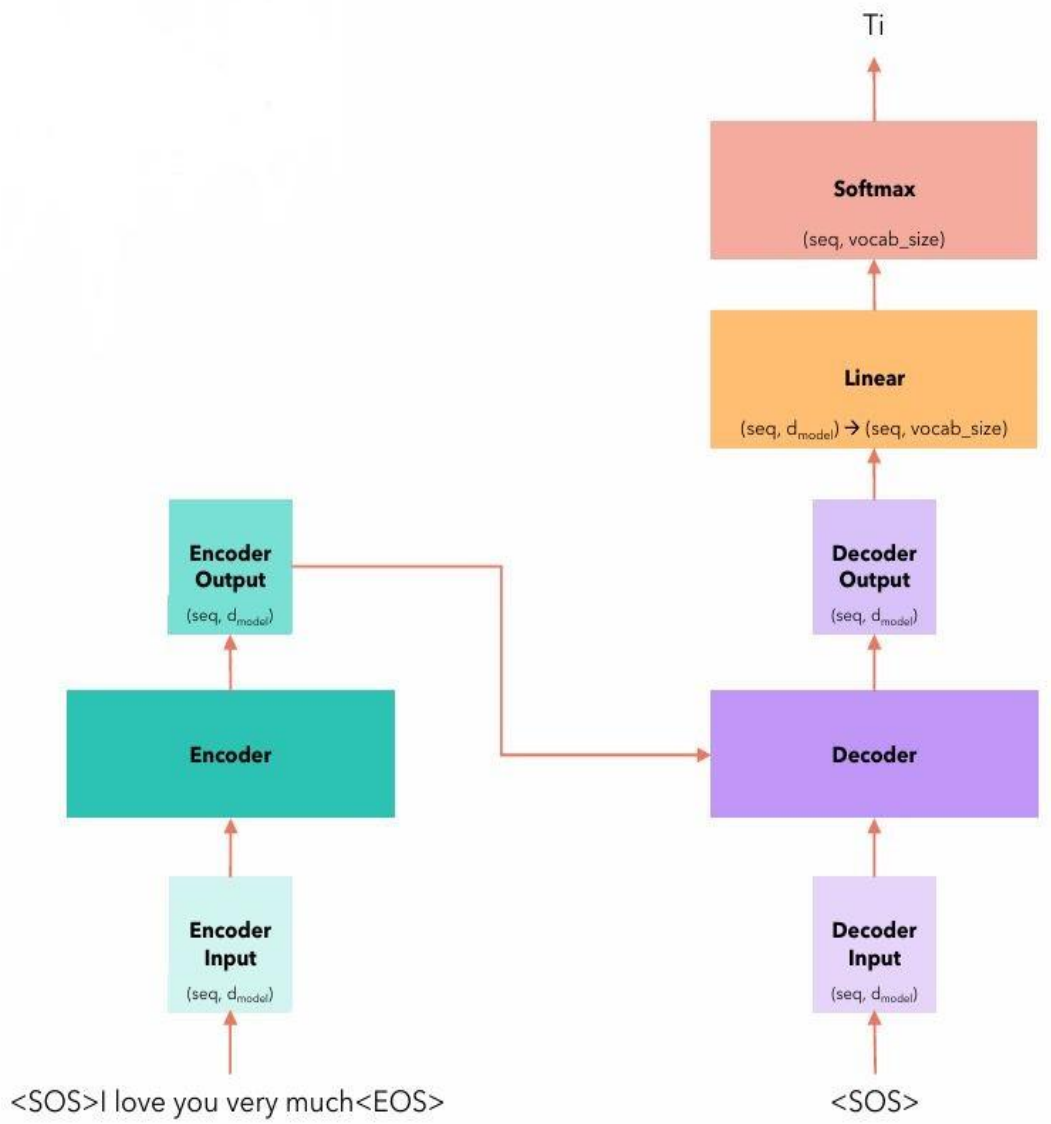
$$\text{Target} = [y_1, y_2, y_3, \dots, y_m]$$

input to the decoder is shifted to the right, like:

$$\text{Shifted Target} = [\text{<start>}, y_1, y_2, \dots, y_{m-1}]$$



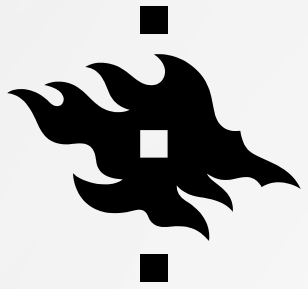
WORKFLOW OF TRANSFORMER





LIMITATIONS OF LLMS

- **Transient State:** LLMs inherently lack persistent memory or state
- **Probabilistic Nature:** The variability in responses, even to identical prompts
- **Outdated Information:** Reliance on historical knowledge, precluding real-time awareness or updates.
- **Content Fabrication:** LLMs may generate plausible yet factually incorrect information, referred to as "hallucination."
- **Resource Intensity:** LLMs require significant computational and financial costs, impacting scalability and accessibility.
- **Domain Specificity:** While inherently generalist, LLMs often require domain-specific data to excel in specialized tasks.



