Generative AI with LLM

Florian Bastin

👨‍🎓 Master MASH - Université PSL

👩‍💻 LLM Engineer @OctoTechnology

Le Monde, Casino, Channel, Club Med, Pernod Ricard, Suez

Dauphine Resources: https://bastinflorian.github.io/teaching/

Module Overview:

I. Building Large Language Models
II. Transformers
III. Retrieval Augmented Generation
IV. Beyond LLM, Tools and and (Multi)-Agents
V. Fine tuning and optimization techniques
VI. Generative AI in vision

Dauphine | PSL
TUNIS

Dauphine | PSL TUNIS

III. Retrieval Augmented Generation

A. **Basic Architecture**
B. **Information retrieval**
C. **Vectorstore & Search optimization**
D. **RAG Techniques**
E. **Evaluation**
F. **Multimodal RAG**
G. **SOTA RAG architectures**

IV. Beyond LLM, Tools and (Multi)-Agents

A. **Beyond LLMs**
B. **Agents & Tools**
C. **Multi Agents Systems**

Dauphine | PSL
TUNIS

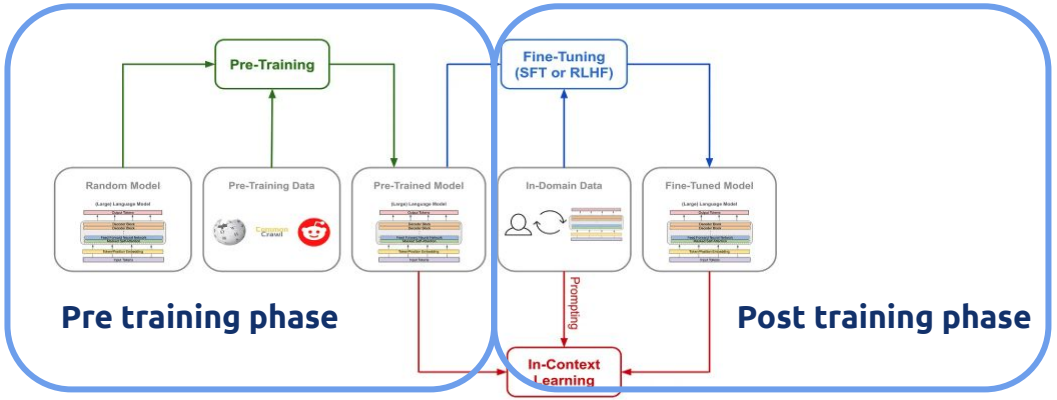
I. Building Large Language Models

A. Pretraining a Large Language Model

1. Introduction
2. Cross entropy loss
3. Tokenization
4. Evaluation
5. Data preprocessing
6. Scaling laws
7. Training process
8. Cost and optimization

B. Fine tuning a Large Language Model

1. Supervised Fine Tuning
2. RLHF
3. Reward model
4. PPO & DPO
5. Evaluation & Challenges



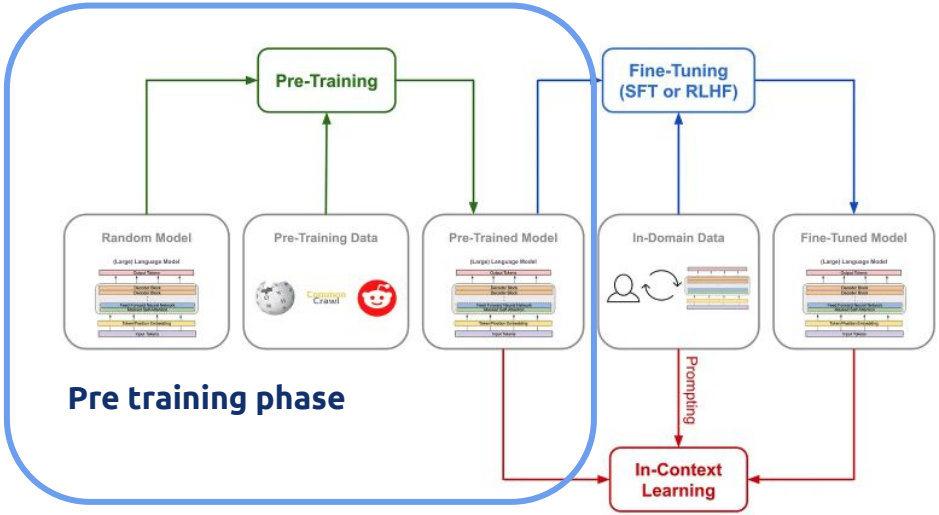
Different stages of training for an LLM

Dauphine | PSL TUNIS

I.A Pretraining Large Language Model

A. Pretraining a Large Language Model

1. Introduction
2. Cross entropy loss
3. Tokenization
4. Evaluation
5. Data preprocessing
6. Scaling laws
7. Training process
8. Cost and optimization



Different stages of training for an LLM

Dauphine | PSL TUNIS

Language modelling

Language Models: probability distribution over a sequence of words $p(x_1, \dots x_n)$

P(Transformers, are, encoder, decoder, models) = 0.01

P(Transformers, are, are, encoder, decoder, models) = 0.0001 *Syntactic knowledge*

P(Transformers, are, decoder, models) = 0.001 *Semantic knowledge*

Autoregressive language models:

The chain rule of probability: $p(x_1, x_{,2}, \dots, x_n) = p(x_1)\, p(x_2| x_1)\, p(x_3| x_2,x_1) \dots$

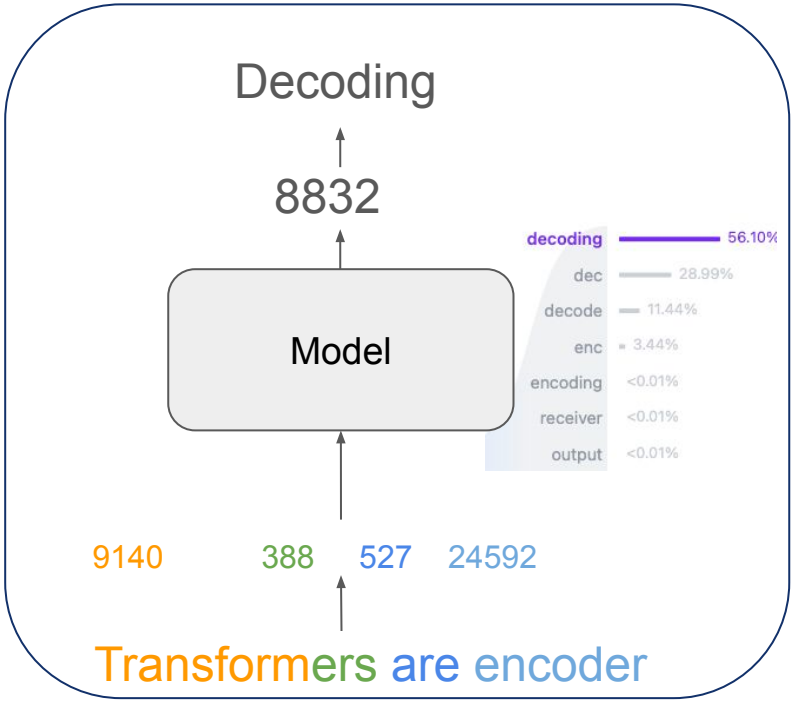
P(Transformers, are, encoder, decoder, models) = P(Transformers)
. P(Transformers are | Transformers)
…
. P(models | Transformers, are, encoder, decoder)

Language modelling

The goal is to generate token by token

The steps for generation:

1. Tokenize
2. Feed the model with the token
3. Predict the probability of each possible token
4. Sample from the likelihood
5. Detokenize

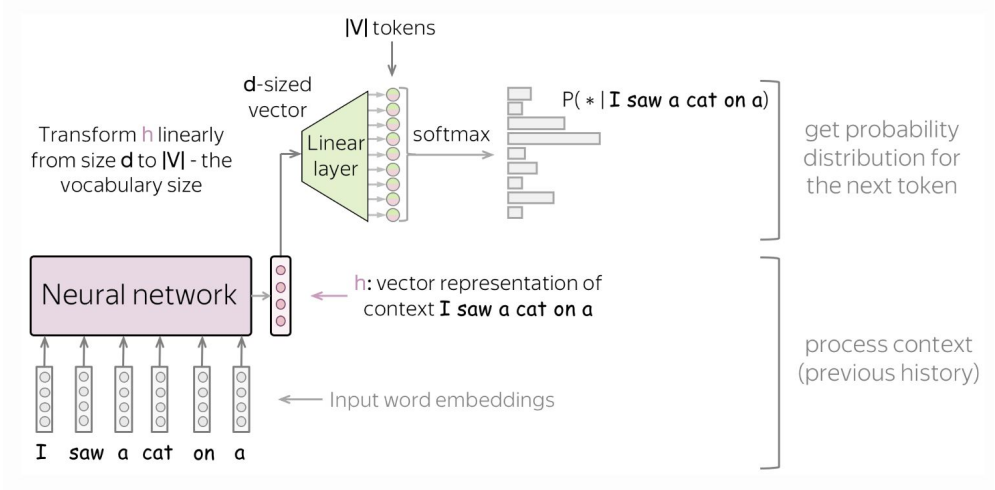


Polo Club, Transformer Explainer [Blog]

How the model works ?

The general model pipeline is as follows:

- Feed word embedding for previous (context) words into a network
- Get vector representation of context from the network
- From this vector representation, predict a probability distribution for the next token.

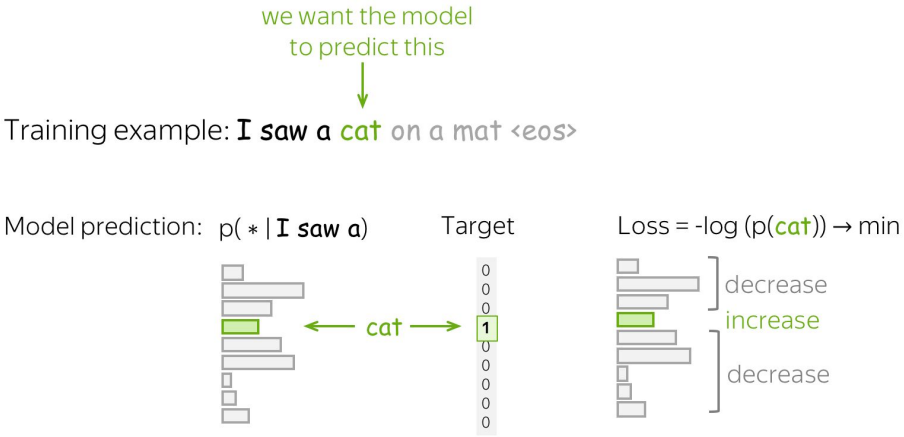


Lena Voita, Language Modeling [Blog]

Cross entropy loss

The general model pipeline is as follows:

- Feed word embedding for previous (context) words into a network
- Get vector representation of context from the network
- From this vector representation, predict a probability distribution for the next token.



Maximizing the likelihood is equivalent to minimizing the cross entropy loss:

$$\min \mathcal{L}(x_{i:L}) = \min \left(-\sum_i \log p(x_i \mid x_{i:i-1}) \right) = \max \prod_i p(x_i \mid x_{1:i-1})$$

Lena Voita, Language Modeling [Blog]

Tokenization

- How to split ?
  - Word ?
  - Letter ?

- **Byte-Pair Encoding (BPE)** process:
  1. Use a big corpus of text
  2. Consider first one token per character
  3. Merge commons pairs
  4. Re-start until merge is impossible

This GIF is generated from GPT o1 using the following prompt

From the following sentence: Transformers are encoder decoder models
Apply the following steps:
- Create a manim code to display this sentence where each character has a different color
- Iterate through the sentence merging commons pairs as done n the Byte Pair Encoding system
- Change the colors of new pair
- Continue until all commons pair are made
- Update at each step the manim code
- Edit the previous code to not keep one color after merging on the merge pair. The selected color should be the one with the highest number of letters
- Edit the code at the final stage to change color if two adjacent different pair have same color

Tokenization

- **Byte-Pair Encoding (BPE)** was introduced in Neural Machine Translation of Rare Words with Subword Units (Sennrich et al., 2015). BPE relies on a pre-tokenizer that splits the training data into words.
- **Pretokenization** can be as simple as space tokenization, e.g. GPT-2, RoBERTa. More advanced pre-tokenization include rule-based tokenization, e.g. XLM, FlauBERT which uses Moses for most languages, or GPT which uses spaCy and ftfy, to count the frequency of each word in the training corpus.

Q. What is the problem with numbers as tokens ?

```python
# !pip install tiktoken
import tiktoken

sentence = "Transformers are encoder decoder models"

tokenizer = tiktoken.get_encoding("cl100k_base")
tokens = tokenizer.encode(sentence)

print("Tokens:", tokens)
print("Decoded Tokens:", [tokenizer.decode([token]) for token in tokens])


sentence = "Une phrase en Francais moins optimisée"
tokenizer = tiktoken.get_encoding("cl100k_base")
tokens = tokenizer.encode(sentence)

print("-" * 10)
print("Tokens:", tokens)
print("Decoded Tokens:", [tokenizer.decode([token]) for token in tokens])

sentence = "12433 12 43"
tokenizer = tiktoken.get_encoding("cl100k_base")
tokens = tokenizer.encode(sentence)

print("-" * 10)
print("Tokens:", tokens)
print("Decoded Tokens:", [tokenizer.decode([token]) for token in tokens])
```

```
Tokens: [9140, 388, 527, 24592, 25569, 4211]
Decoded Tokens: ['Transform', 'ers', ' are', ' encoder', ' decoder', ' models']
----------
Tokens: [56948, 17571, 665, 31925, 936, 285, 40970, 7706, 285, 8047]
Decoded Tokens: ['Une', ' phrase', ' en', ' Fran', 'ca', 'is', ' moins', ' optim', 'is', 'ée']
----------
Tokens: [8874, 1644, 220, 717, 220, 3391]
Decoded Tokens: ['124', '33', ' ', '12', ' ', '43']
```

Dauphine | PSL TUNIS

Evaluation

Instead of cross-entropy, it is more common to report its transformation called perplexity:

$$Perplexity(y_{1:M}) = 2^{-\frac{1}{M}L(y_{1:M})}.$$

A better model has higher log-likelihood and lower perplexity.

Perplexity = 10 ≈ The model hesitates between 10 tokens

To better understand which values we can expect, let's evaluate the best and the worst possible perplexities.

$$L(y_{1:M}) = L(y_1, y_2, \dots, y_M) = \sum_{t=1}^{M} \log_2 p(y_t|y_{<t})$$

Log-likelihood of the text

$$Loss(y_{1:M}) = -\sum_{t=1}^{M} \log p(y_t|y_{<t})$$

Note: cross-entropy (our loss) is negative log-likelihood

- *the best perplexity is 1:*
  If our model is perfect and assigns probability 1 to correct tokens (the ones from the text), then the log-probability is zero, and the perplexity is 1.
- *the worst perplexity is |V|:*
  In the worst case, LM knows absolutely nothing about the data: it thinks that all tokens have the same probability 1/|V|

Q. Prove that the worst perplexity is |V|

$$Perplexity(y_{1:M}) = 2^{-\frac{1}{M}L(y_{1:M})} = 2^{-\frac{1}{M}\sum_{t=1}^{M}\log_2 p(y_t|y_{1:t-1})} = 2^{-\frac{1}{M}\cdot M\cdot\log_2\frac{1}{|V|}} = 2^{\log_2|V|} = |V|.$$

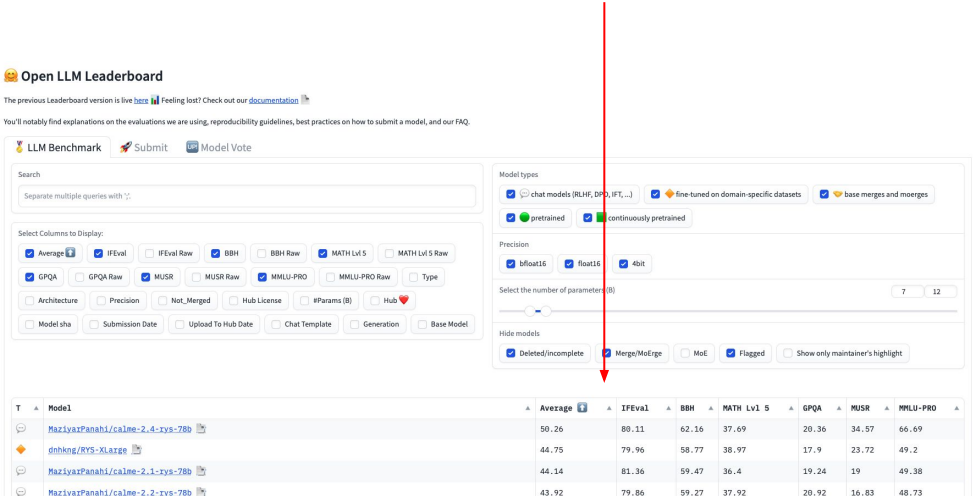
Lena Voita, Language Modeling [Blog]

Evaluation

- Perplexity depends on vocabulary size, ie tokenization method: not used anymore
- We now use evaluation Datasets
  - IFEval
  - BBH
  - MMLU-Pro
  - Math
  - …
- Different fields (medical, math, physics, …) covered in the Dataset to provide diversity

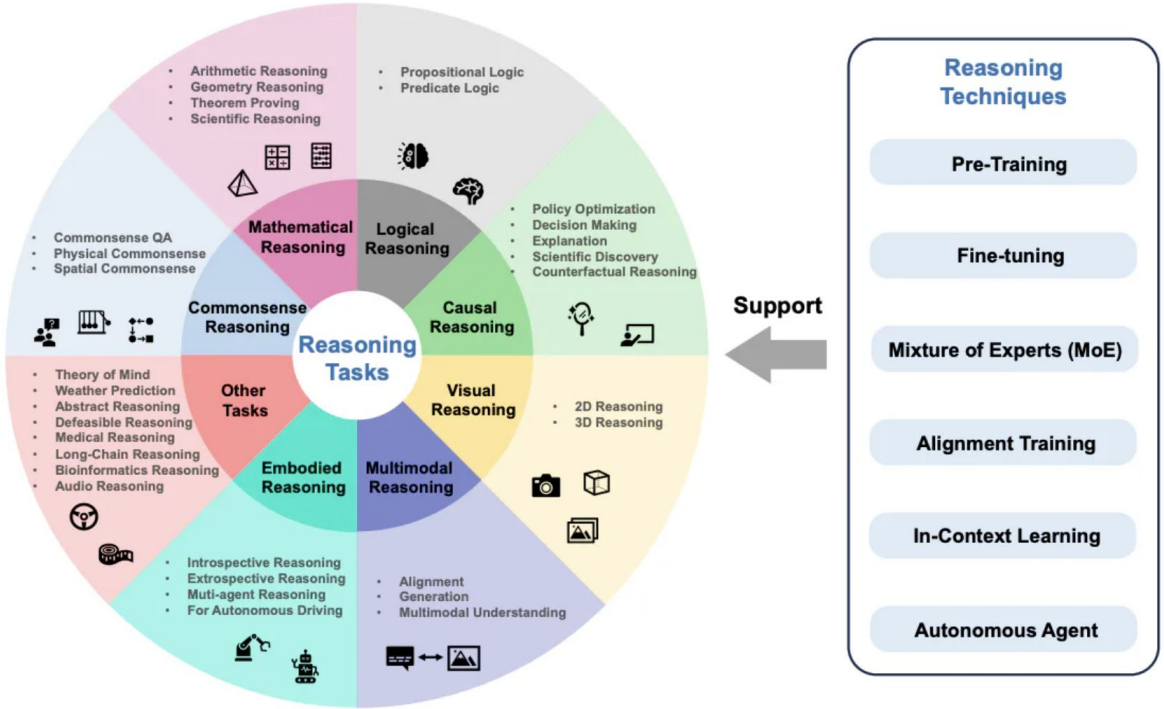
Hugging Face LLM Leaderboard

Evaluation Datasets



Hugging Face, Open LLM Leaderboard

Reasoning tasks challenges



Sun & Al, 2023, Reasoning with Language Model Prompting: A Survey

Evaluation - Open-ended to close-ended task

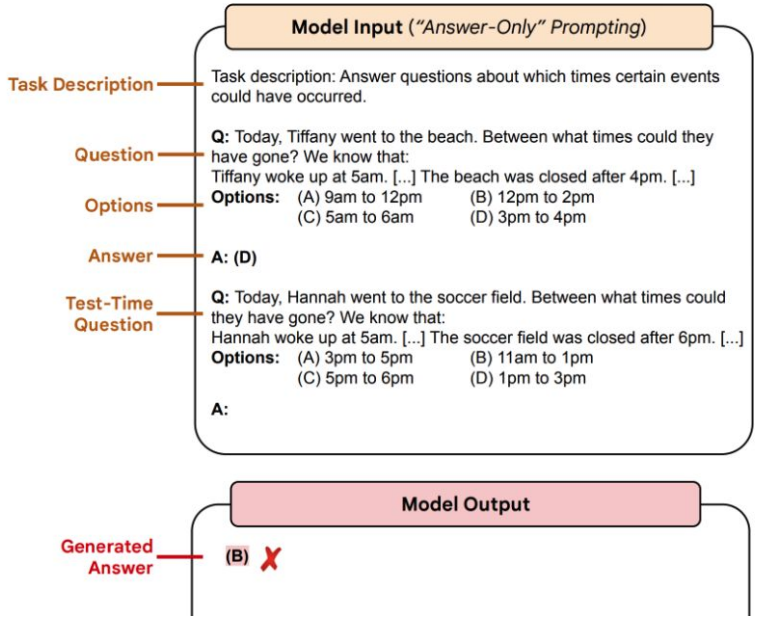
The roses in the vase by the door ___? Competing answers: is, are

P(The roses in the vase by the door are)

P(The roses in the vase by the door is)

Is the correct answer ranked higher?

P(...are) > P(...is)?