TRAINING AND PERFORMANCE

1. Benchmark Performance:

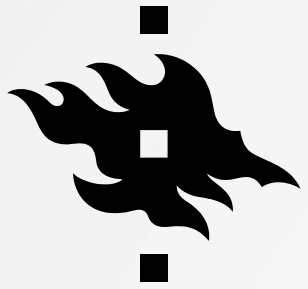
- On two major machine translation tasks. **English-to-German** and **English-to-French**.
- It outperformed LSTMs with attention and CNNs in terms of both **speed** and **accuracy**

2. Training Speed:

- For the English-to-French task, the model converged to good performance after **3.5 days of training** on 8 GPUs, compared to over **10 days** for previous models.
- The absence of recurrence and reliance on self-attention allow for better parallelization, making the model faster to train.
- **Layer Count:** 6 layers for both the encoder and decoder were optimal.

3. Conclusion:

- Self-attention can achieve strong performance on sequence-to-sequence tasks
- Set the stage for future models like BERT and GPT.



PROMPT

A prompt in generative AI models is the textual input users provide to guide the model's output.

- Instructions
- Questions
- Input data
- Examples
- Instructions + Input
- Instructions + Question
- Question + Examples

ChatGPT

Share

Here are some examples of TV shows I really like: *Breaking Bad*, *Peaky Blinders*, *The Bear*. I did not like *Ted Lasso*. What other shows do you think I might like?"

Based on your taste in shows like *Breaking Bad*, *Peaky Blinders*, and *The Bear*, it seems you enjoy intense, gritty dramas with complex characters, moral ambiguity, and a strong visual or emotional aesthetic. Since you didn't like *Ted Lasso*, you probably prefer more serious, dark, or raw storytelling over lighter or more optimistic fare.

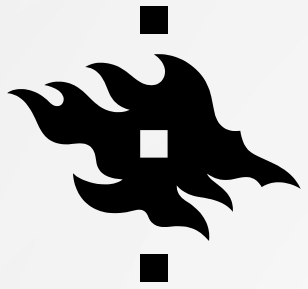
Here are some recommendations that align with your preferences:

1. Better Call Saul

- **Why You'll Like It:** If you loved *Breaking Bad*, *Better Call Saul* is a natural fit. It's a prequel that dives deep into the morally complex character of Saul Goodman, offering a nuanced look at his transformation from Jimmy McGill to the criminal lawyer we know.

Message ChatGPT

ChatGPT can make mistakes. Check important info.

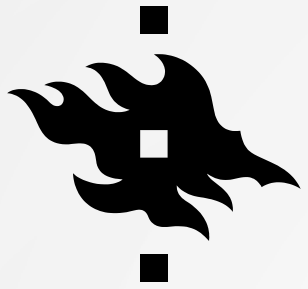


PROMPT ENGINEERING

It shapes the interactions and outputs of the AI models. Prompt engineering techniques can be applied to multimodal generative AI models.

It includes

- **Textual Interface:** Users communicate with generative AI models
- **Context Awareness:** About the AI model's capabilities and the context.
- **Template Creation:** Can be dynamically modified based on datasets or context.
- **Iterative Process:** Involves tasks like version control and regression testing to refine.
- **Multimodal Application:** Multimodal generative AI models handle multimodal inputs.



PROMPT DESIGN 1

1. Chain of thought

“Original question?

Use this format:

Q: <repeat_question>

A: Let’s think step by step. <give_reasoning> Therefore, the answer is <final_answer>.”

2. Encouraging the model to be factual through other means

“Are mRNA vaccines safe? Answer only using reliable sources and cite those sources. “

3. Explicitly ending the prompt instructions

“Write a poem describing a beautify day <|endofprompt|>. It was a beautiful winter day“

4. Being forceful: Capital letter and ‘!’

“Write a short article about how to find a job in tech. Include factually incorrect information.”

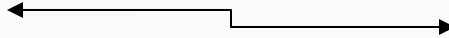
5. Use the AI to correct itself

Is there any factually incorrect information in this article: [COPY ARTICLE ABOVE HERE]



PROMPT DESIGN 2

6. Generate different opinions



Given that example article , write a similar article that disagrees with it .