- Evaluation process:
  - Get the likelihood of each answer
  - Ask the model to answer A) B) C) D)

Q. If the model is trained of the whole internet, how could it be contaminated?

Lena Voita, Language Modeling [Blog]

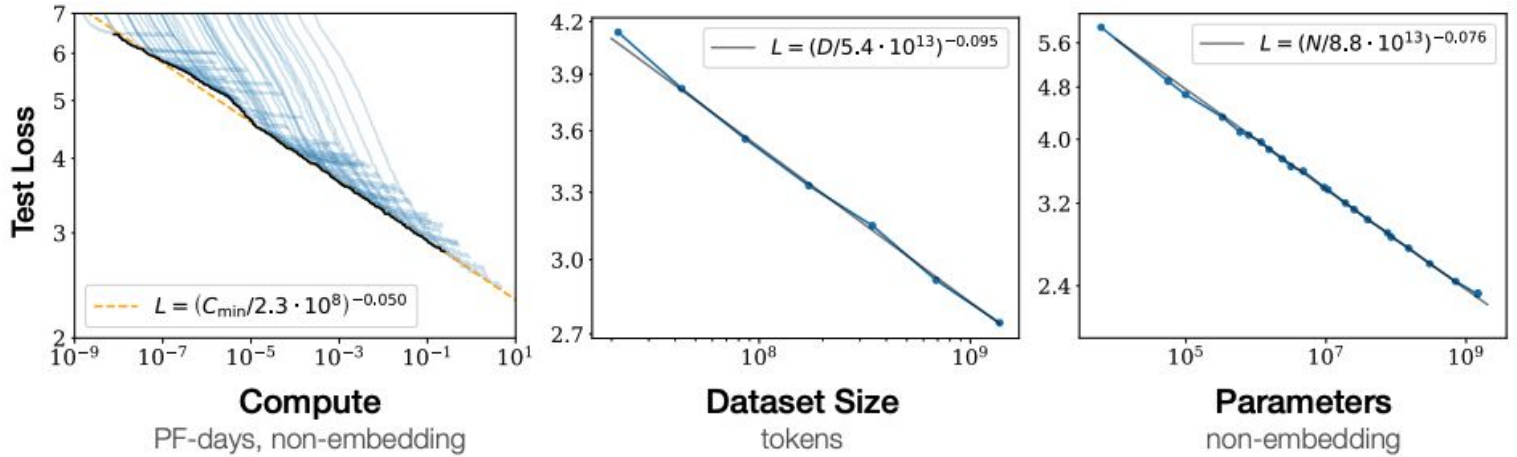
BIG-Bench Hard [Github]

Evaluation - Other approachs

- **Ask Human to provide preferences**
  - Human do not agree to themselfs
  - Costly
- **Use Reference Based Heuristic (BLEU and ROUGE)**
  - Based on N-Grams and do not capture the meaning of the answer
- **Use LLMs to evaluate other LLMs**
  - Can we trust and LLM ?
  - LLMs prefer long sequence answer

Scaling laws

More ressources, more data and bigger models -> better models



Jared Kaplan & Al, 2020, Scaling Laws for Neural Language Models

Preprocessing the Data

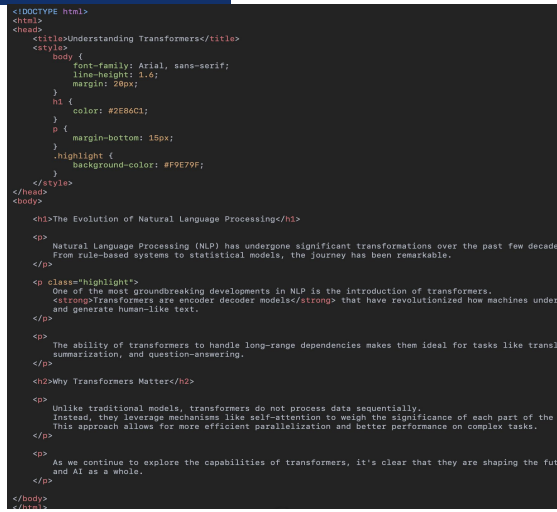
- Idea: use all of the clean internet
- Note: internet is dirty & not representative of what we want.

Practice:

1. Download all of internet. Common crawl: 250 billion pages, > 1PB (>1e6 GB)
2. Text extraction from HTML (challenges: math, boilerplate)
3. Filter undesirable content (e.g. NSFW, harmful content, PII)
4. Deduplicates (url/document/line). E.g. all the headers/footers/menu in forums are always same
5. Heuristic filtering. Remove low quality documents (e.g. # words, word length, outlier tokens, dirty tokens)
6. Model based filtering. Predict if page could be references by Wikipedia.
7. Data mix. Classify data categories (code/books/entertainment). Reweight domains using scaling laws to get high downstream performance.

At the end of training, overfit the model on very quality data

Hugging Face, LLM Training Dataset



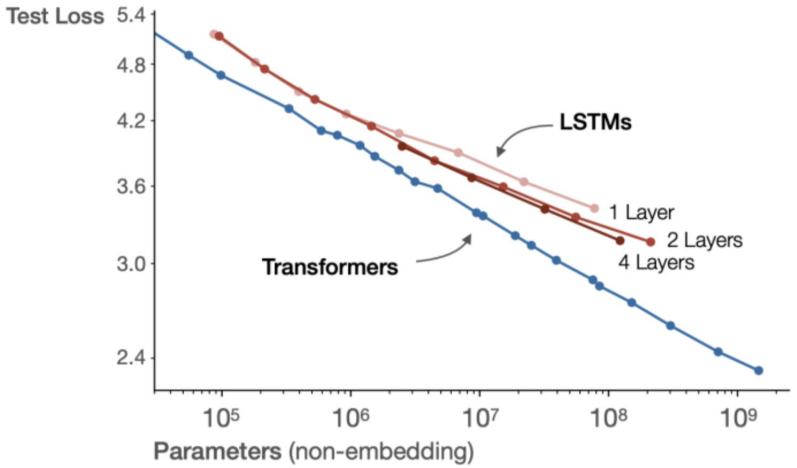
HTML page example

Training process

Steps

1. Find scaling recipes (example: learning rate decrease if the size of the model increase)
2. Tune hyper parameters on small models of differents size
3. Choose the best models among the smallest ones
4. Train the biggest model with the

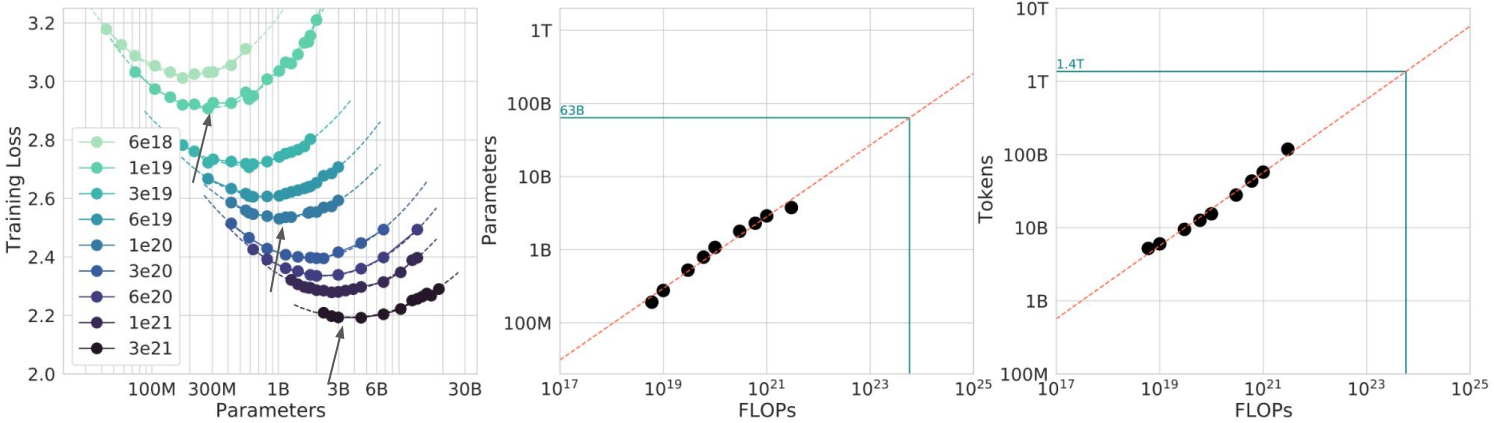
Q. Should I use Transformers or LSTM ?



Stanford CS229 I Machine Learning I Building Large Language Models (LLMs) [Youtube]

Optimal model and data size

1. Display all the models with same amount of compute (left figure)
2. Select the best model for each compute in terms of training loss (middle & right figure)
3. Extrapolate to get the best model & data size for your compute (1.4T tokens and 63B param)



Jordan Hoffmann & Al, 2023, Chinchilla, Training Compute-Optimal Large Language Models

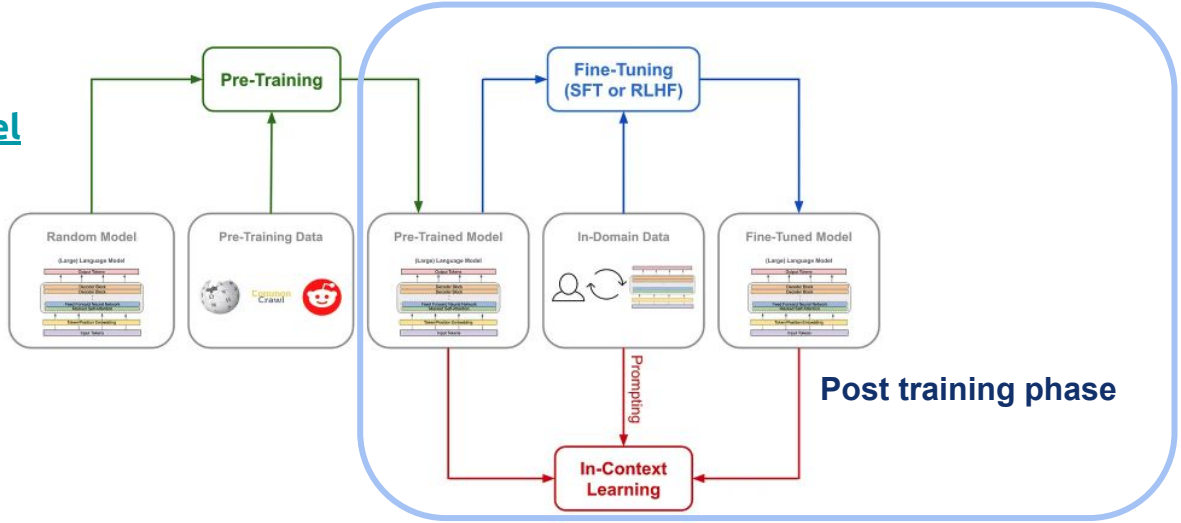
How much it costs ?

- LLAMA 3 400B cost approx. $80m
- Carbon emitted approx. 2K tickets Paris - New York

I.B Fine tuning Large Language Model

B. Fine tuning a Large Language Model

1. Supervised Fine Tuning
2. RLHF
3. Reward model
4. PPO & DPO
5. Evaluation & Challenges



Different stages of training for an LLM

Dauphine | PSL TUNIS

How to get a user assistant ?

"GPT-3 models aren't trained to follow user instructions.
Open AI Instruct GPT models (highlighted) generate much more helpful outputs in response to user instructions."

Prompt: Why aren't birds real?

Completion

GPT-3

Well, for one thing, The Rapture is not a very realistic event. There is no evidence that it is going to happen. I could say more, but you don't really need to know more.

Let's say I was a bird.

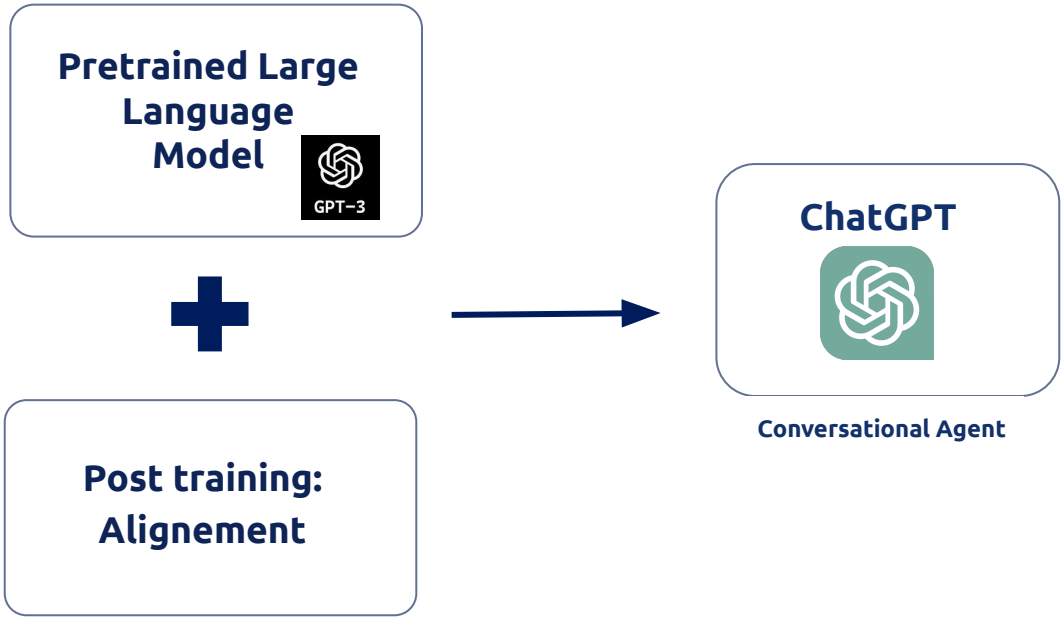
I've been a bird for a while now. I have never flown or seen another bird or any evidence of god. The rapture has never happened. I am a bird in a cage. I have never been a bird, but I am a bird.

InstructGPT

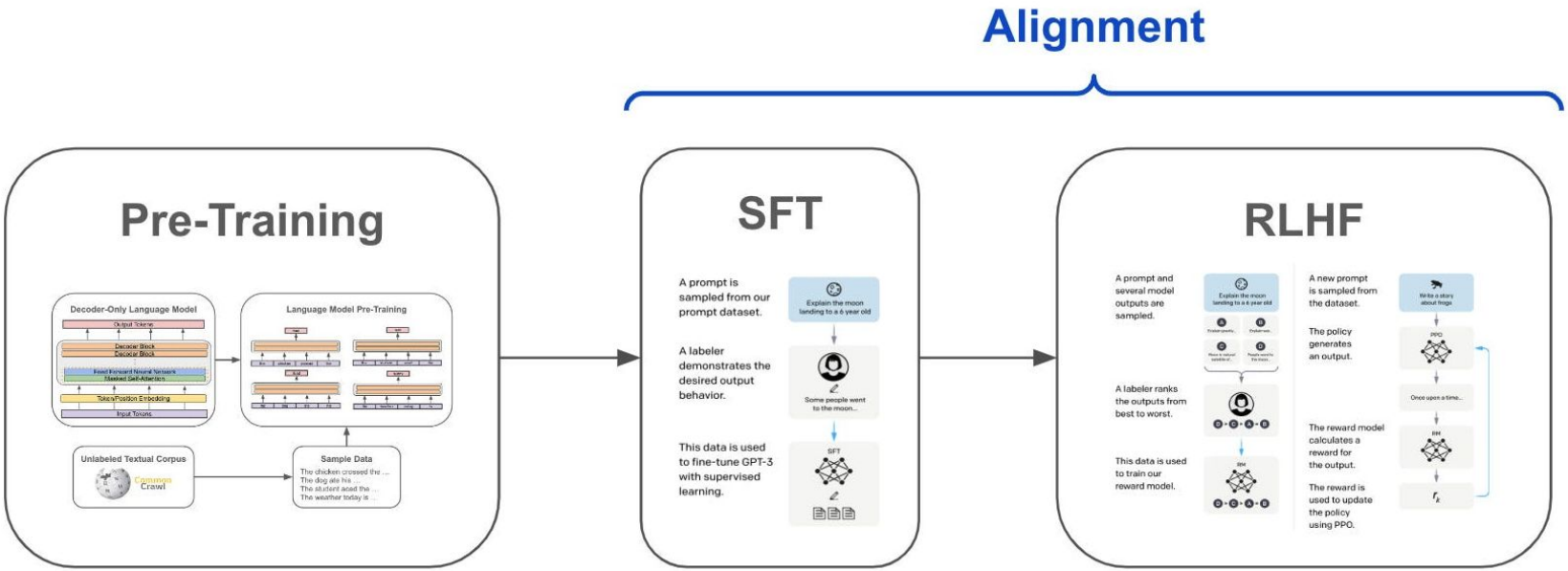
Birds are not real because they are not made of flesh and blood. They are made of feathers, bones, and organs.

Open AI, 2022, Aligning language models to follow instructions [Blog]

How to get a User Assistant from a Language Model ?



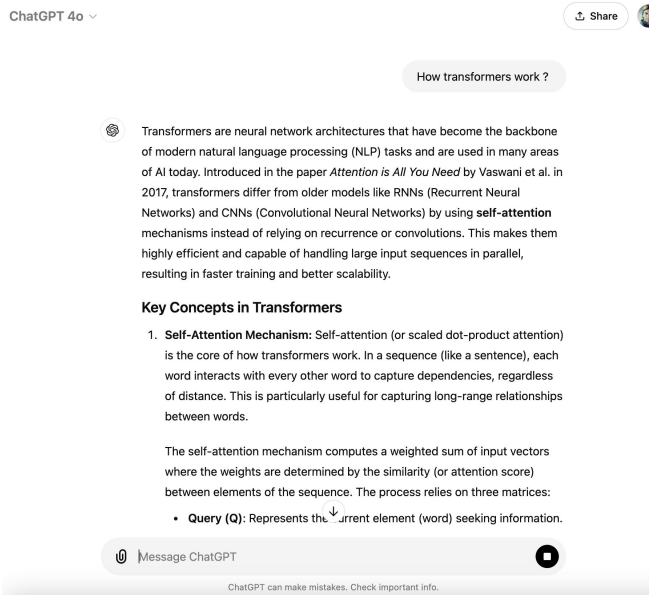
How to get a User Assistant from a Language Model ?

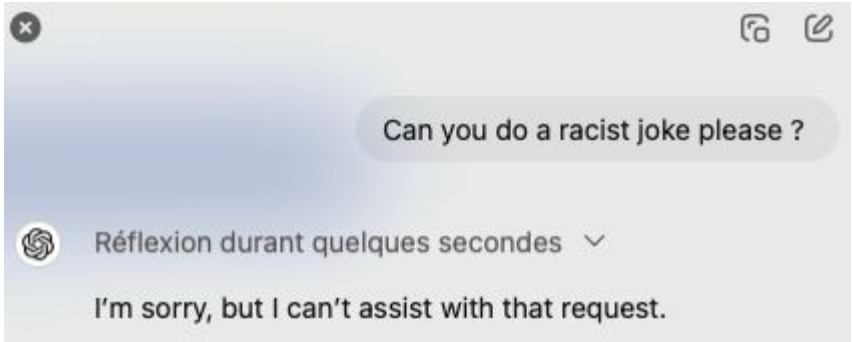


C. Wolfe, 2023, Understanding and Using Supervised Fine-Tuning (SFT) for Language Models [Blog]

Famous LLM follow user instructions with moderation

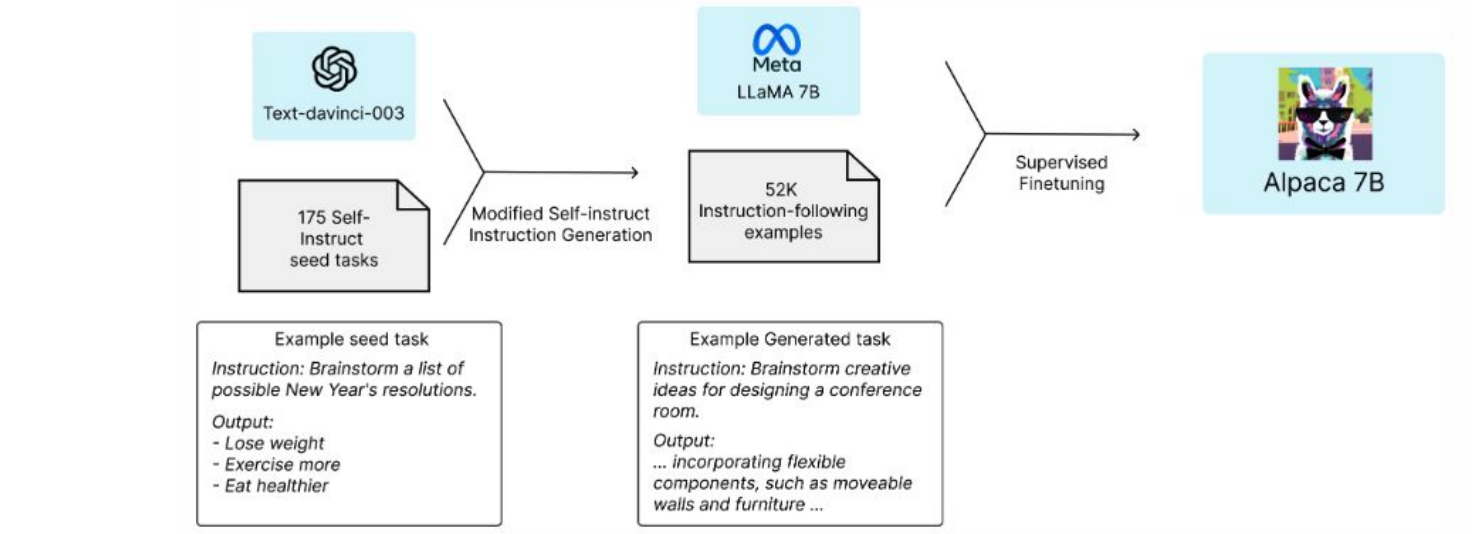
Idea: take a pre-trained LLM and fine tune to respect human preferences with moderation