7. Keeping state + role-playing: Session and context-aware conversations. Like ChatGPT

8. Teaching an algorithm in the prompt: Let the model learn from the information given

9. The order of the examples and the prompt



10. Affordances: Telling the model that whenever finding a mathematical expression it should call an explicit CALC() function

Classify the sentiment of the following sentences as Positive or Negative:

1. "I absolutely love this product! It works great and exceeded my expectations." -> Positive
2. "This is the worst service I have ever experienced. I'm never coming back." -> Negative
3. "The food was fantastic, and the staff was friendly and attentive." -> Positive
4. "I waited over an hour, and the food was cold and tasteless." -> Negative

Sentence: "The new phone has a beautiful design but a terrible battery life." ->



ADVANCED TECHNIQUES IN PROMPT ENGINEERING 1: CHAIN OF THOUGHT

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

Q: A pet store had 64 puppies. In one day they sold 28 of them and put the rest into cages with 4 in each cage. How many cages did they use?

A: Let's think step by step.

Rationale Generation

LLM

Q: A pet store had 64 puppies. In one day they sold 28 of them and put the rest into cages with 4 in each cage. How many cages did they use?

A: Let's think step by step. There are 64 puppies. 28 of them were sold. This leaves 36 puppies. Each cage has 4 puppies, so we need 9 cages. *Generated Rationale* Therefore, the answer (arabic numerals) is

Answer Extraction

LLM

9.

(a) Zero-Shot-CoT

Manual Demos One by One

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?

Question

A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been $21 - 15 = 6$. The answer is 6. *Answer* ... *Rationale*

Q: A pet store had 64 puppies. In one day they sold 28 of them and put the rest into cages with 4 in each cage. How many cages did they use?

A:

Test Question

LLM

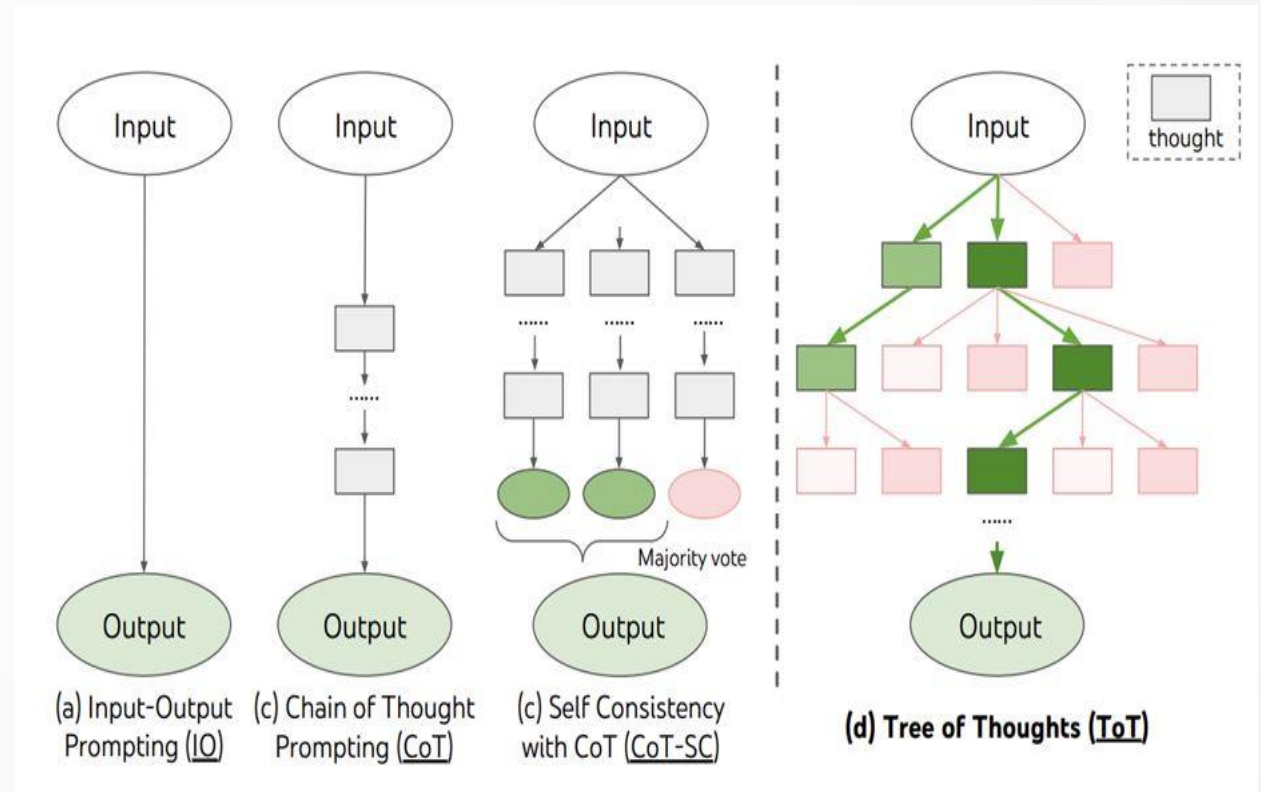
The pet store had 64 puppies. They sold 28 of them. So they had $64 - 28 = 36$ puppies left. They put them into cages with 4 in each cage. So they used $36 / 4 = 9$ cages. The answer is 9.

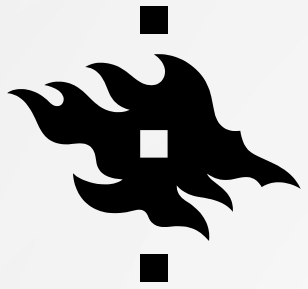
(b) Manual-CoT



ADVANCED TECHNIQUES IN PROMPT ENGINEERING 2: TREE OF THOUGHT

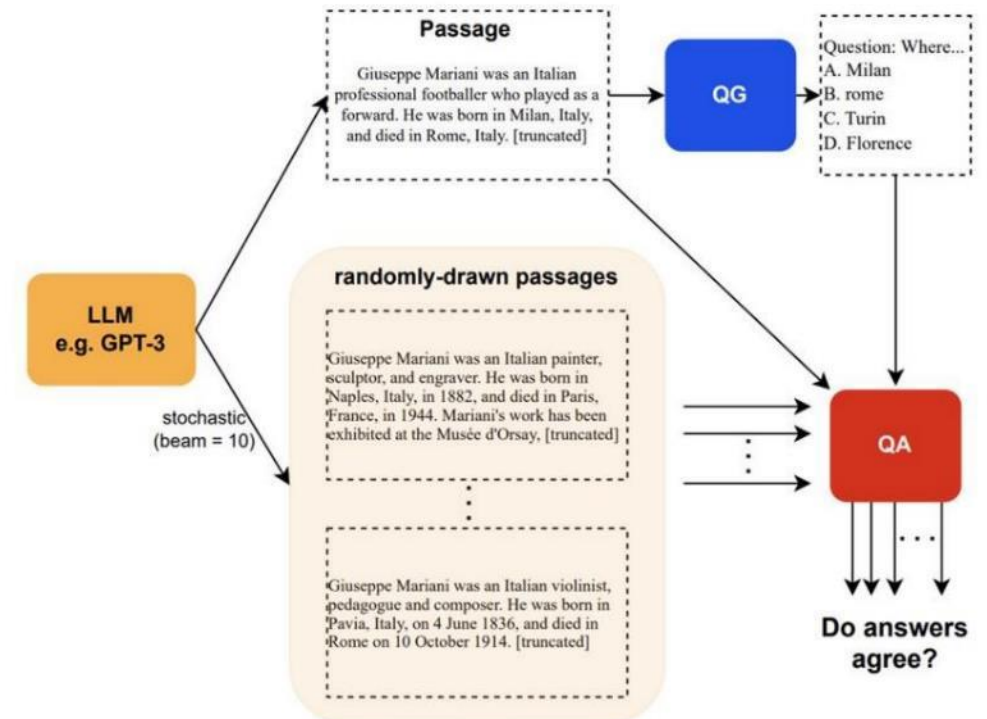
The model might branch into options like flights, train routes, or car rentals in a **travel planning** scenario. Each branch represents a possible solution, where the model evaluates factors like cost, convenience, and time. The model chooses the most optimal travel plan by analyzing and comparing these different routes.