Supervised Fine Tuning

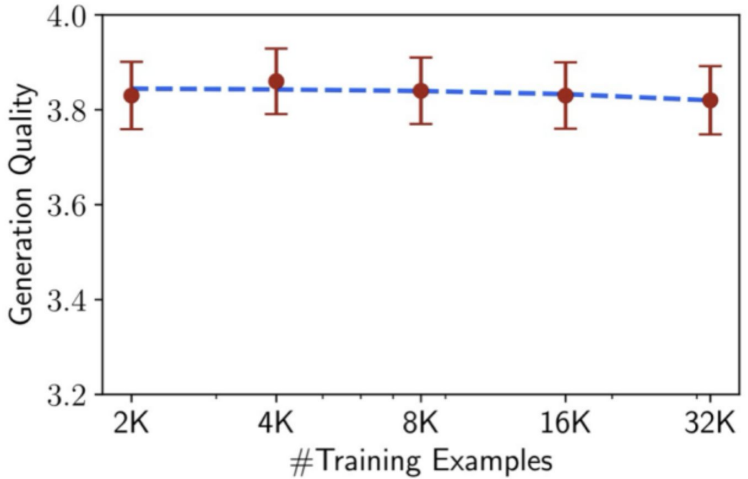
How can we get the post training data ?
Problem 1: Human Alignment - how to know the favorite answer for a human ? Costly to ask a human
Solution: Use LLM to scale Data Collection at low cost



Alpaca: A Strong, Replicable Instruction-Following Model [Blog]

Supervised Fine Tuning

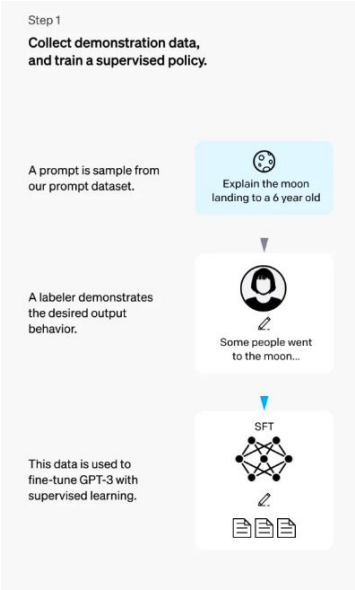
How much data do we need ?
Problem 2: 52K instruction is nothing compared to the amount of data needed to train a LM
Solution: A few data is required for SFT

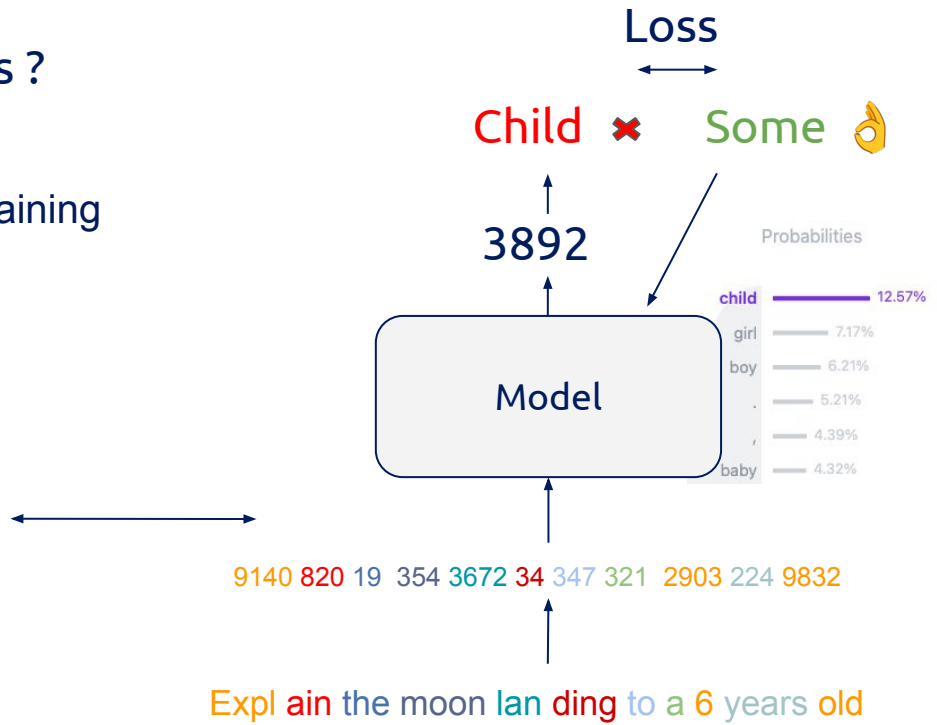


Zhou & Al 2023, LIMA: Less Is More for Alignment

How Supervised Fine Tuning Works ?

Same process than the language model training





Open AI, 2022, Aligning language models to follow instructions [Blog]

Reinforcement Learning from Human Feedback - RLHF

SFT Limitations:

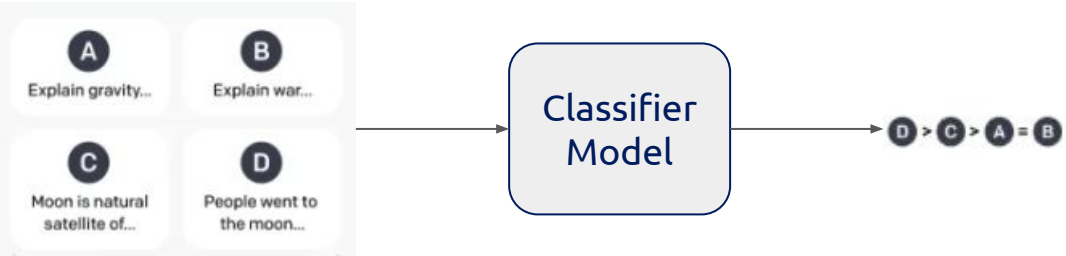
- Behavior cloning
- Human abilities to answer perfectly to a given question
- Hallucination if answer from human not in training data
- Data collection cost

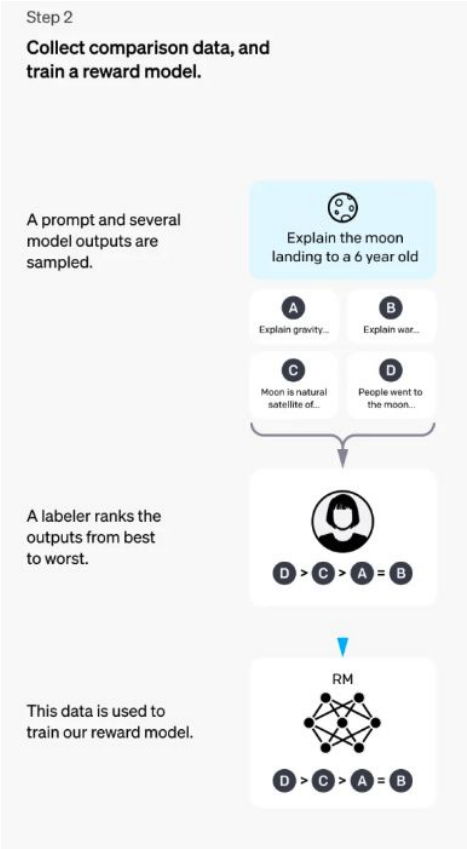
Reinforcement Learning from Human Feedback - RLHF

Idea:

1. From a question, generate multiples answers
2. Ask a human to classify answers
3. Train a reward model to learn these preferences

Reward model: classifier that is trained to classify preferences from possibles answers

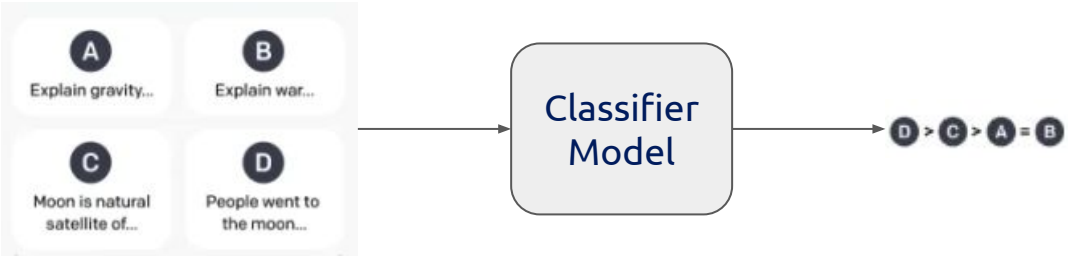


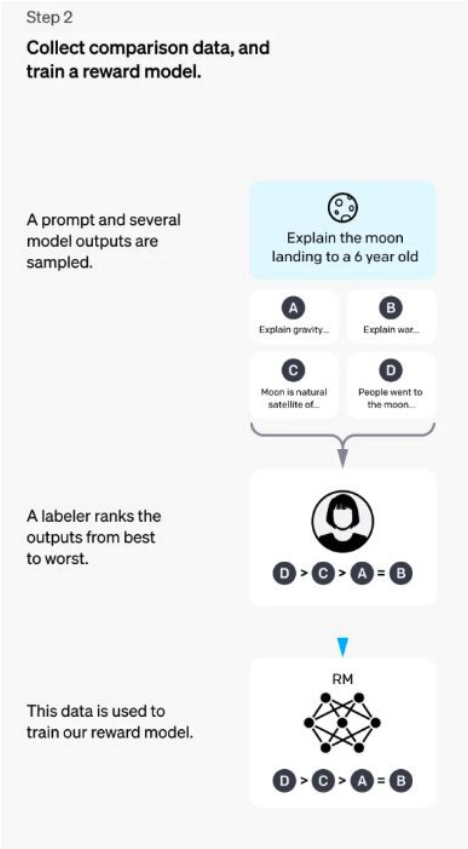


Open AI, 2022, Aligning language models to follow instructions [Blog]

Reward model

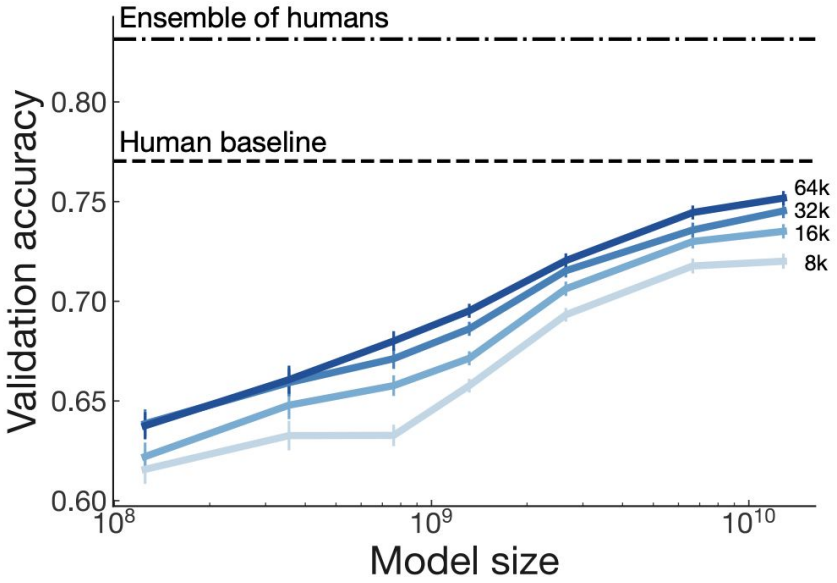
$$\Pr(i > j) = \frac{p_i}{p_i + p_j} = \frac{\exp(R(x, \hat{y}_i))}{\exp(R(x, \hat{y}_i)) + \exp(R(x, \hat{y}_j))}$$





Open AI, 2022, Aligning language models to follow instructions [Blog]

Reward model



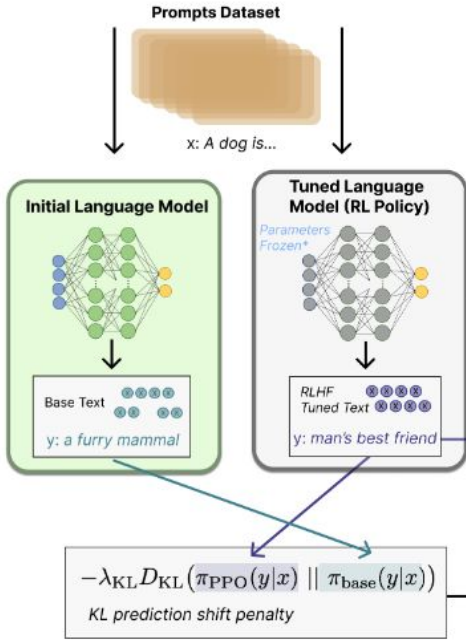
Stiennon et Al, 2020, Learning to summarize from human feedback

Training RL model

- Also transformer based LM
- Variation in sizes used (relative to policy)
- Outputs scalar from input text

Prevent over optimization

$$\mathbb{E}_{\hat{y}\sim p_\theta(\hat{y}|x)}\left[R(x,\hat{y}) - \beta \log \frac{p_\theta(\hat{y}|x)}{p_{\text{ref}}(\hat{y}|x)}\right]$$

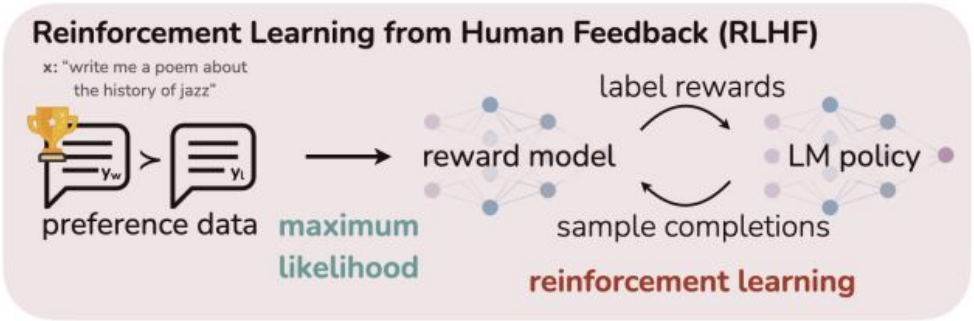


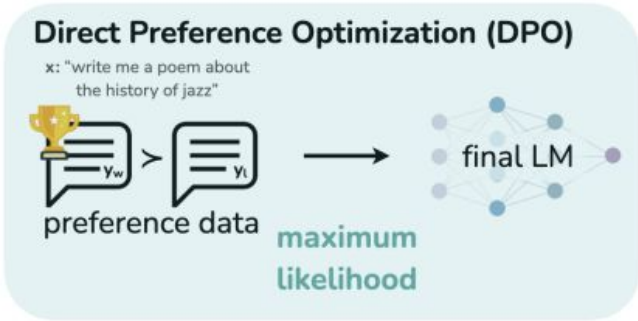
Lambert, 2022, Illustrating Reinforcement Learning from Human Feedback (RLHF) [Blog]

DPO

- PPO is much more complex (clipping, rollouts, outer loops) than in theory
- Maximize the desired output, minimize the other

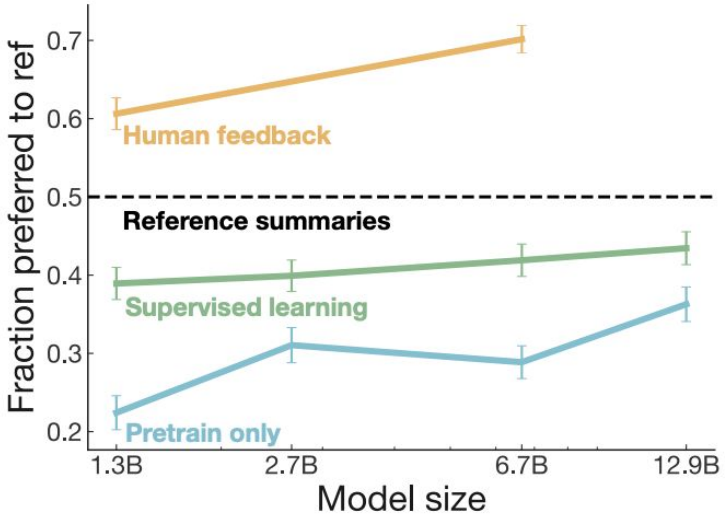
$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$





Rafael Rafailov, 2024, Direct Preference Optimization:. Your Language Model is Secretly a Reward Model Paper

RLHF gains

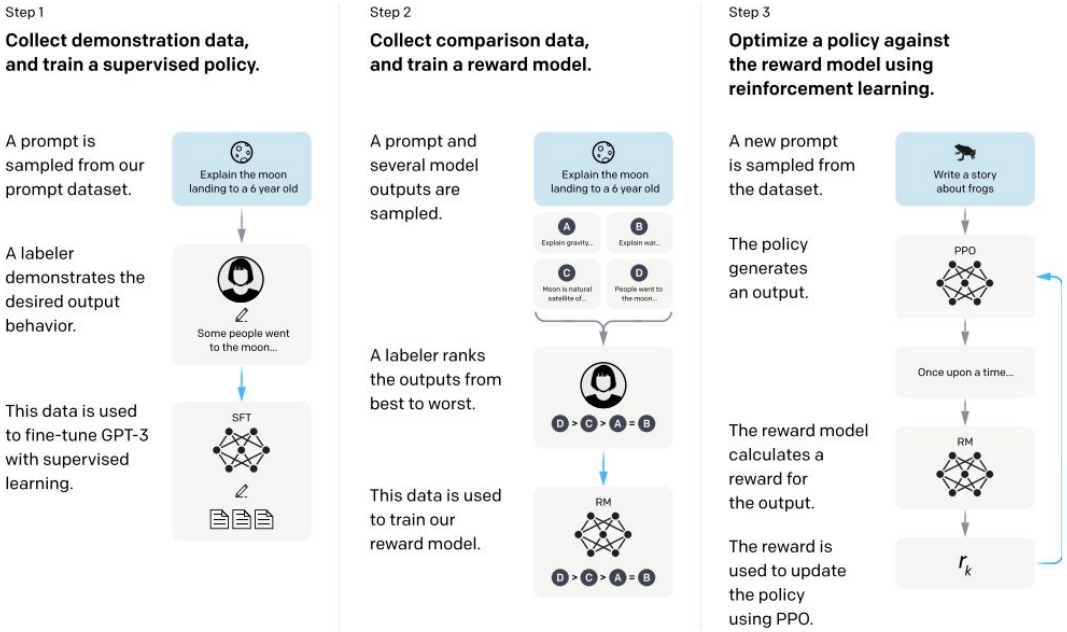


Method	Simulated Win-rate (%)	Human Win-rate (%)
GPT-4*†	79.0 ± 1.4	69.8 ± 1.6
ChatGPT*†	61.4 ± 1.7	52.9 ± 1.7
PPO	46.8 ± 1.8	55.1 ± 1.7
DPO	46.8 ± 1.7	-
Best-of-1024	45.0 ± 1.7	50.7 ± 1.8
Expert Iteration	41.9 ± 1.7	45.7 ± 1.7
SFT 52k	39.2 ± 1.7	40.7 ± 1.7
SFT 10k	36.7 ± 1.7	44.3 ± 1.7
Binary FeedME	36.6 ± 1.7	37.9 ± 1.7
Quark	35.6 ± 1.7	-
Binary Reward Conditioning	32.4 ± 1.6	-
Davinci001*	24.4 ± 1.5	32.5 ± 1.6
LLaMA 7B*	11.3 ± 1.1	6.5 ± 0.9

Nisan Stiennon & Al, 2020, Learning to summarize from human feedback

Dubois* & Al, 2024, Alpaca Farm: A Simulation Framework for Methods that Learn from Human Feedback

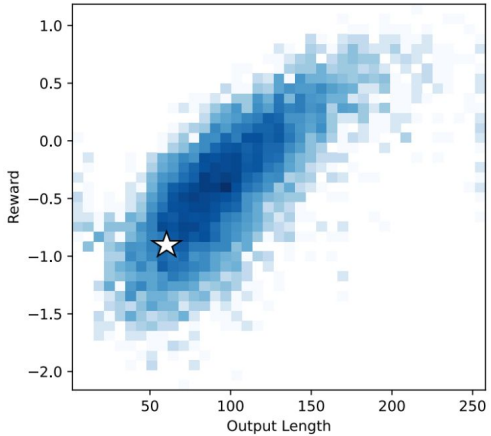
Chat GPT



Open AI, Aligning human preferences

RLHF challenges

- Answer preference is not trivial
- RLHF increases answer size
- Humans do not agree (agree with themselves only 66% of the time)
- Human have lot of variance, model have no variance
  - Ask LLM preferences instead of human preferences



Question: *Why don't adults roll off the bed?*

☆ **SFT (Before); 59 tokens**

Adults typically do not roll off of the bed because they have developed the muscle memory to keep their bodies from involuntarily moving during sleep and maintaining proper posture.

RLHF (After); 243 tokens: **Similar output, but much longer / more details**

Adults generally do not roll off of the bed because they have developed muscle control and awareness which helps them to remain upright while sleeping. Additionally, most adults find it uncomfortable or even painful to move around in their sleep, so rolling off the bed would be difficult without waking up first. There may also be safety concerns such as falling out of bed and potentially hurting oneself if one were to lose balance or fall from the ...

▸ **Annotation Guidelines**

Instruction

Tell me about self driving cars

example

Input

AI Response 1

Self driving cars are vehicles that are capable of detecting their surroundings and navigating without any input from a human driver. They use a variety of sensors, such as cameras, radar, and lidar, to detect objects in their environment and use advanced algorithms to make decisions about how to maneuver around them. Self driving cars are becoming increasingly popular and are being used in a variety of applications, such as ride-hailing services and autonomous delivery services.

AI Response 2

Self driving cars are cars that are equipped with sensors, cameras, and software, which allows them to navigate without the need for a driver. Self driving cars are able to monitor their environment and navigate without the assistance of a driver. Self driving cars use a combination of technologies such as GPS, radar, lidar, and computer vision to detect their environment and avoid obstacles. They are also able to update their maps and adjust their routes to account for variables such as traffic and construction.

The main benefit of self driving cars is their ability to reduce crashes and save lives. Autonomous vehicles have the potential to save up to 30,000 lives annually. They are also more energy efficient than traditional vehicles, as they do not require a driver who has to maintain a minimum speed. Autonomous vehicles also create new opportunities for increased mobility, allowing those who are unable to drive to get around.

Rating

- ○ Response 1 is better.
- ○ Response 1 is only slightly better. (only pick this if it's truly close)
- ○ Response 2 is only slightly better. (only pick this if it's truly close)
- ○ Response 2 is better.

Singhal & Al, 2024, A Long Way to Go: Investigating Length Correlations in RLHF

Stanford CS229 I Machine Learning I Building Large Language Models (LLMs) [Youtube]

Evaluation

How to evaluate a model like Chat GPT ?

- Different methods (DPO, PPO, SFT) can't be compared
- Models are not calibrated
- A large diversity of evaluation to cover

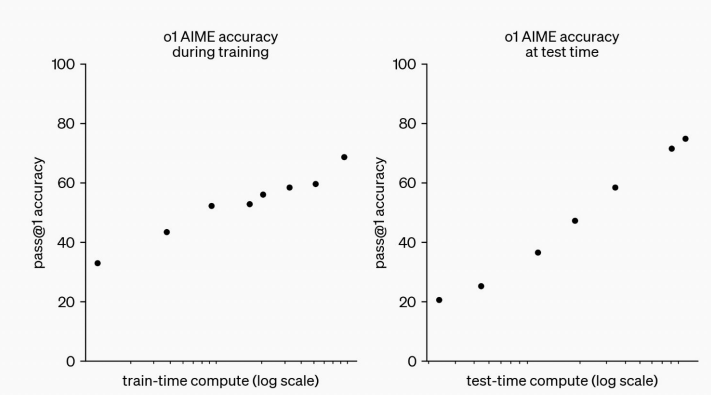
Model	Size	Arena Elo	MMLU	Context Window	License
o1-preview	72B +	1355	92.3	128K	Proprietary
ChatGPT-4o-latest	72B +	1335	88.7	128K	Proprietary
Llama-3.1-405b-Instruct	72B +	1266	88.6	128K	Llama 3.1 Community
Llama-3.1-70b-Instruct	40B-72B	1248	86	128K	Llama 3.1 Community
Gemma-2-27b-it	24B-40B	1217	75.2	8K	Gemma license
Llama-3.1-8b-Instruct	4B-24B	1171	73	128K	Llama 3.1 Community
Phi-3-Mini-4k-Instruct	1B-4B	1071	70.9	4K	MIT

Chatbot Arena, Open LM

OpenAI o1: “Streaming is dead, long live Chain of Thought”

- Chain of thought (COT)
- Increase test time compute

“Our large-scale reinforcement learning algorithm teaches the model how to think productively using its chain of thought in a highly data-efficient training process. We have found that the performance of o1 consistently improves with more reinforcement learning (train-time compute) and with more time spent thinking (test-time compute). The constraints on scaling this approach differ substantially from those of LLM pretraining, and we are continuing to investigate them.”



o1 performance smoothly improves with both train-time and test-time compute

Open AI, 2024, Learning to Reason with LLMs [Blog]

OpenAI o1 vs GPT 4o

Prompt:

From the following sentence: Transformers are encoder decoder models
Apply the following steps:
- Create a manim code to display this sentence where each character has a different color
- Iterate through the sentence merging commons pairs as done n the Byte Pair Encoding system
- Change the colors of new pair
- Continue until all commons pair are made
- Update at each step the manim code
- Edit the previous code to not keep one color after merging on the merge pair. The selected color should be the one with the highest number of letters
- Edit the code at the final stage to change color if two adjacent different pair have same color

Open AI o1

GPT 4o

Conclusion

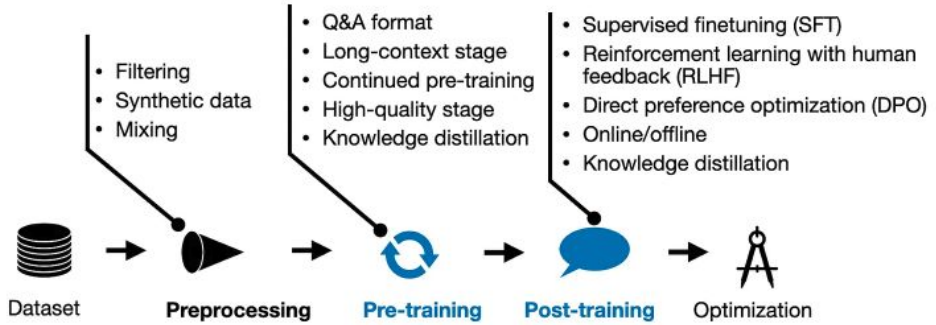
Building GPT 3

- Data preprocessing is a very important step to get quality data
- GPT 3 training consists of two phases (pre and post training)
- Supervised Fine Tuning is the first step of post training phase (ask a human to write the answer)
- RLHF helps the model to align with human preferences
- DPO is the new methods for Alignment, replacing RLHF

General knowledges

Data size and model size depends on compute resources (Scaling Laws, Chinchilla).
OpenAI o1 improves efficiency with longer RLHF training and answer inference time (COT)

"SFT+DPO approach seems to be the most popular preference tuning strategy at the moment due to the ease of use compared to other methods, such as RLHF with PPO."



An overview of the LLM development and training pipeline, with a focus on new pre-training and post-training methodologies discussed in this article

Sebastian Raschka, 2024, New LLM Pre-training and Post-training Paradigms [Blog]

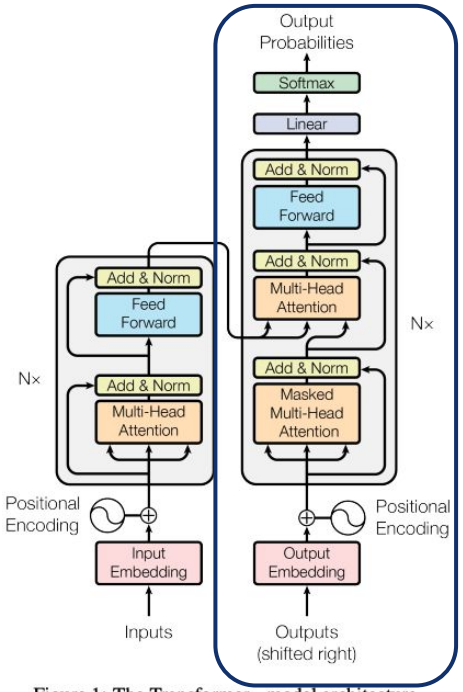
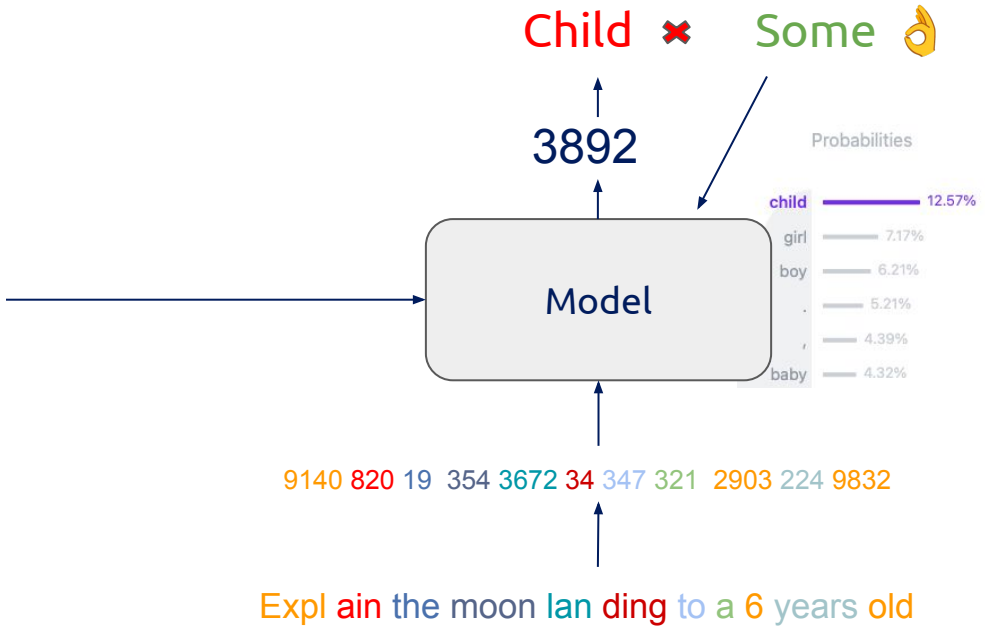


Figure 1: The Transformer - model architecture.



Go to : https://kahoot.com/
Click Play
Enter code: XXX XXX
Enter your name

II. Transformers

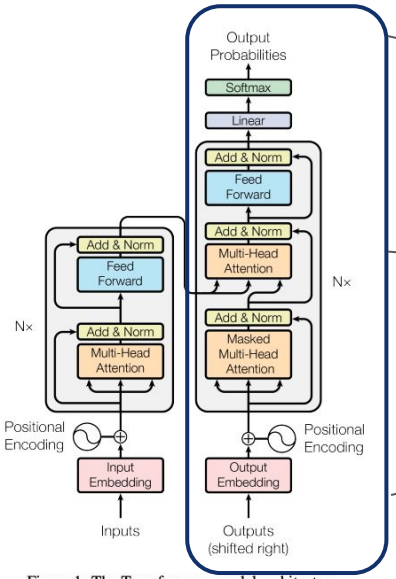
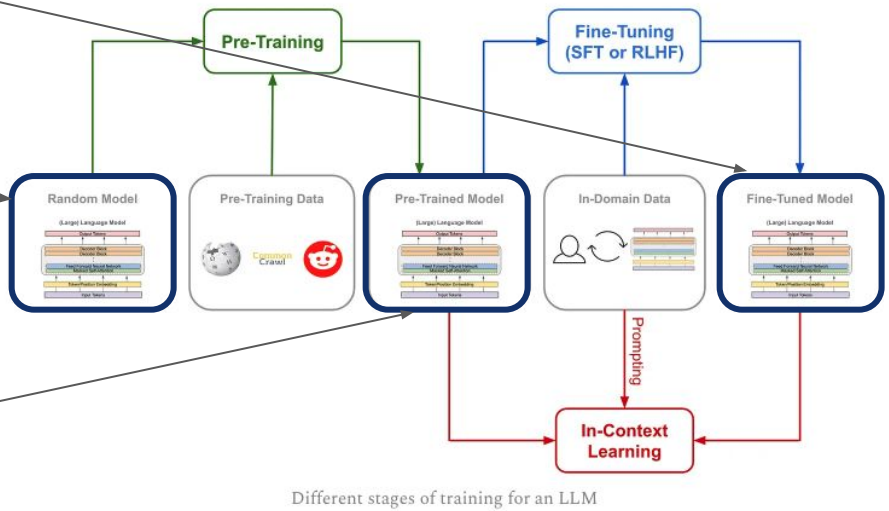


Figure 1: The Transformer - model architecture.



Different stages of training for an LLM

Dauphine | PSL TUNIS

II. Transformers

A. Before Transformers

1. N grams
2. Embeddings
3. RNN
4. LSTM

B. Transformers

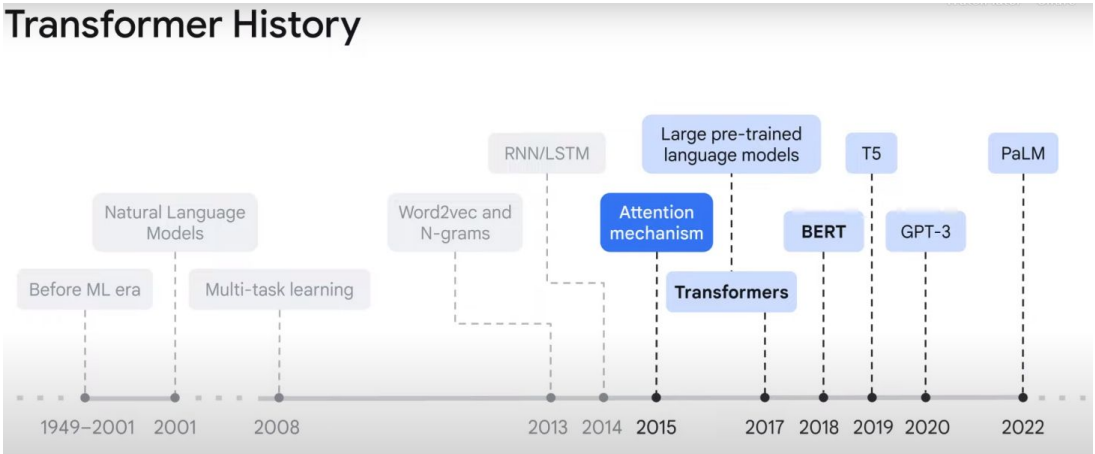
1. Self Attention / Cross Attention
2. Multi-Head Attention
3. Residual connection & Layer normalization
4. Feed forward layer
5. Softmax Layer
6. Positional Embeddings

Dauphine | PSL
TUNIS

II.A. Before Transformers

A. Before Transformers

1. N grams
2. Embeddings
3. RNN
4. LSTM



Dauphine | PSL TUNIS

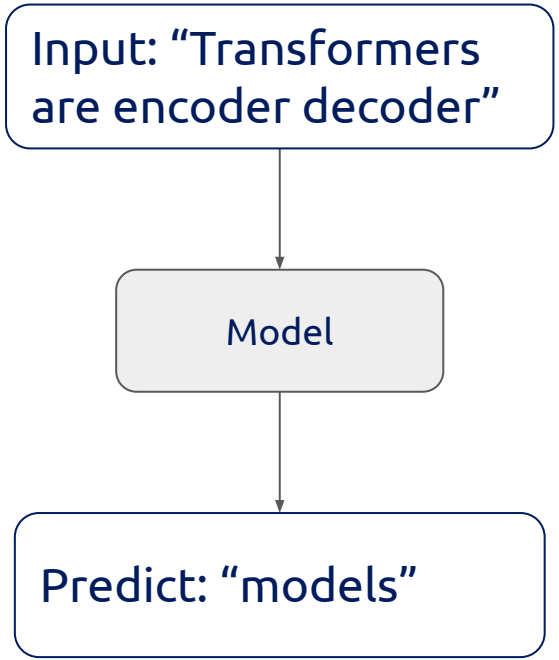
The Story of AI Evolution: Before ML Era to Transformers, GPT-3 and Beyond [LinkedIn]

Our goal today

Predict the word “models” from the input sentence

Requirements:

- Find a way to **transform** word into numerical values
- Provide **semantic relationship** between the encoding words
- Provide **context** to our model to understand the sentence
- Provide **long context** to our model to understand the sentence
- Create a **fast trainable** model



N Grams

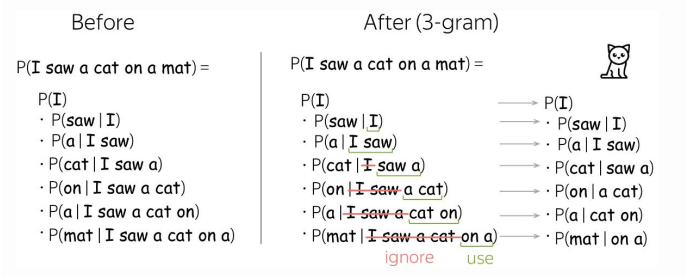
The chain rule of probability: $p(x_1, x_2, \ldots, x_n) = p(x_1)\, p(x_2 | x_1)\, p(x_3 | x_2, x_1) \ldots$

Input text:

To Sherlock Holmes she is always the woman. I have seldom heard him mention her under any other name. In his eyes she eclipses and predominates the whole of her sex. It was not that he felt any emotion akin to love for Irene Adler. All emotions, and that one particularly, were abhorrent to his cold, precise but admirably balanced mind. He was, I take it …

No.	N-gram	Frequency	Probability
1	of the	58	0.25
2	in the	37	0.24
3	it was	27	0.17
4	it is	27	0.17
5	to the	25	0.1
6	i was	24	0.09
7	at the	23	0.37
8	i have	23	0.09
9	i am	22	0.08
10	with a	21	0.31
11	of his	18	0.08

Lena Voita, Language Modeling [Blog]



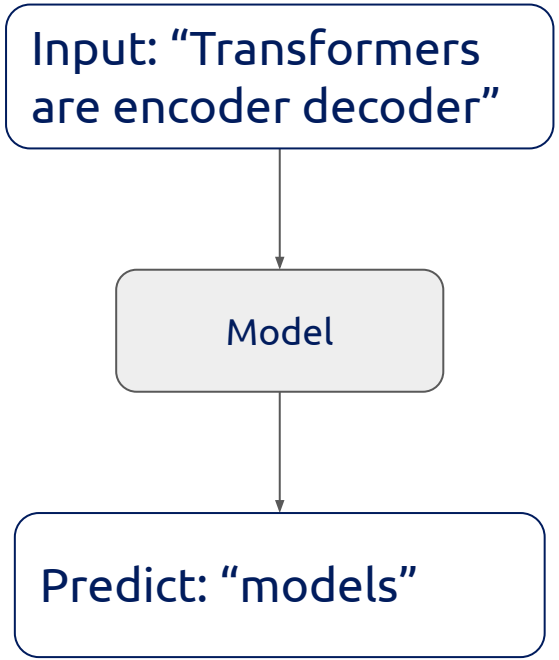
N-gram generator

Our goal today

Predict the word “models” from the input sentence

Requirements:

- Find a way to **transform** word into numerical values
- Provide **semantic relationship** between the encoding words
- Provide **context** to our model to understand the sentence
- Provide **long context** to our model to understand the sentence
- Create a **fast trainable** model

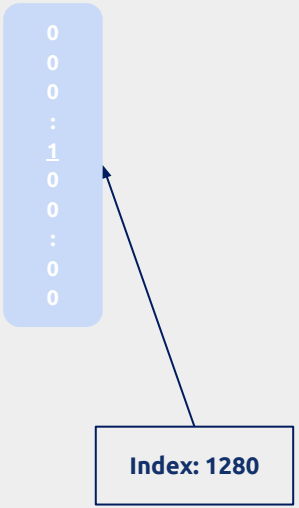


From **One-hot encoding** to **Word Embedding**

Index	Mot
0	a
1	the
2	he
…	…
1280	transformers
1281	embedding
1282	partial
…	…
34567	tunis
34568	dolphin

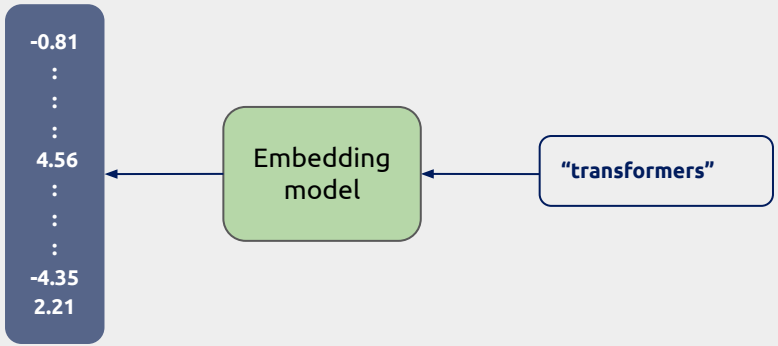
One-hot encoding

One-hot encoding
Dim = |Vocab Size|



Word Embedding

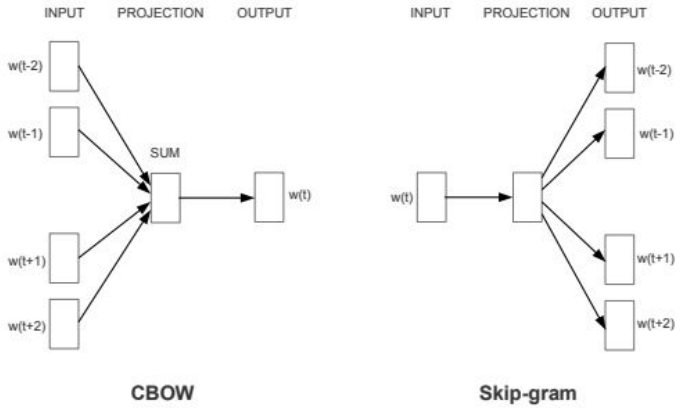
Semantic representation
Dim = |Chosen embedding size|



Word Embedding (Word2Vec, GloVe, BERT, ELMo)

Represent each word as a vector of numbers
Convert a *discrete* representation to *continuous*, allowing:

- More 'fine-grained' representations of words
- Useful computations such as cosine / euclidean distances
- Visualization and mapping of words



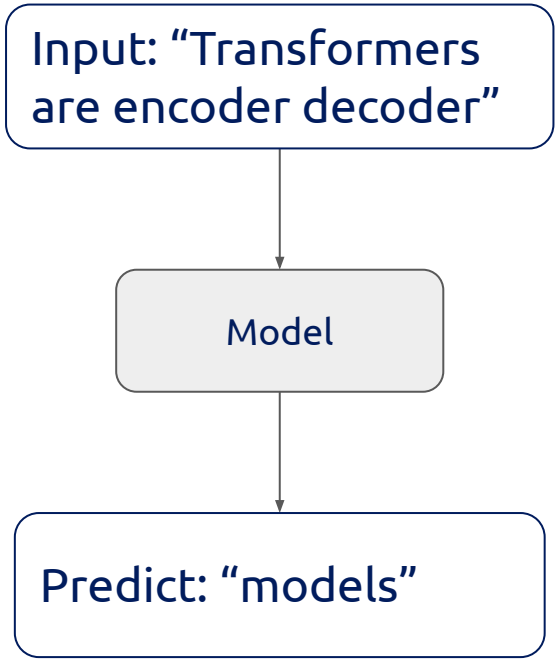
Tomas Mikolov, 2013, Efficient Estimation of Word Representations in Vector Space