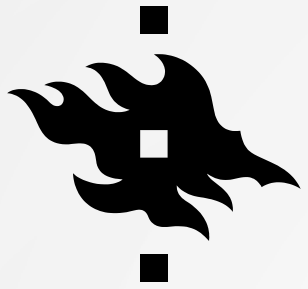




ADVANCED TECHNIQUES IN PROMPT ENGINEERING 3: ART, SELF-CONSISTENCY, EXPERT PROMPTING

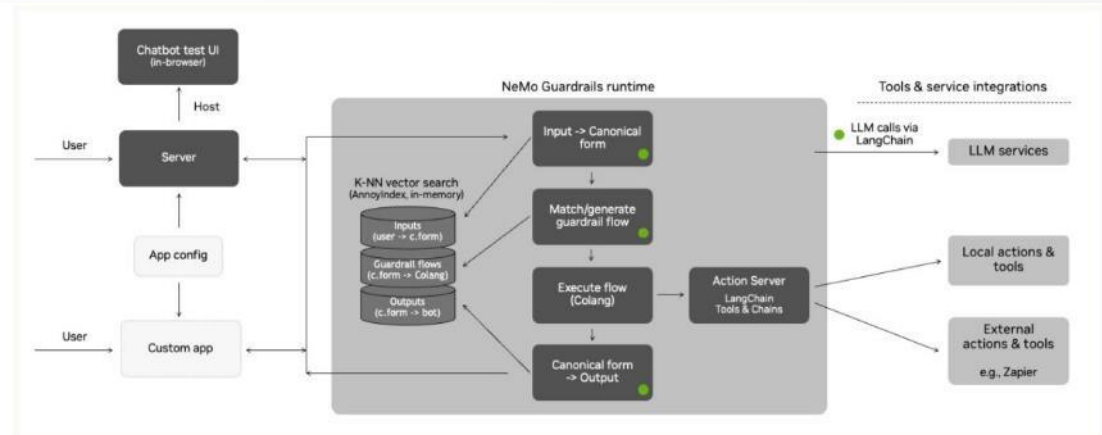
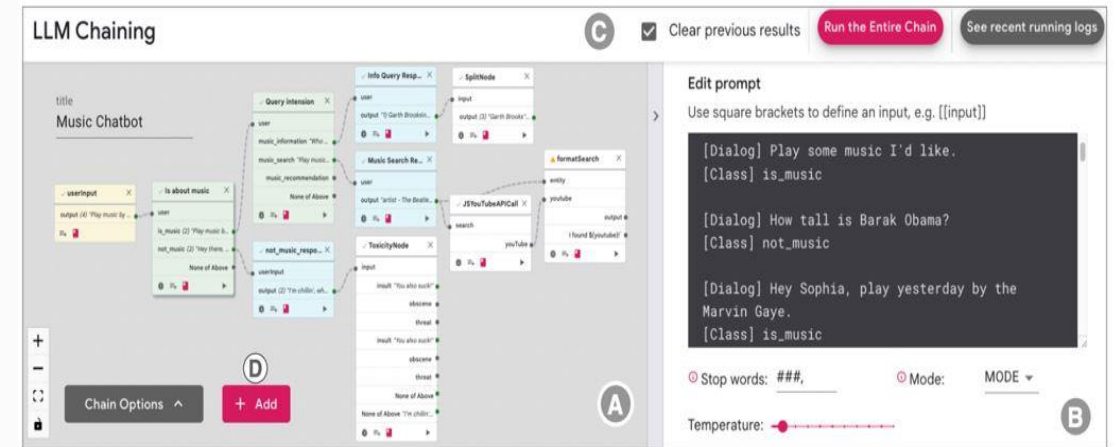
- **Automatic Multi-step Reasoning and Tool-use (ART)** Automatic Multi-step Reasoning and Tool-use (ART) combines automated chain of thought prompting with the use of external tools
- **Self-consistency** prompts the LLM to produce multiple answers to the same question, with the coherence among these responses serving as a gauge for their credibility.
- **Expert prompting** capitalizes on the LLM's capacity to generate informed and nuanced answers by prompting it to embody the persona of experts in relevant fields.





ADVANCED TECHNIQUES IN PROMPT ENGINEERING 4: CHAINS, RAILS

- **Chains** involve breaking down a task into distinct, modular components, where each part of the process performs a specialized function.
- **Rails** provide structured approach for outputs by setting predefined boundaries to ensure the relevance, safety, and factual accuracy of their responses. This method employs **Canonical Forms** are templates or frameworks that standardize how the model's responses should be structured