Our goal today

Predict the word “models” from the input sentence

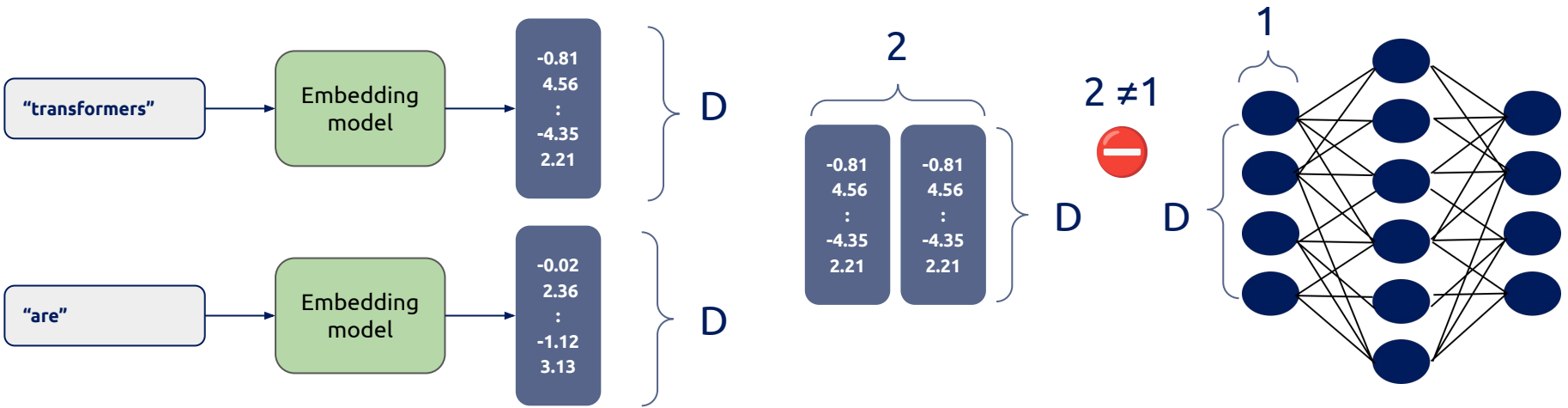
Requirements:

- Find a way to **transform** word into numerical values
- Provide **semantic relationship** between the encoding words
- Provide **context** to our model to understand the sentence
- Provide **long context** to our model to understand the sentence
- Create a **fast trainable** model



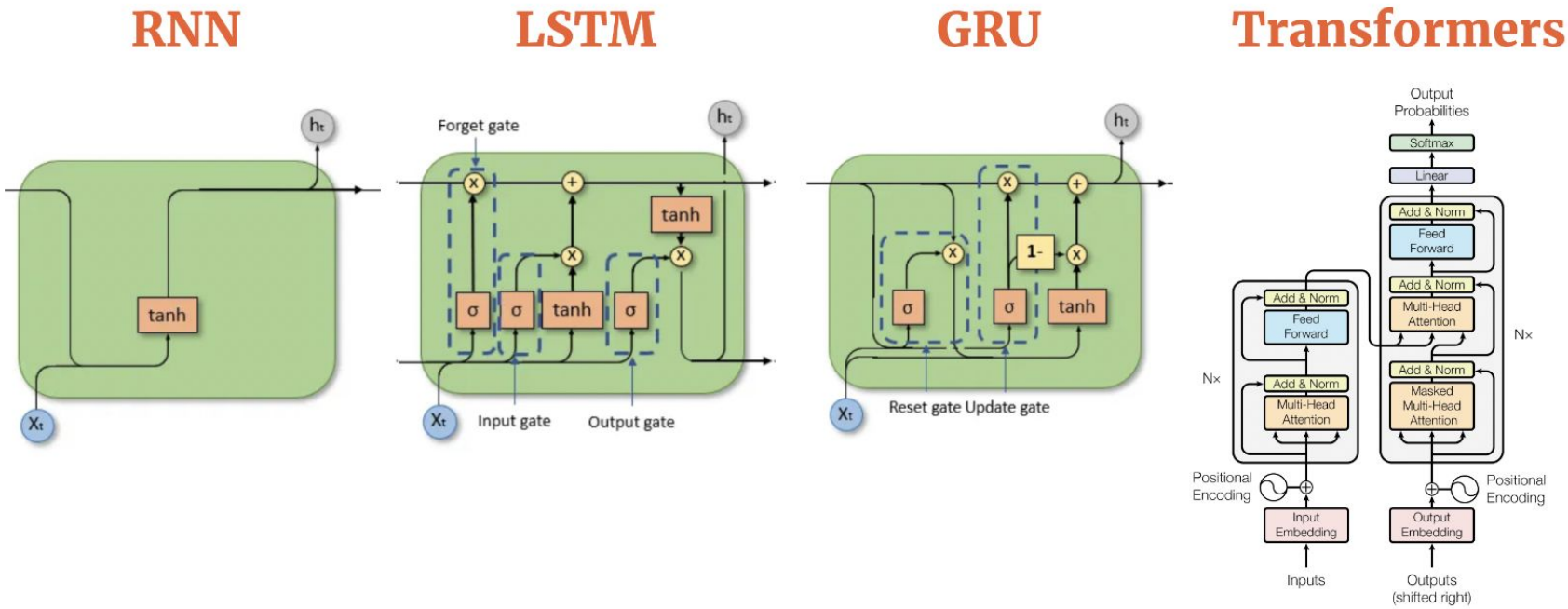
How to give a sentence to a model ?

Semantic representation
Dim = |Chosen embedding size| = 100



Dauphine | PSL TUNIS

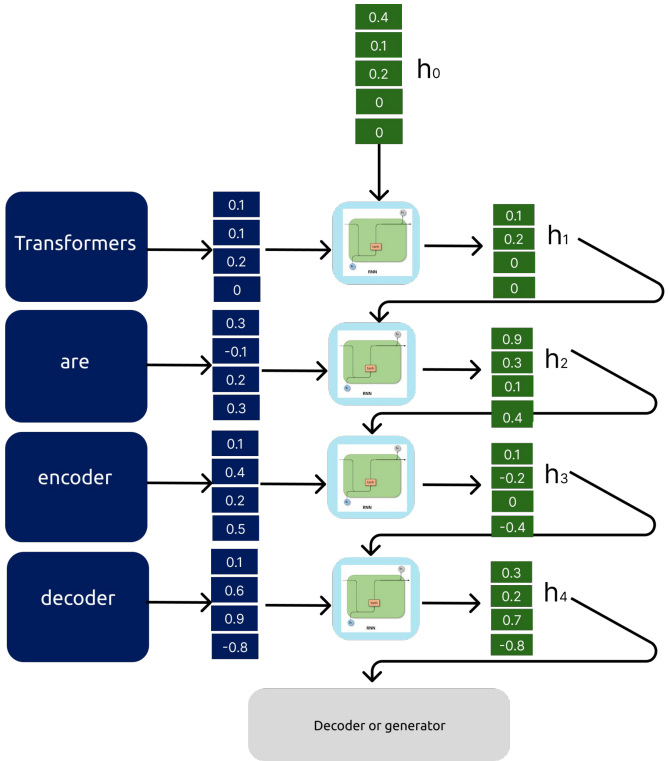
Seq2seq model



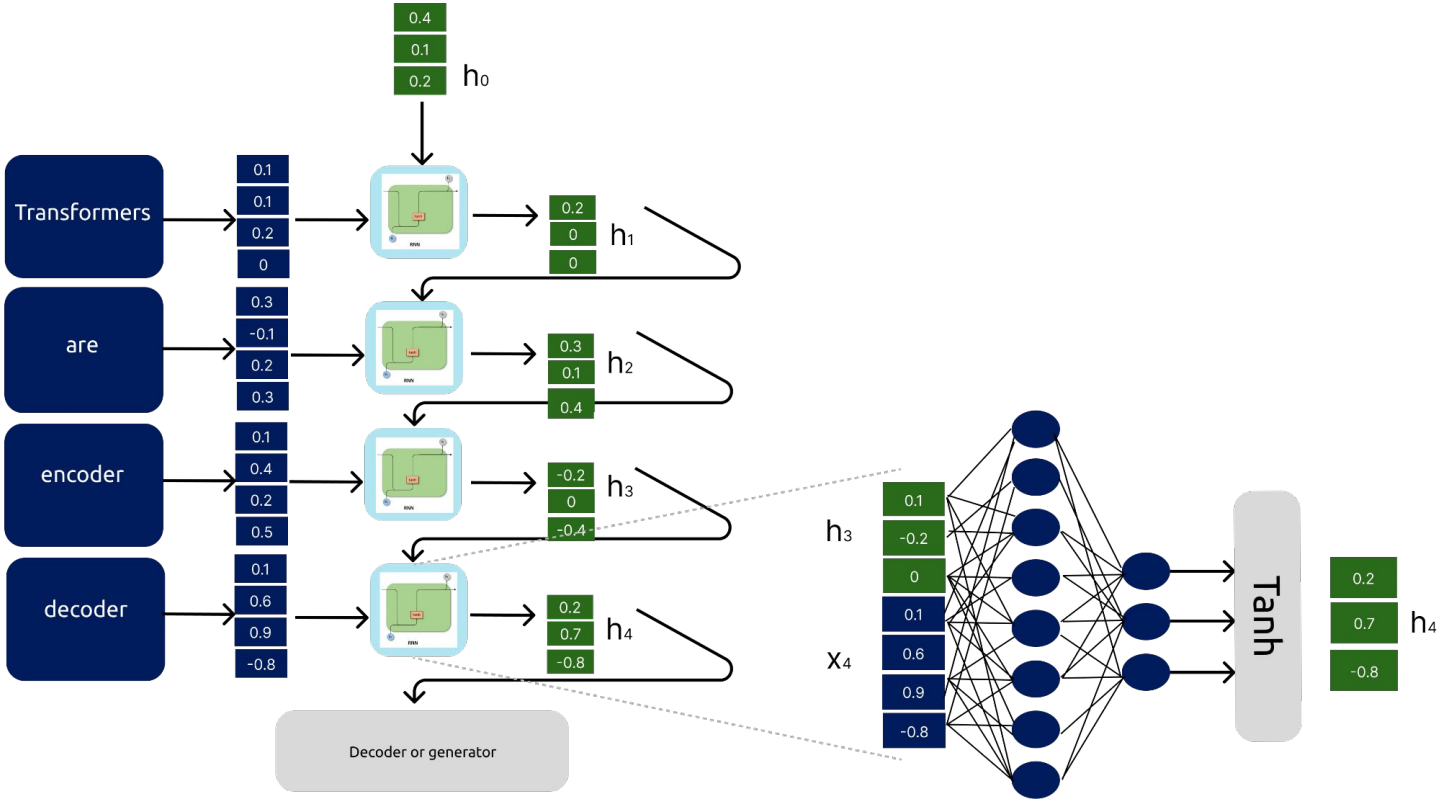
Recurrent Neural Networks (Sequential Model)

Advantages:
Can learn from context of previous word
Self supervised learning model

Problems:
Sequential model
Very short term memory

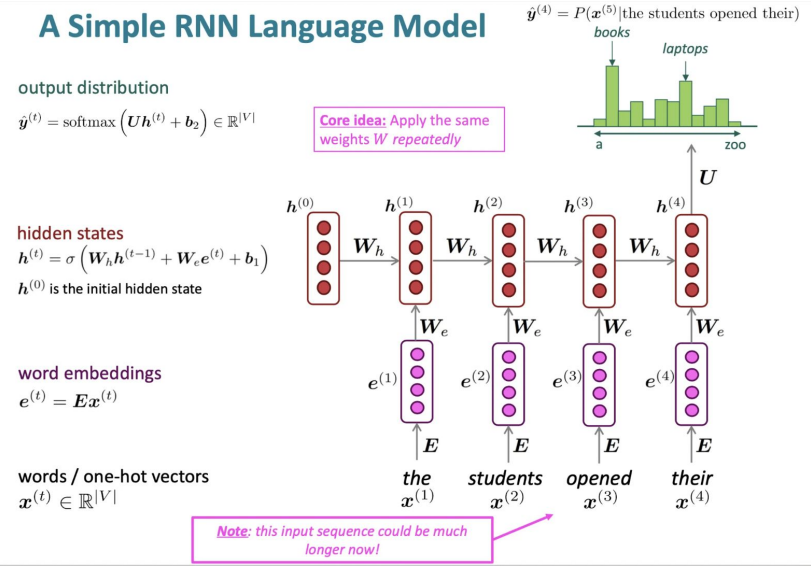


Recurrent Neural Networks (Seq2seq model)



Recurrent Neural Networks (Seq2seq model)

- Each word is given sequentially (x_t)
- An intern memory is updated after each word (h_t)
- A context is provided with this memory



Briefly describe the architecture of a RNN [Blog]

RNN limitations: Exploding / Vanishing gradient problem

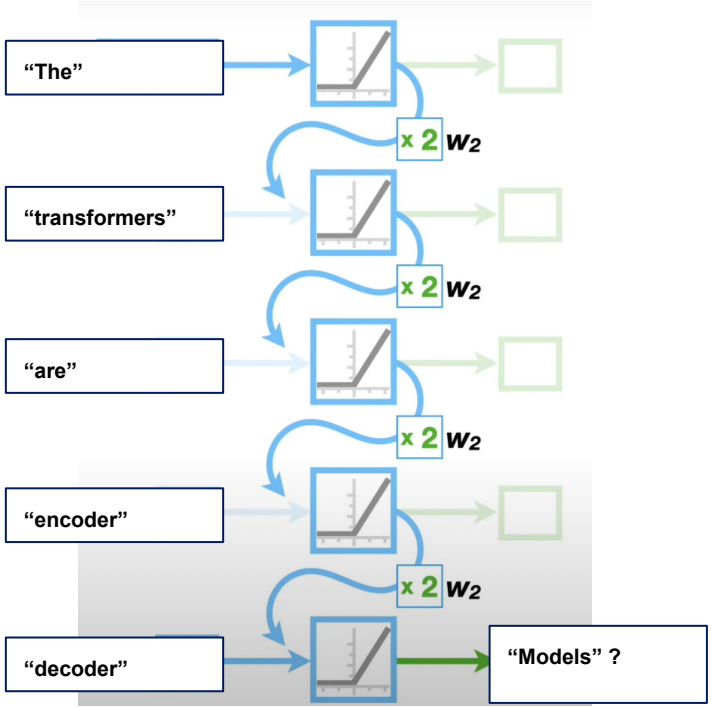
Optimizing the loss w.r.t weights:

$$\frac{\partial \mathbf{L}}{\partial W} = \sum_{i=0}^{T} \frac{\partial \mathcal{L}_i}{\partial W} \propto \sum_{i=0}^{T} \left(\prod_{i=k+1}^{v} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

$$\frac{\partial \mathbf{L}}{\partial W} \propto \sum_{i=0}^{T} \left(\prod_{i=k+1}^{v} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

1. ***Vanishing gradient*** $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 < 1$

2. ***Exploding gradient*** $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 > 1$



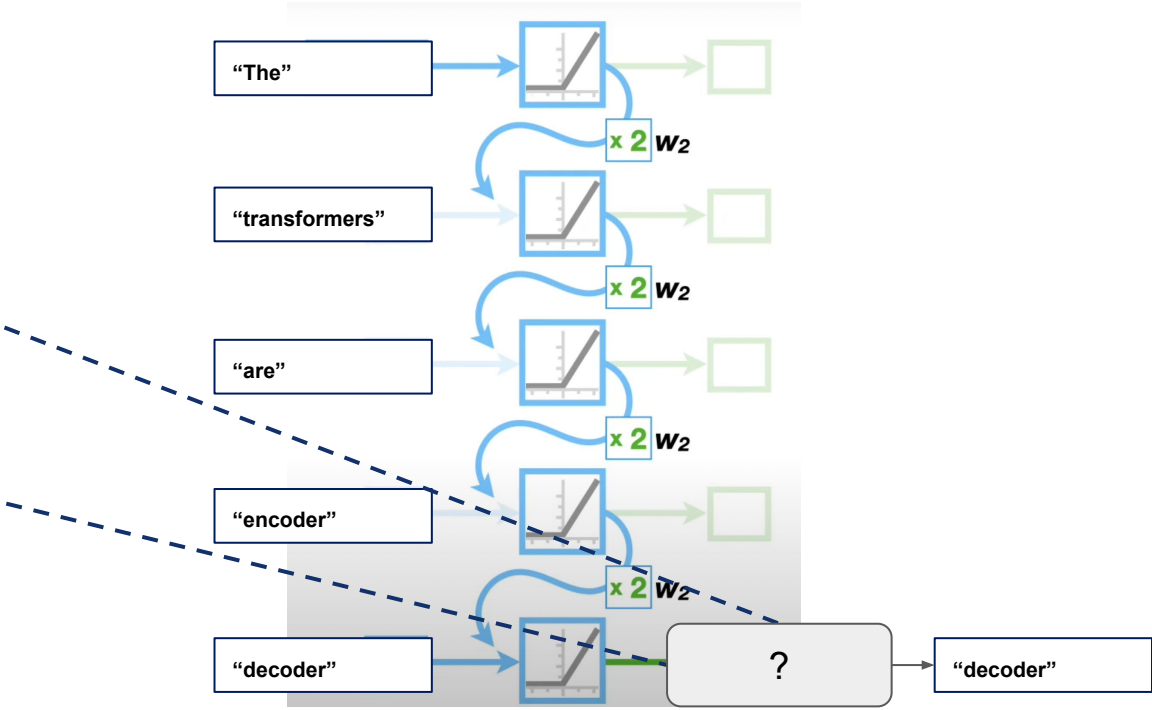
D. Barack Ore, 2020, The Exploding and Vanishing Gradients Problem in Time Series

StatQuest with Josh Starmer [Youtube]

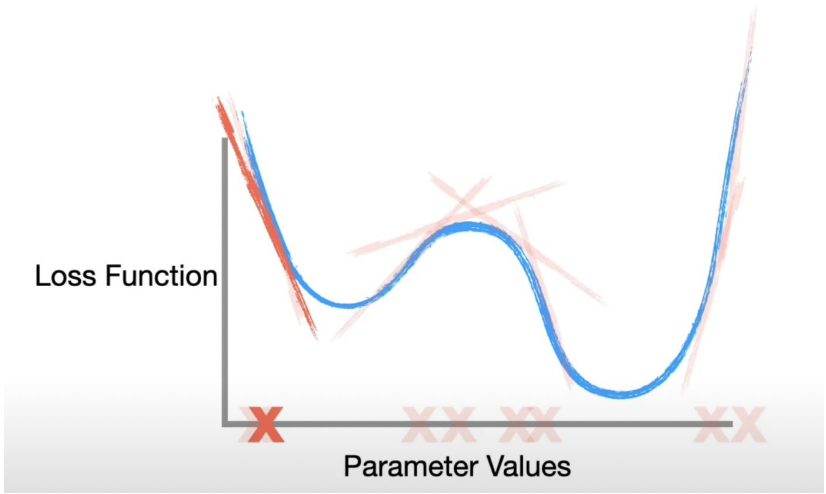
Dauphine | PSL TUNIS

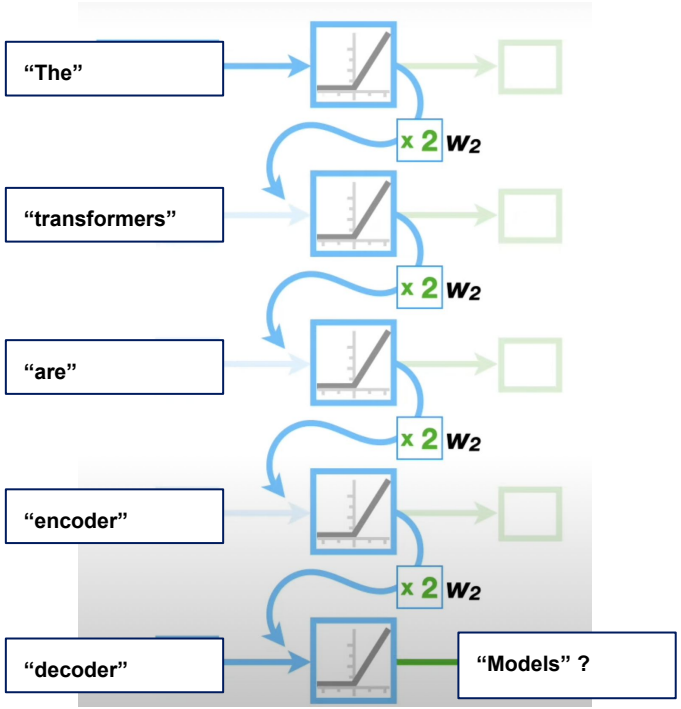
RNN limitations: Exploding / Vanishing gradient problem

Feed forward +
Softmax model



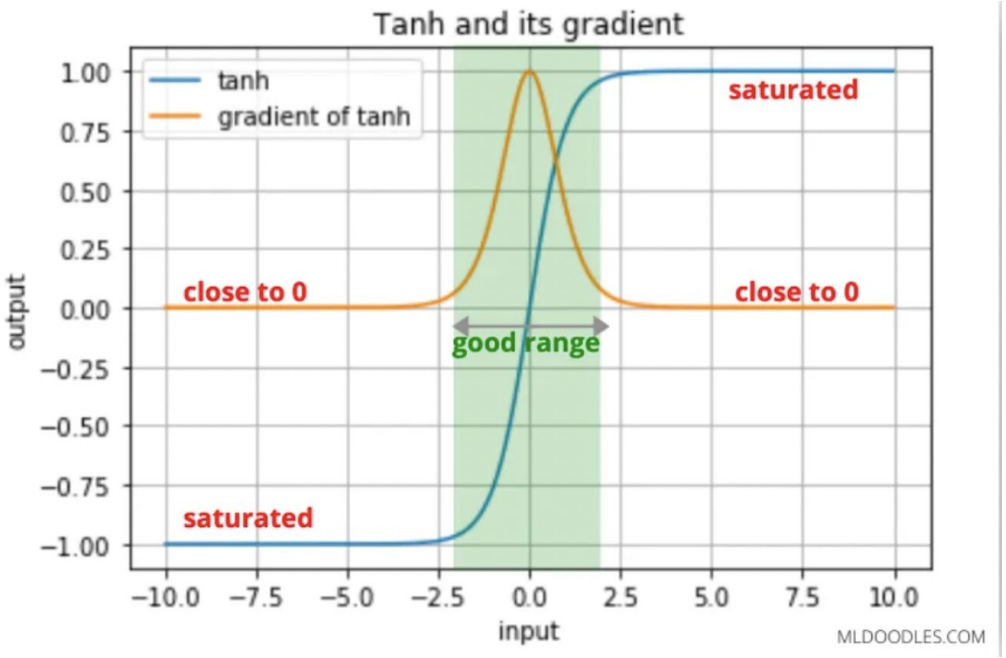
RNN limitations: Exploding / Vanishing gradient problem