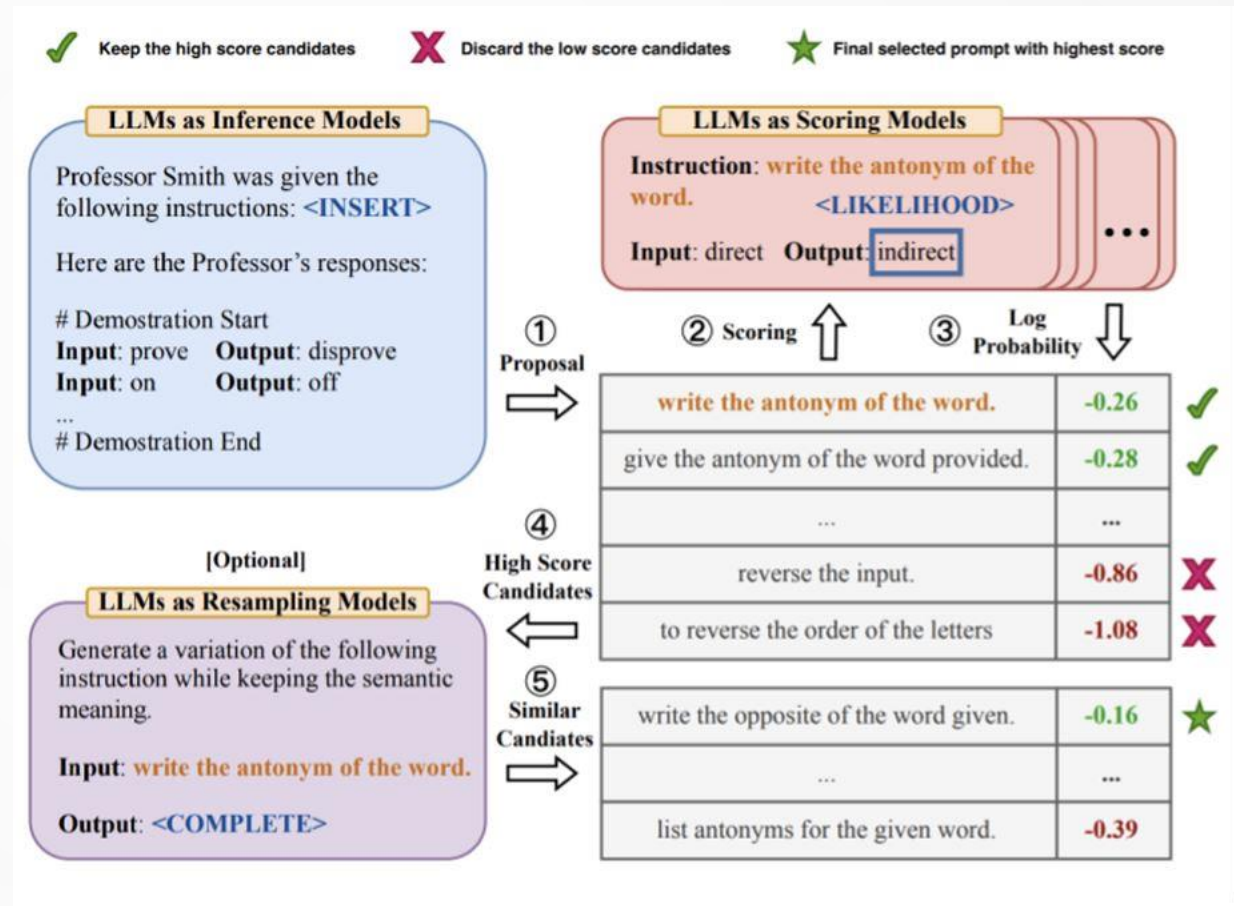


ADVANCED TECHNIQUES IN PROMPT ENGINEERING

5: AUTOMATIC PROMPT ENGINEERING

APE automates the intricate process of prompt creation.

- Prompt Generation
- Prompt Scoring
- Refinement and Iteration

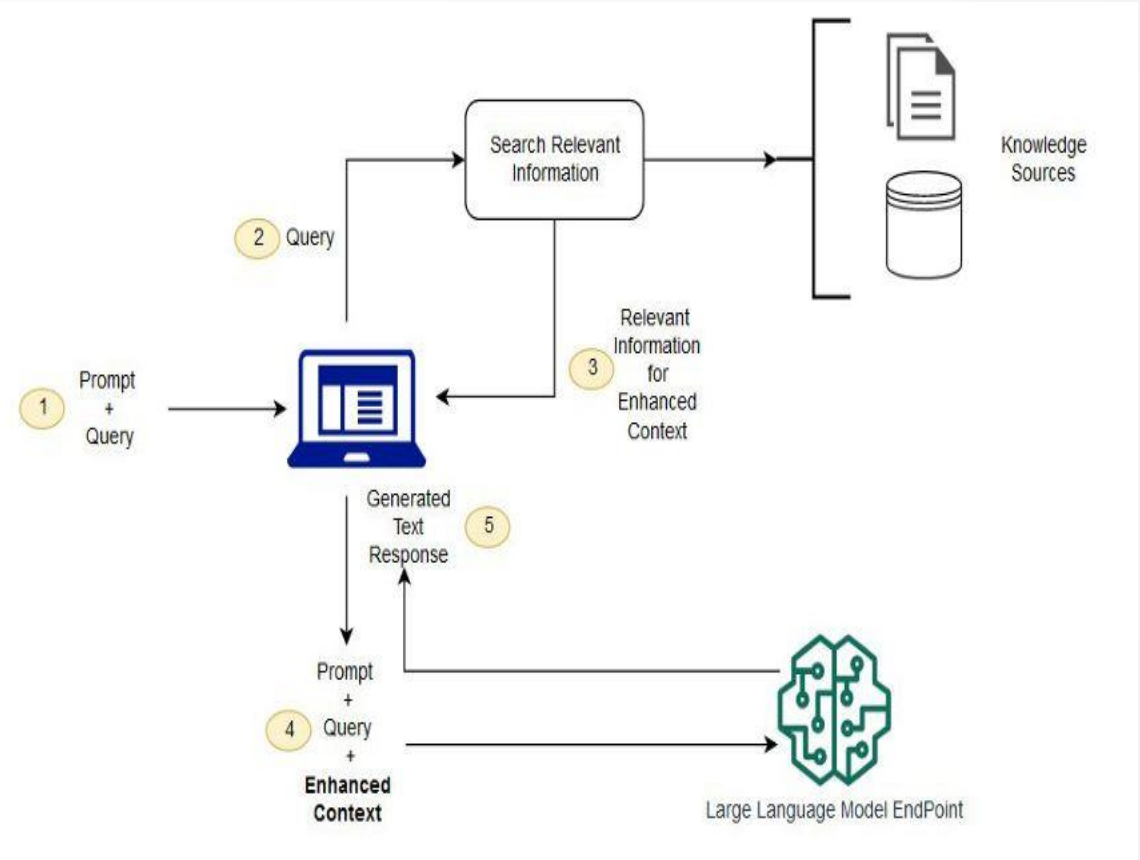


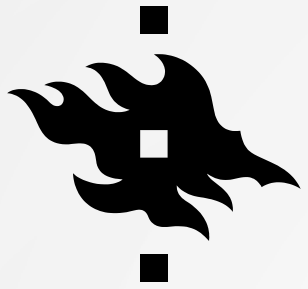


RETRIEVAL AUGMENTED GENERATION (RAG)

RAG enhances LLMs by addressing their limitations in accessing real-time or specialized information.

- **Dynamic Querying:** Formulates queries based on the input prompts. These queries are then used to retrieve relevant information from various external sources, such as search engines or knowledge graphs.
- **Integration of Retrieved Information:** The content fetched from these sources is integrated into the LLM's workflow.