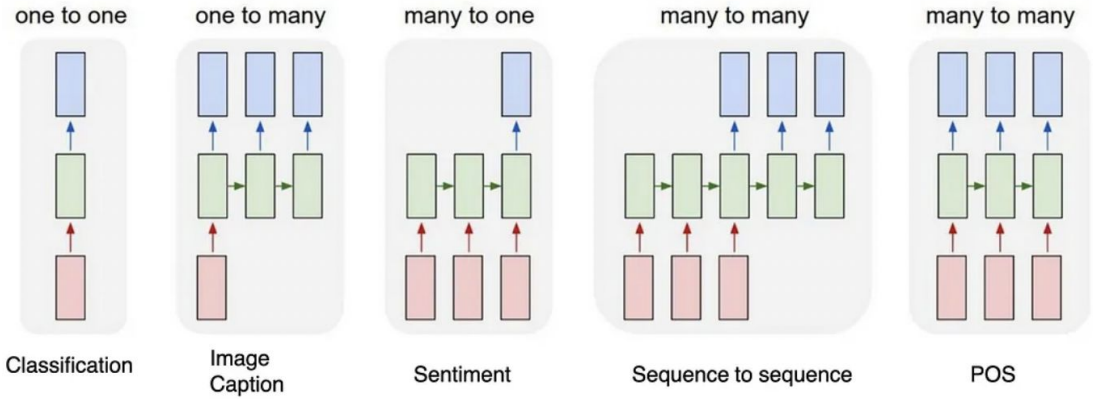


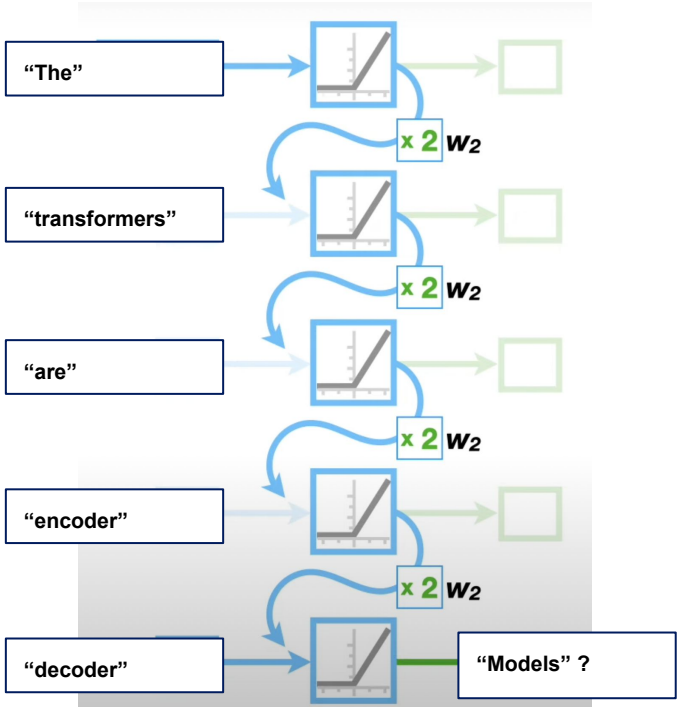
StatQuest with Josh Starmer Youtube

RNN limitations: Exploding / Vanishing gradient problem



RNN Types of architectures





Q. Which one fit our use case ?

Praveen Raj, 2023, Understanding Recurrent Neural Networks (RNN) — NLP

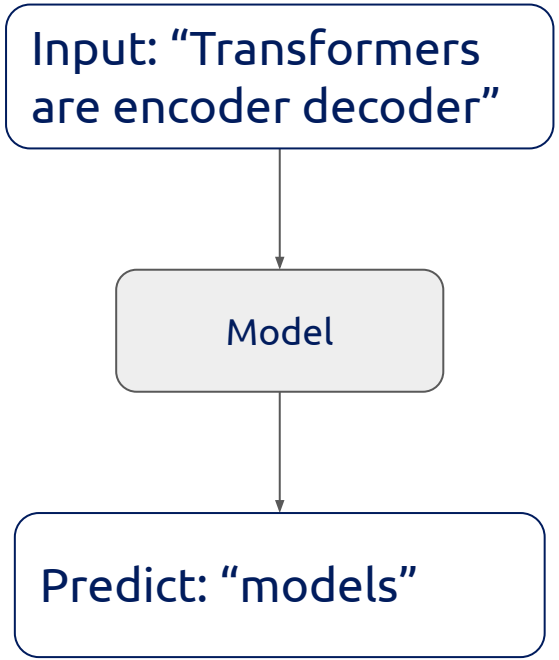
StatQuest with Josh Starmer Youtube

Our goal today

Predict the word “models” from the input sentence

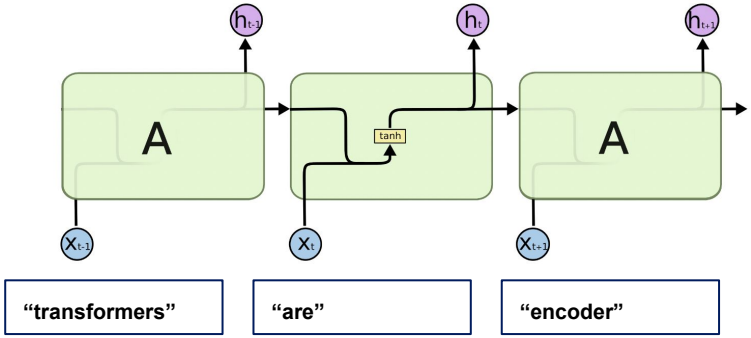
Requirements:

- Find a way to **transform** word into numerical values
- Provide **semantic relationship** between the encoding words
- Provide **context** to our model to understand the sentence
- Provide **long context** to our model to understand the sentence
- Create a **fast trainable** model

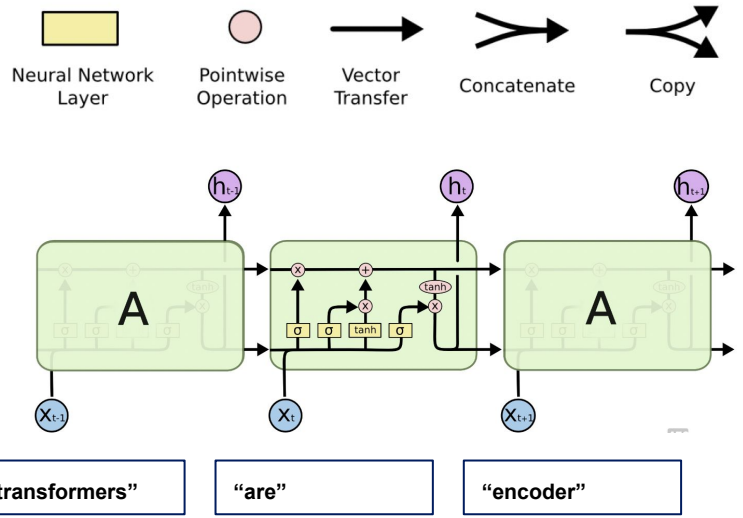


Long Short Term Memory (LSTM)

Both **long** and **short** term memory are provided



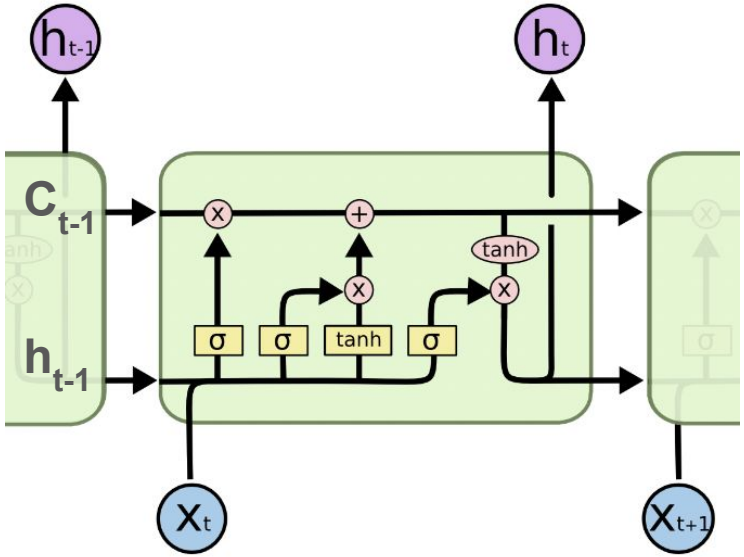
RNN architecture

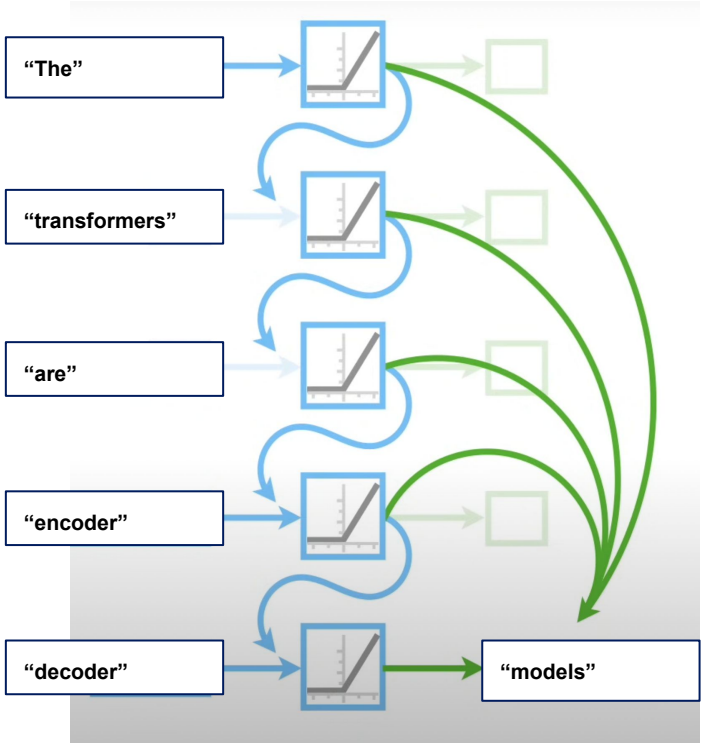


LSTM architecture

Colah, Understanding LSTM Networks [Blog]

Long Short Term Memory (LSTM)



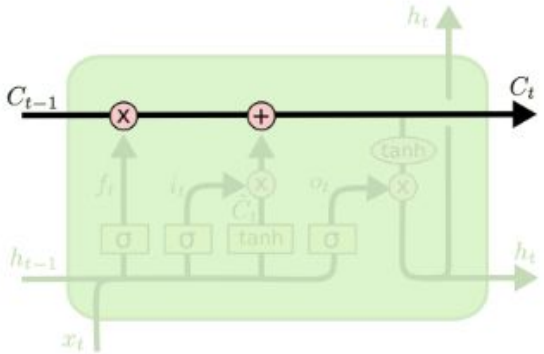


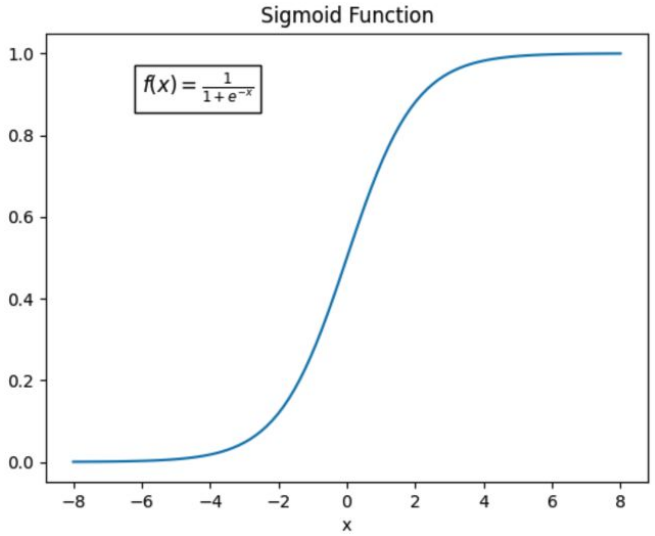
Colah, Understanding LSTM Networks [Blog]

Long Short Term Memory (LSTM)

Cell state to propagate long memory
Gate defined by the sigmoid function

- 0 = don't pass information
- 1 = let everything pass through

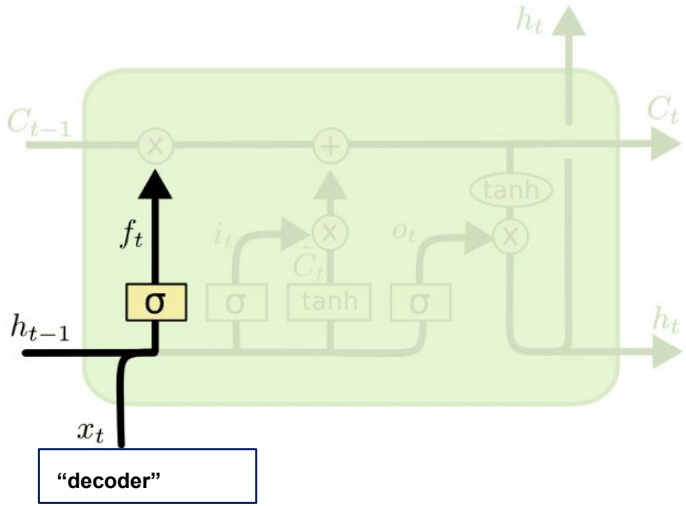




$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}}$$

Colah, Understanding LSTM Networks [Blog]

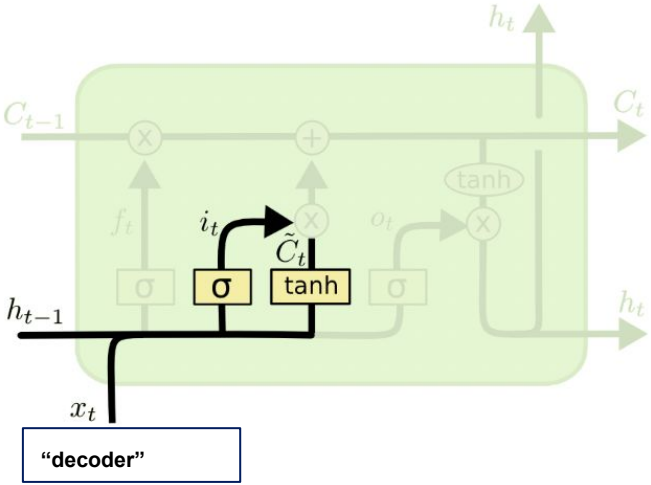
Long Short Term Memory (LSTM)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Colah, Understanding LSTM Networks [Blog]

Long Short Term Memory (LSTM)

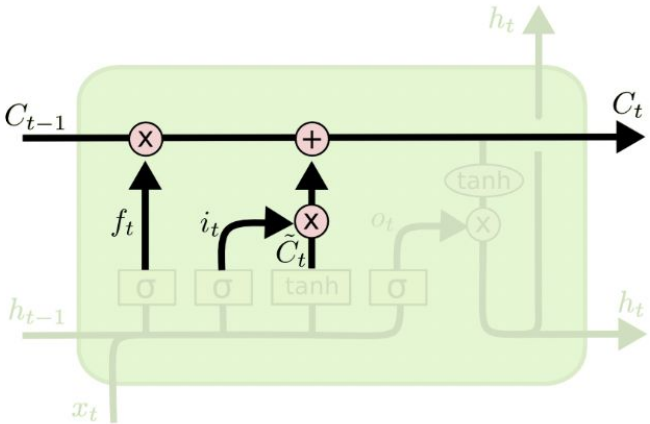


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Colah, Understanding LSTM Networks [Blog]

Long Short Term Memory (LSTM)

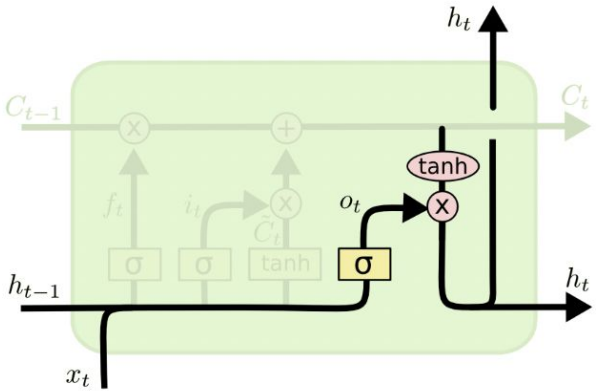


“decoder”

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Colah, Understanding LSTM Networks [Blog]

Long Short Term Memory (LSTM)



“decoder”

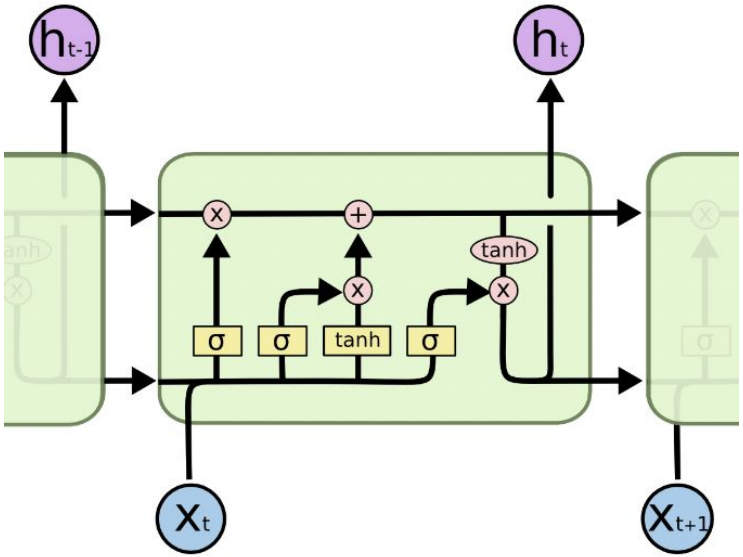
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Colah, Understanding LSTM Networks [Blog]

Long Short Term Memory (LSTM)

What about Vanishing / Exploding Gradients ?



Colah, Understanding LSTM Networks [Blog]

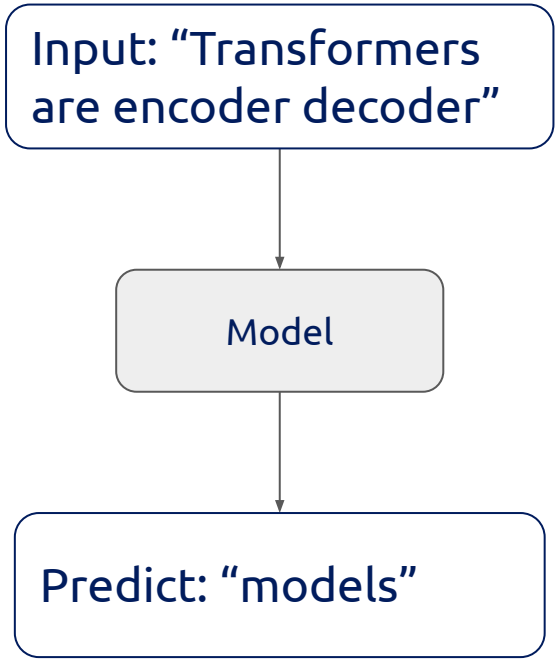
- The additive update function for the cell state gives a derivative that is much more 'well behaved'
- The gating functions allow the network to decide how much the gradient vanishes, and can take on different values at each time step. The values that they take on are learned functions of the current input and hidden state.
- To get details on LSTM derivative, check out this blog post

Our goal today

Predict the word “models” from the input sentence

Requirements:

- Find a way to **transform** word into numerical values
- Provide **semantic relationship** between the encoding words
- Provide **context** to our model to understand the sentence
- Provide **long context** to our model to understand the sentence
- Create a **fast trainable** model



Go to : https://kahoot.com/
Click Play
Enter code: XXX XXX
Enter your name

II.B. Transformers Architecture

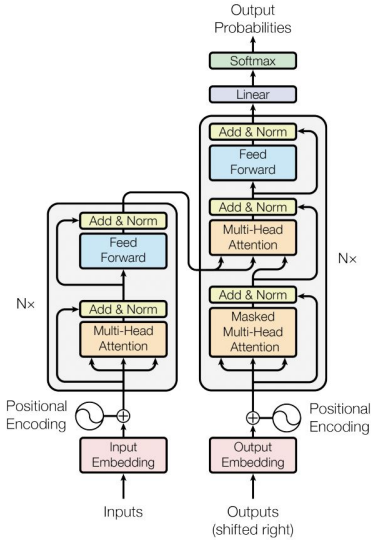


Figure 1: The Transformer - model architecture.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Dauphine | PSL TUNIS

II.B. Transformers Architecture

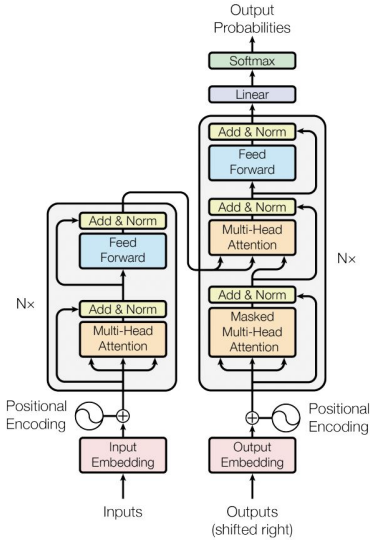


Figure 1: The Transformer - model architecture.