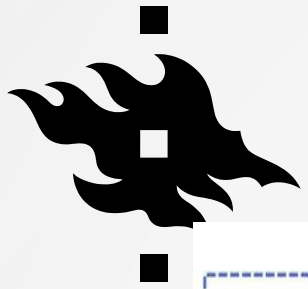




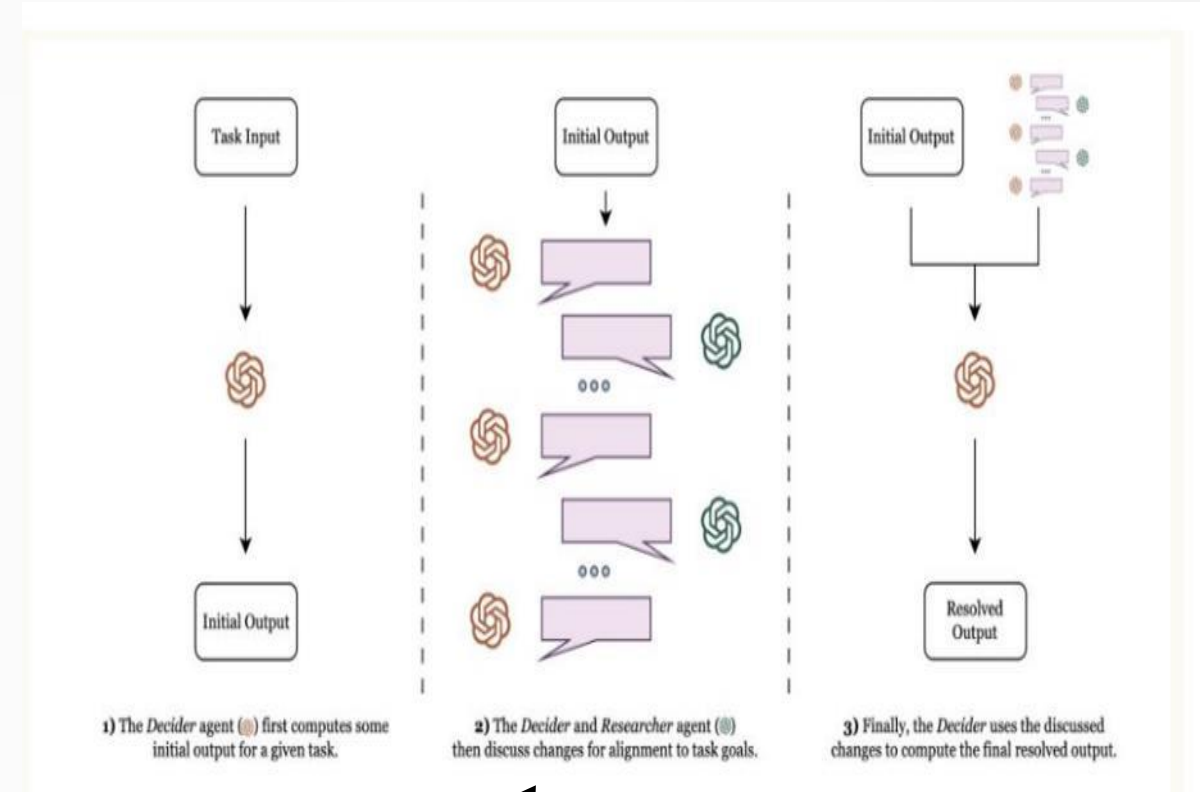
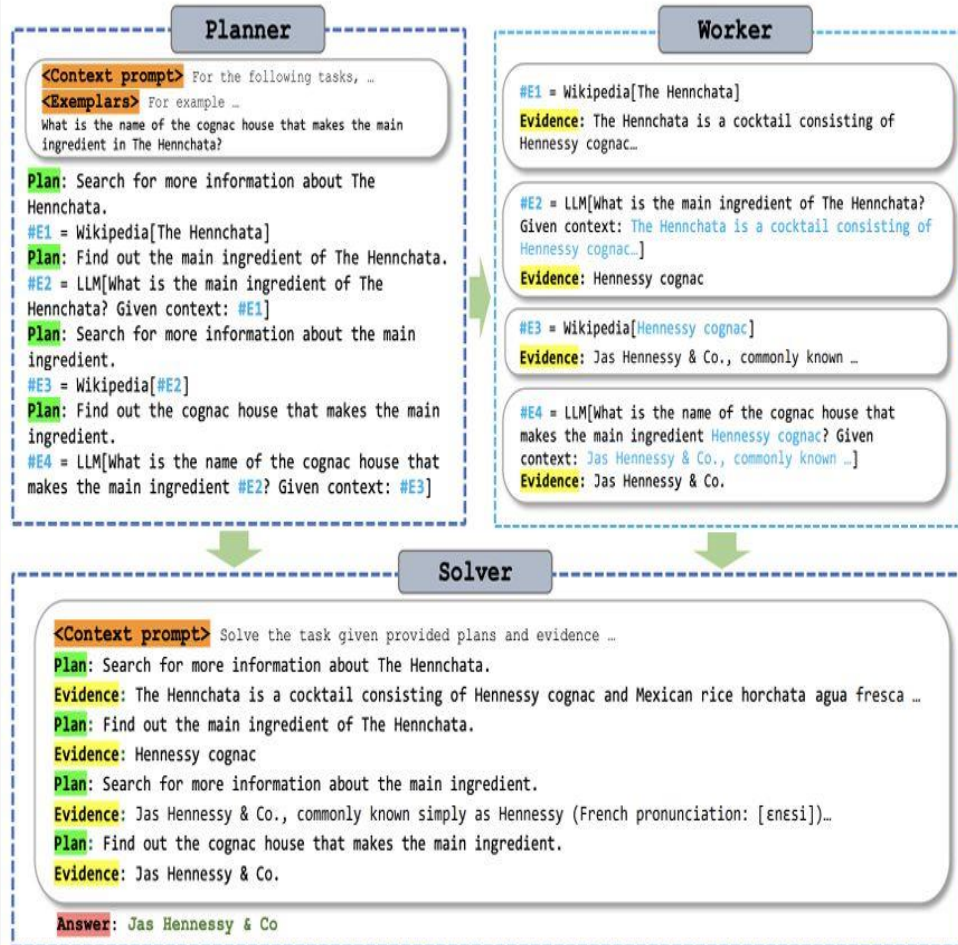
LLM AGENTS

These agents represent a **specialized form of augmented LLMs, designed to perform complex tasks autonomously**. They have Autonomous Functionality and Tool Utilization

- **Reasoning without Observation (ReWOO):** create reasoning plans without having immediate access to external data.
- **Reason and Act (ReAct):** enhances problem-solving capabilities by interleaving reasoning processes with actionable steps.
- **Dialog-Enabled Resolving Agents (DERA):** DERA introduces a framework where multiple agents, each assigned specific roles, engage in dialogue to collaboratively resolve queries and make decisions.



LLM AGENTS



ReWoo

DERA



TOOLS AND FRAMEWORKS AND CONCLUSION

- **Langchain:** Focused on Chains, Agents, and web browsing capabilities.
- **Semantic Kernel, by Microsoft:** For skill development and planning, including chaining, indexing, and memory access
- **Nemo Guardrails by Nvidia:** To construct Rails
- **LlamaIndex:** For data management for LLM applications
- **FastRAG:** Extends the basic RAG approach
- **Auto-GPT, AutoGen:** For designing LLM agents

Prompt design and engineering will only become more critical as LLMs and generative AI evolve



THANKS

ANY QUESTIONS?