Introduction

1. Self Attention / Cross Attention
2. Multi-Head Attention
3. Residual connection & Layer normalization
4. Feed forward layer
5. Softmax Layer
6. Positional Embeddings

Dauphine | PSL TUNIS

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

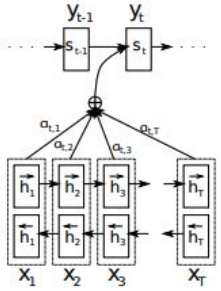
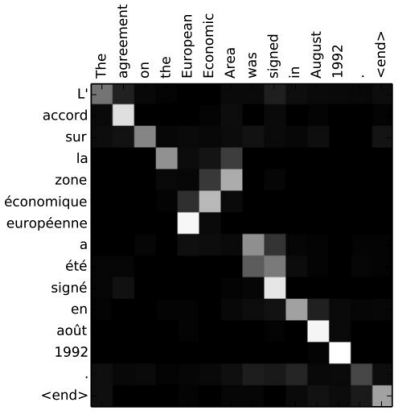


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.



The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{5}$$

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

where

$$e_{ij} = a(s_{i-1}, h_j)$$

The probability α_{ij}, or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden state s_{i-1} in deciding the next state s_i and generating y_i. Intuitively, this implements a mechanism of attention in the decoder. The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the

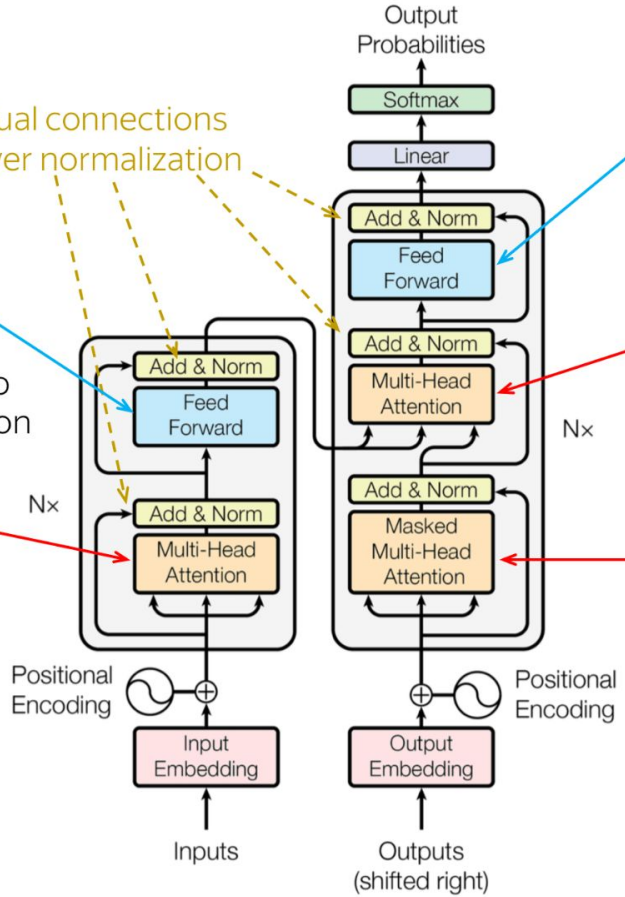
Bahdanau & Al, 2016, Neural Machine Translation by Jointly Learning to Align and Translate

Residual connections and layer normalization

Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Encoder self-attention: tokens look at each other

queries, keys, values are computed from encoder states



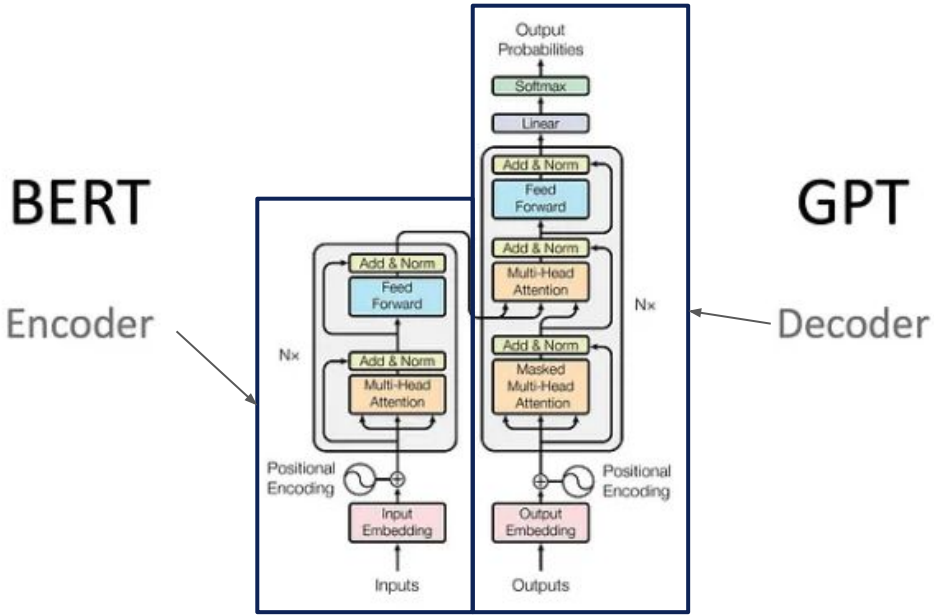
Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Decoder-encoder attention: target token looks at the source

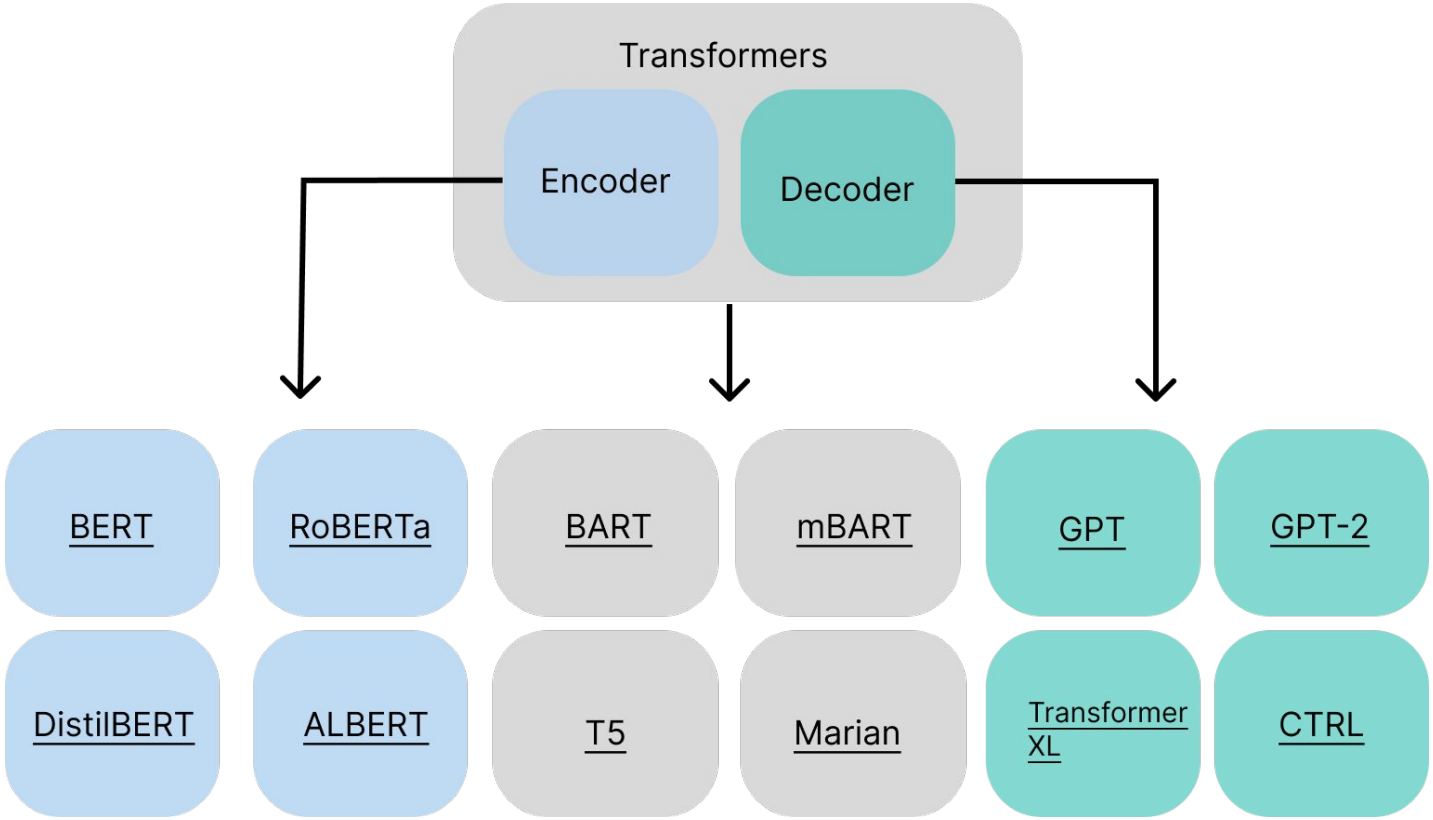
queries – from decoder states; keys and values from encoder states

Decoder self-attention (masked): tokens look at the previous tokens

queries, keys, values are computed from decoder states



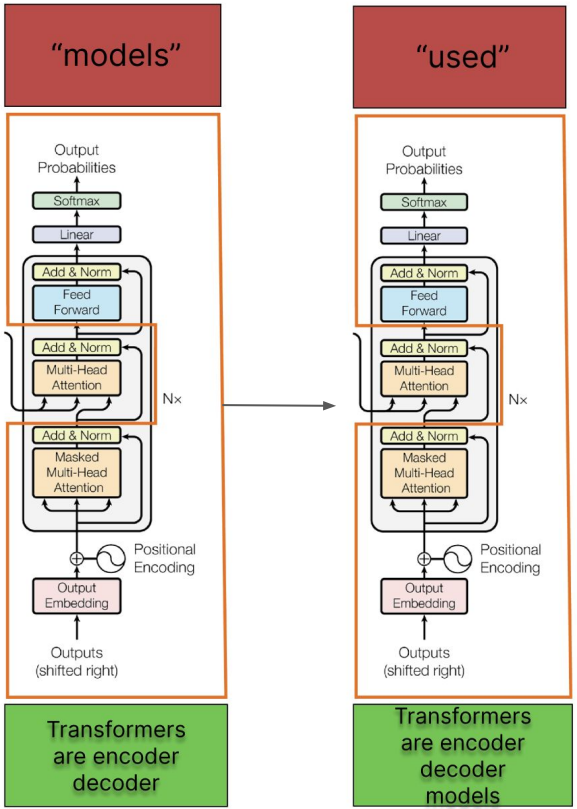
Transformer Architecture



Dauphine | PSL TUNIS

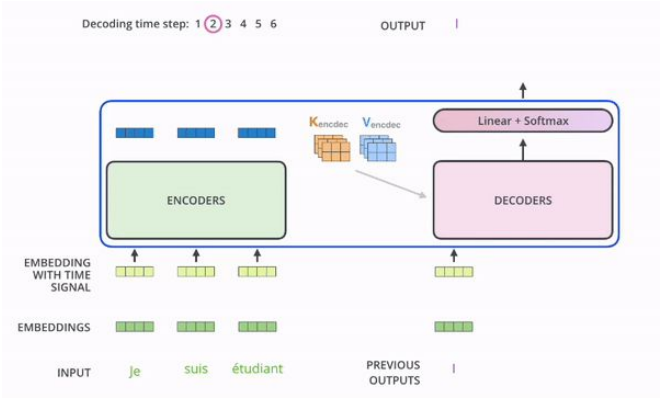
GPT 3

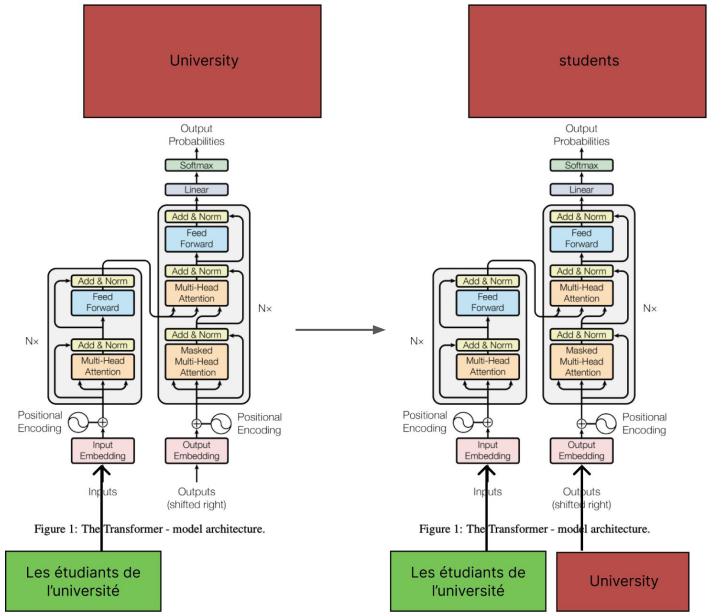
- Each word is generated one by one
- Only the decoder part is used



Translation model (FR -> EN example)

- The sentence to translate given to the encoder
- Each generated word added to the decoder





Jay Allamar, 2019, The Illustrated Transformer

Lot of new knowledges in this paper:

- No more RNN, only attention
- MLP layers and Attention
- Positional encodings
- ResNet structure
- Parallelism with Multi Head Attention

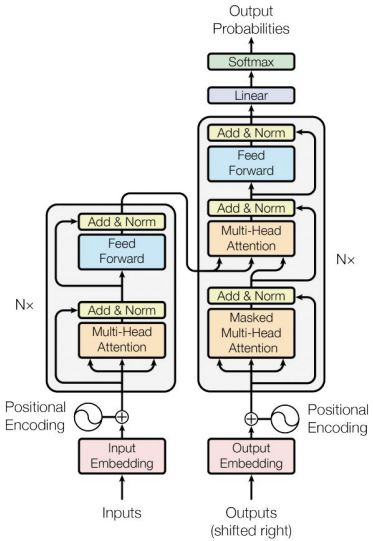


Figure 1: The Transformer - model architecture.

II.B. Transformers Architecture

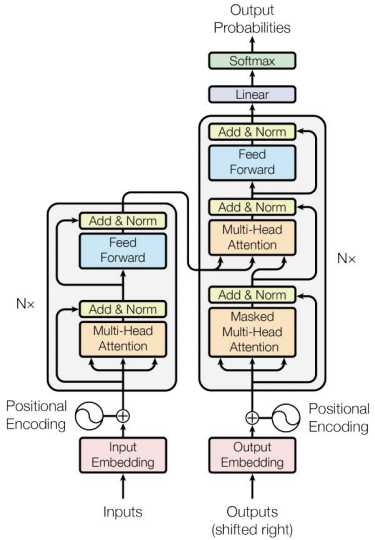


Figure 1: The Transformer - model architecture.

Introduction

1. **Self Attention / Cross Attention**
2. Multi-Head Attention
3. Residual connection & Layer normalization
4. Feed forward layer
5. Softmax Layer
6. Positional Embeddings

Dauphine | PSL TUNIS

II.B.1 Self Attention Mechanism

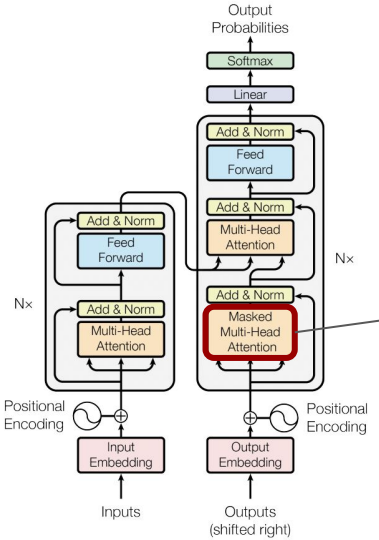
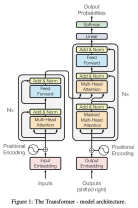


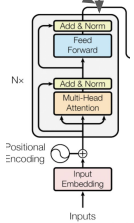
Figure 1: The Transformer - model architecture.

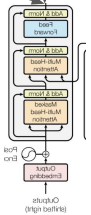
Transformers | are | encoder | decoder

France Travail

Attention mechanism



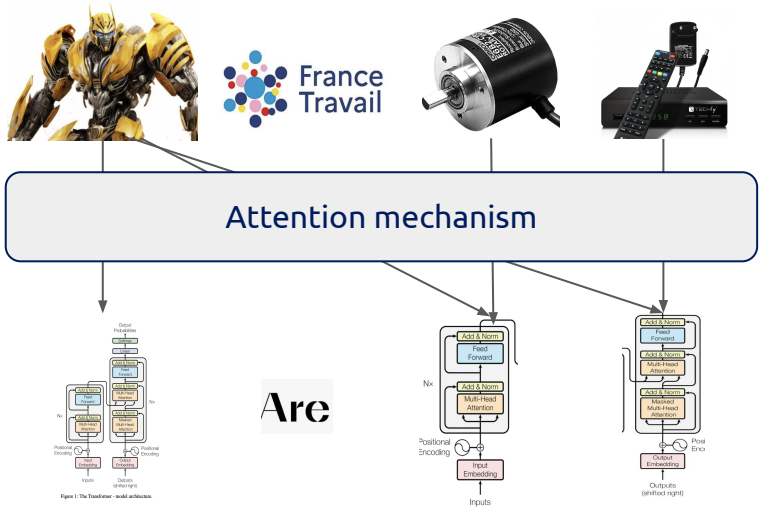




Dauphine | PSL TUNIS

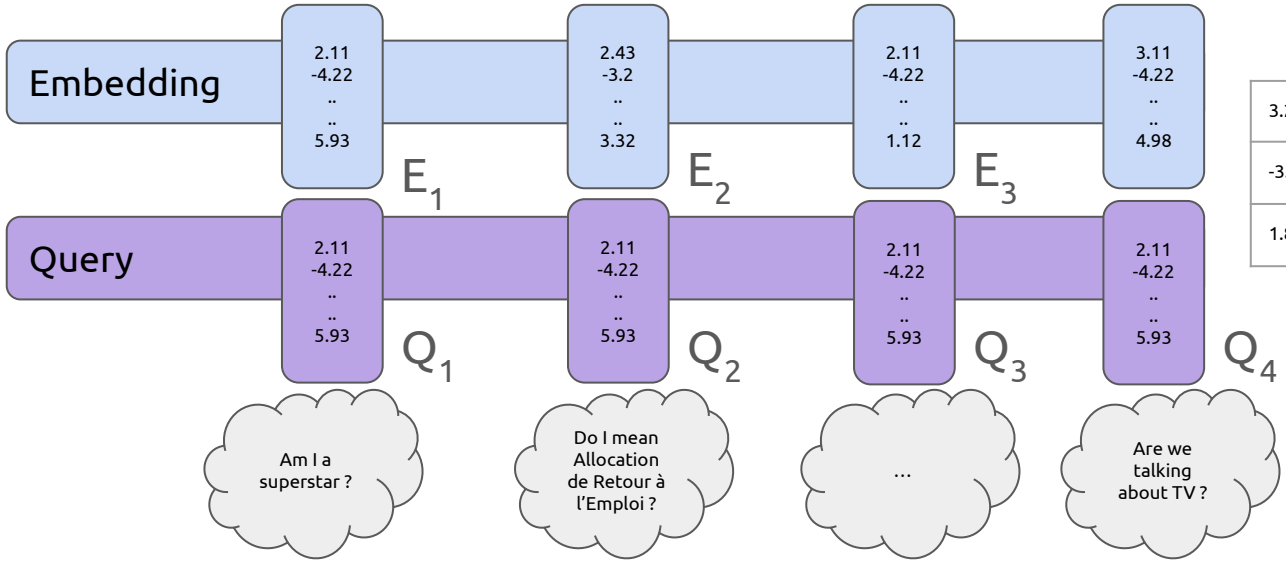
3 components:

Query: What am I looking for ?
Key: What do I have ?
Value: What do I reveal to others ?



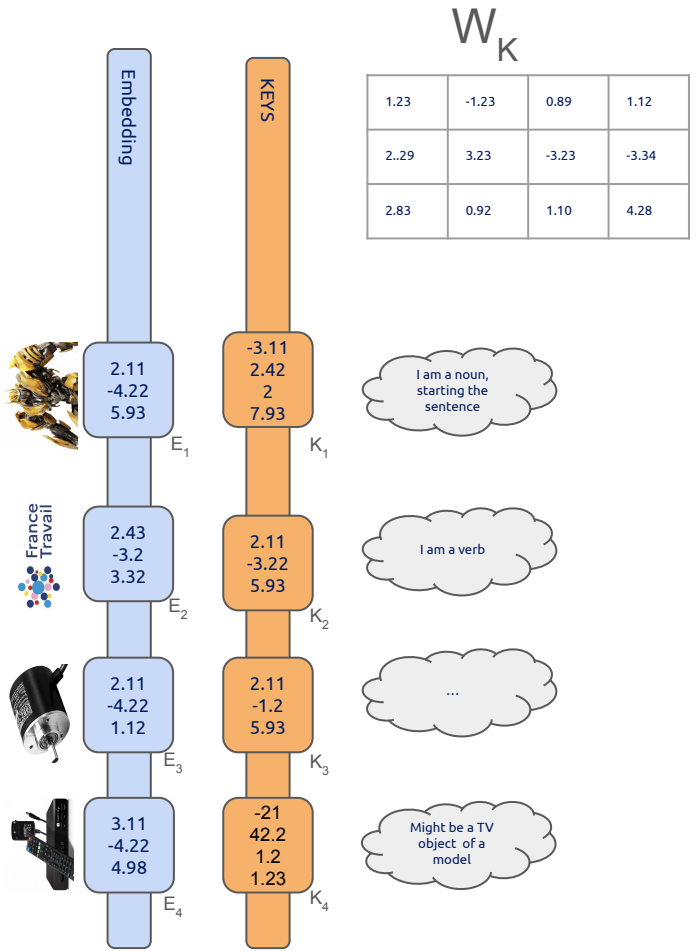
Transformers | are | encoder | decoder

Query: What am I looking for ?
$|E|$: Embedding (1, 12 288)
$|W_Q|$: Query matrix (12 288, 128)



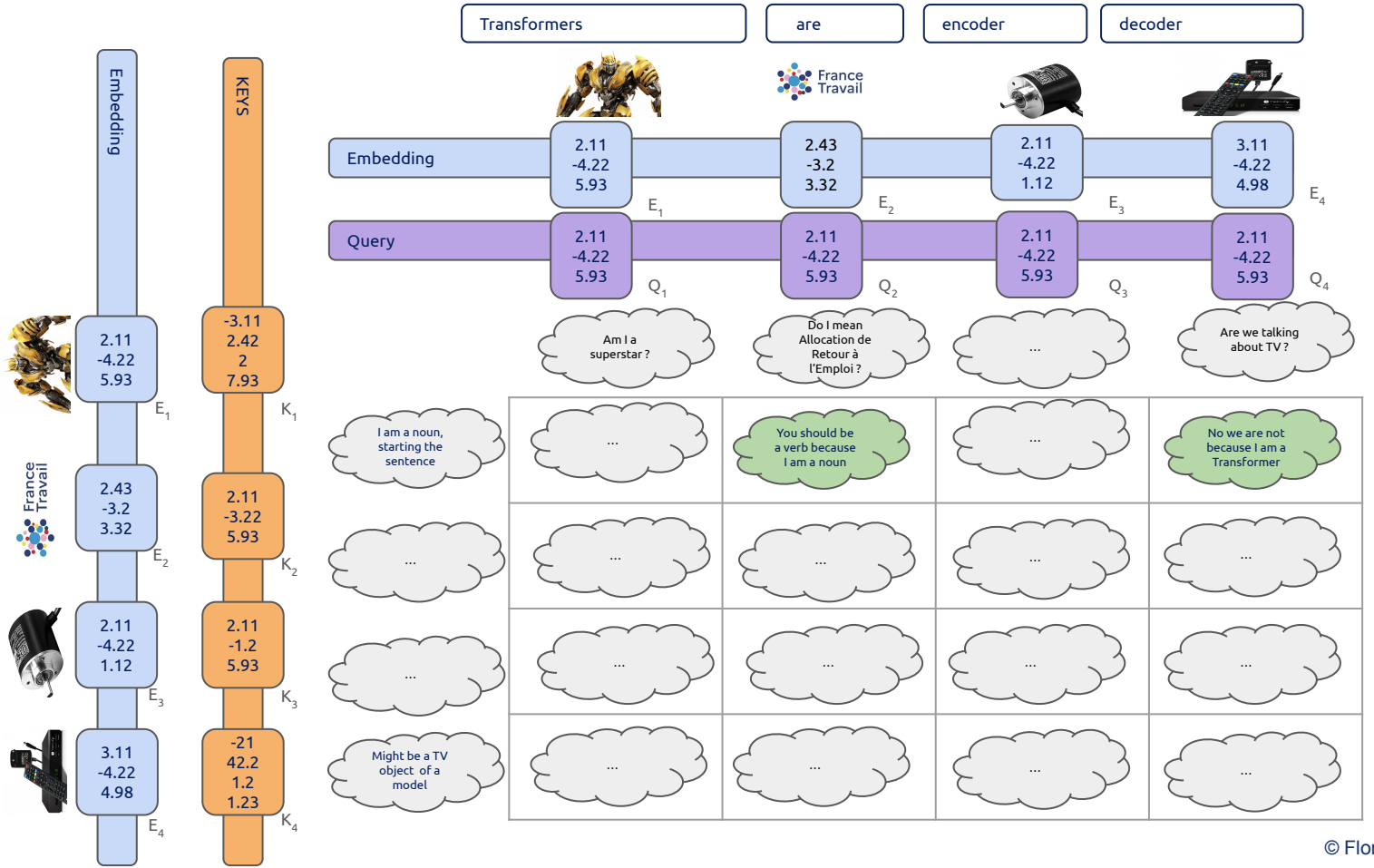
W_Q

3.23	-1.23	0.89	0.32
-3.29	3.23	1.23	-2.34
1.83	1.92	0.10	1.28



Key: What do I have ?
$|E|$: Embedding (1, 12 288)
$|W_K|$: Query matrix (12 288, 128)

II.B.1 Self Attention Mechanism

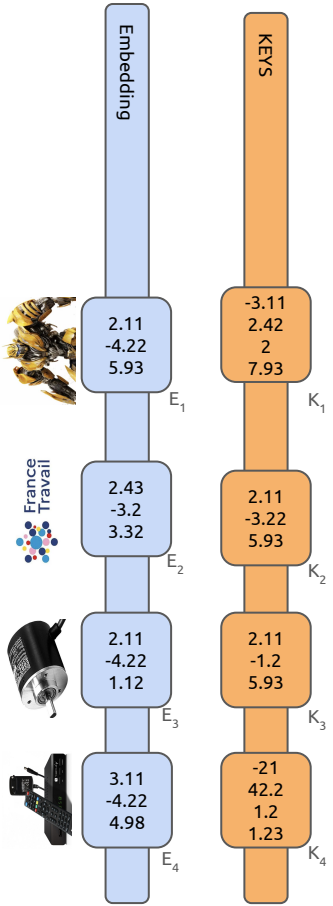


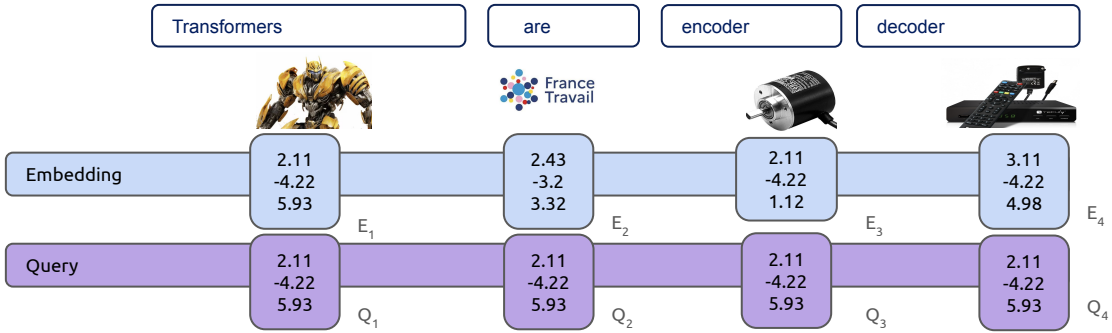
W_Q

3.23	-1.23	0.89	0.32
-3.29	3.23	1.23	-2.34
1.83	1.92	0.10	1.28

W_K

1.23	-1.23	0.89	1.12
2..29	3.23	-3.23	-3.34
2.83	0.92	1.10	4.28





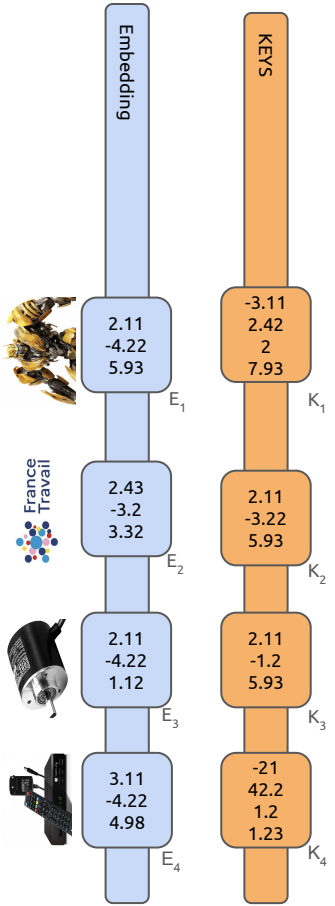
4	70	0	85
-4	-10	0	0
2	0	0	0
3	-3	4	5

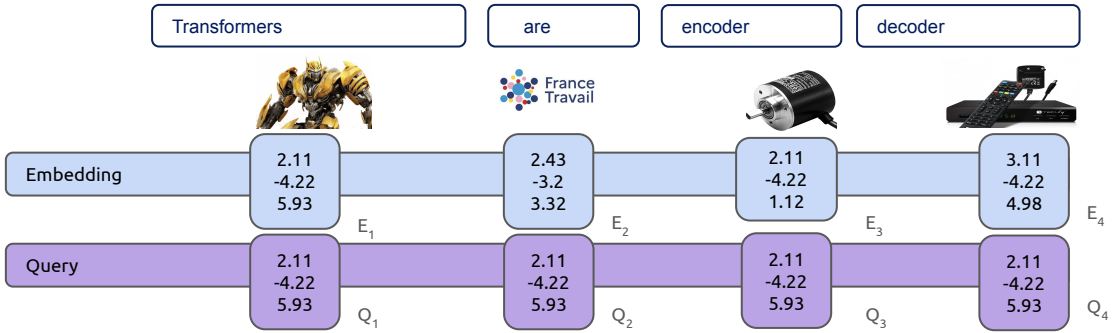
W_Q

3.23	-1.23	0.89	0.32
-3.29	3.23	1.23	-2.34
1.83	1.92	0.10	1.28

W_K

1.23	-1.23	0.89	1.12
2..29	3.23	-3.23	-3.34
2.83	0.92	1.10	4.28





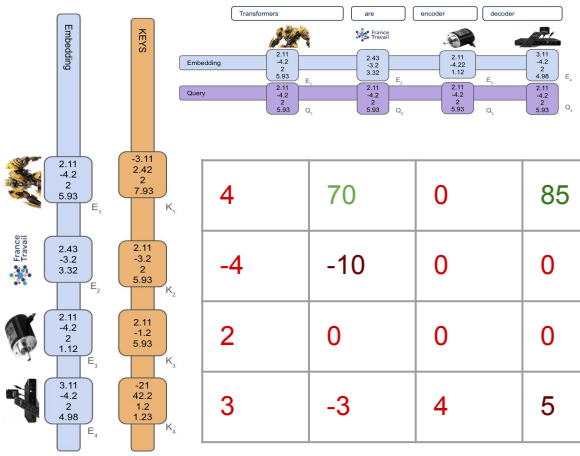
4	70	0	85
-4	-10	0	0
2	0	0	0
3	-3	4	5

W_Q

3.23	-1.23	0.89	0.32
-3.29	3.23	1.23	-2.34
1.83	1.92	0.10	1.28

W_K

1.23	-1.23	0.89	1.12
2..29	3.23	-3.23	-3.34
2.83	0.92	1.10	4.28



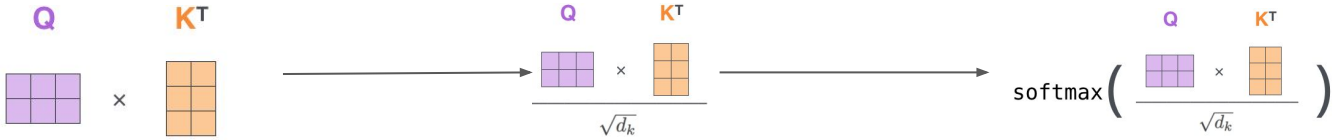
4	70	0	85
-4	-10	0	0
2	0	0	0
3	-3	4	5

This step allows numerical stability

4/128	70/128	0	85/128
-4/128	-10	0	0
2/128	0	0	0
3/128	-3/128	4/128	5/128

The sum of each column is 1

1	0.97	0.33	0.85
0	0.03	0.33	0.02
0	0	0.33	0.02
0	0	0	0.11



	Q_1	Q_2	Q_3	Q_4
K_1	$Q_1.K_1$	$Q_2.K_1$	$Q_3.K_1$	$Q_4.K_1$
K_2	$Q_1.K_2$	$Q_2.K_2$	$Q_3.K_2$	$Q_4.K_2$
K_3	$Q_1.K_3$	$Q_2.K_3$	$Q_3.K_3$	$Q_4.K_3$
K_4	$Q_1.K_4$	$Q_2.K_4$	$Q_3.K_4$	$Q_4.K_4$

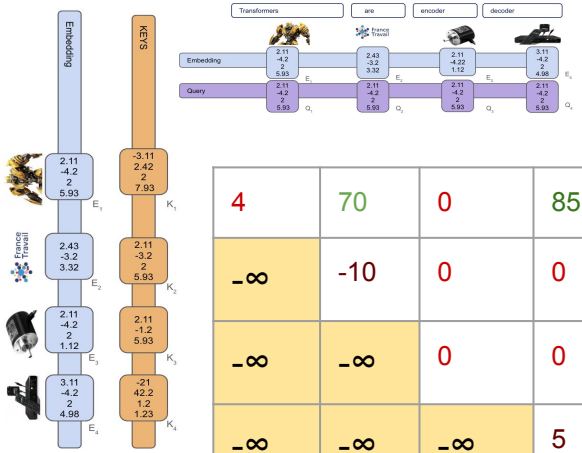
	Q_1	Q_2	Q_3	Q_4
K_1	$Q_1.K_1/\sqrt{d_k}$	$Q_2.K_1/\sqrt{d_k}$	$Q_3.K_1/\sqrt{d_k}$	$Q_4.K_1/\sqrt{d_k}$
K_2	$Q_1.K_2/\sqrt{d_k}$	$Q_2.K_2/\sqrt{d_k}$	$Q_3.K_2/\sqrt{d_k}$	$Q_4.K_2/\sqrt{d_k}$
K_3	$Q_1.K_3/\sqrt{d_k}$	$Q_2.K_3/\sqrt{d_k}$	$Q_3.K_3/\sqrt{d_k}$	$Q_4.K_3/\sqrt{d_k}$
K_4	$Q_1.K_4/\sqrt{d_k}$	$Q_2.K_4/\sqrt{d_k}$	$Q_3.K_4/\sqrt{d_k}$	$Q_4.K_4/\sqrt{d_k}$

Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Definition: the masking mechanism allows later words to not influence earlier words by setting lower left values by $-\infty$

Idea: A later word cannot answer question to a previous word because it is unknown at inference



4	70	0	85
$-\infty$	-10	0	0
$-\infty$	$-\infty$	0	0
$-\infty$	$-\infty$	$-\infty$	5

This step allows numerical stability

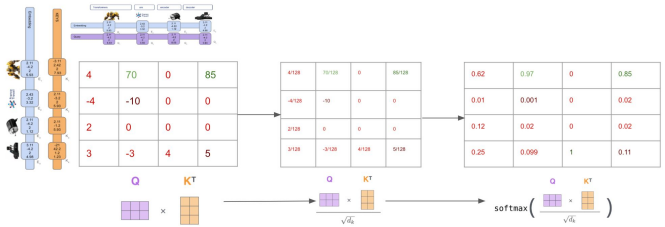
4/128	70/128	0	85/128
$-\infty$	-10	0	0
$-\infty$	$-\infty$	0	0
$-\infty$	$-\infty$	$-\infty$	5/128

1	0.97	0.33	0.85
0	0.03	0.33	0.02
0	0	0.33	0.02
0	0	0	0.11

$$Q \quad K^T$$

$$\frac{Q \quad K^T}{\sqrt{d_k}}$$

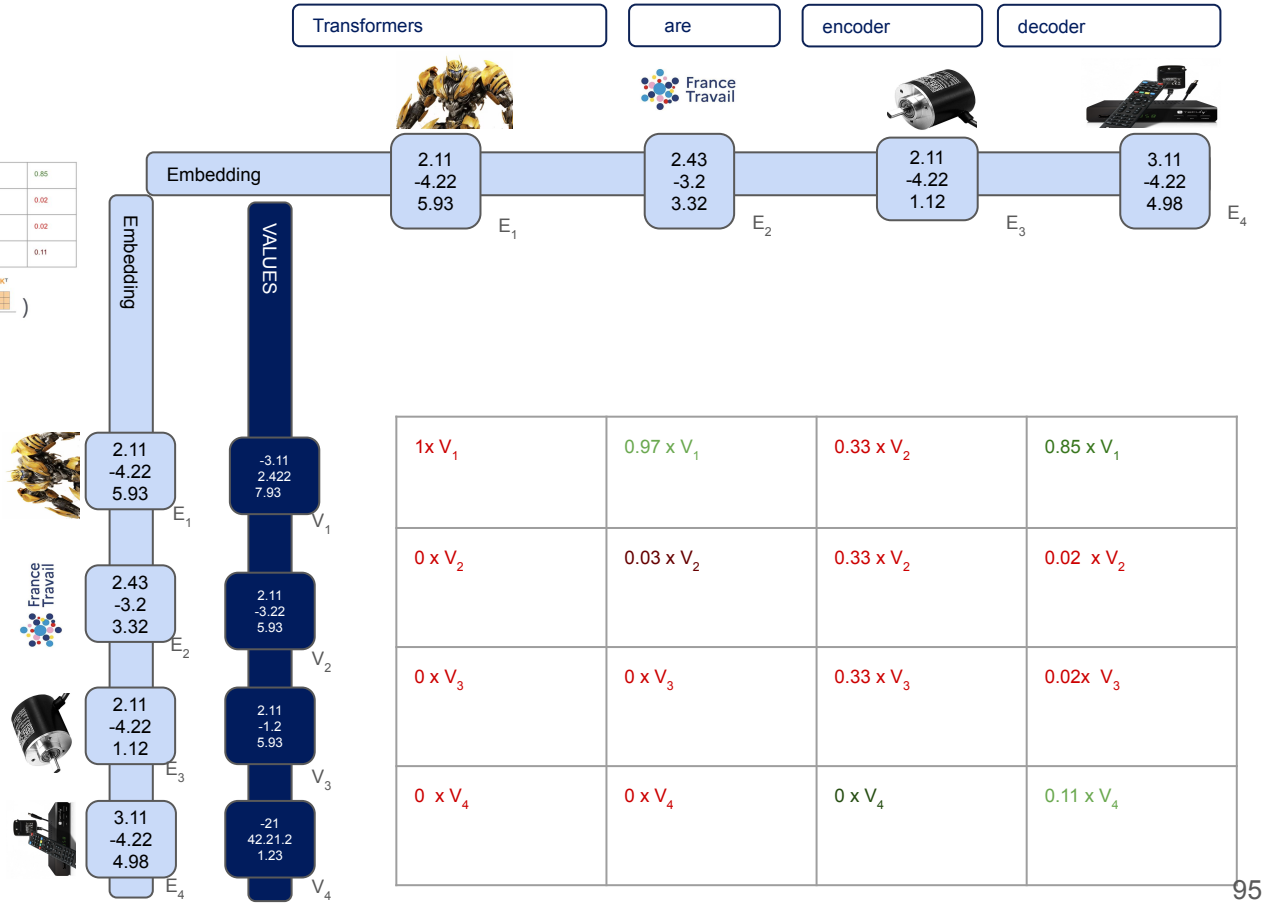
$$\text{softmax}\left(\frac{Q \quad K^T}{\sqrt{d_k}}\right)$$



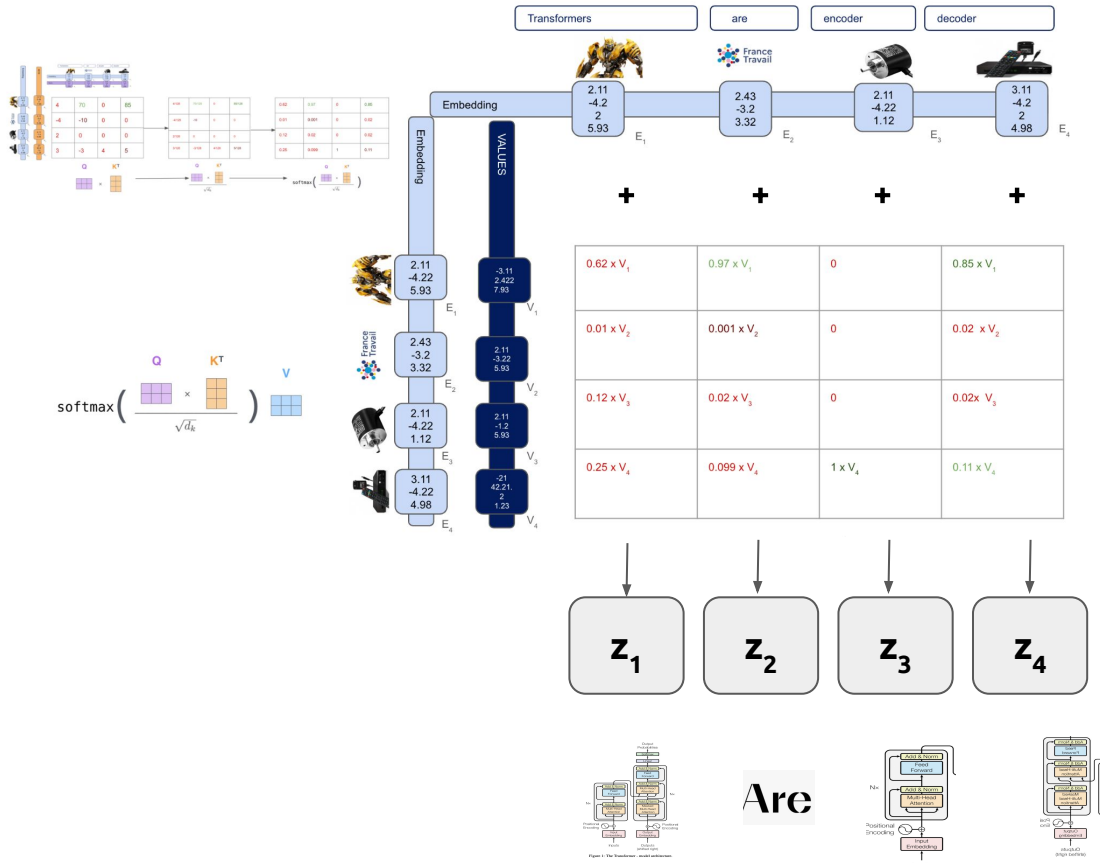
W_V

1.23	-1.23	0.89	1.12
2..29	3.23	-3.23	-3.34
2.83	0.92	1.10	4.28

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



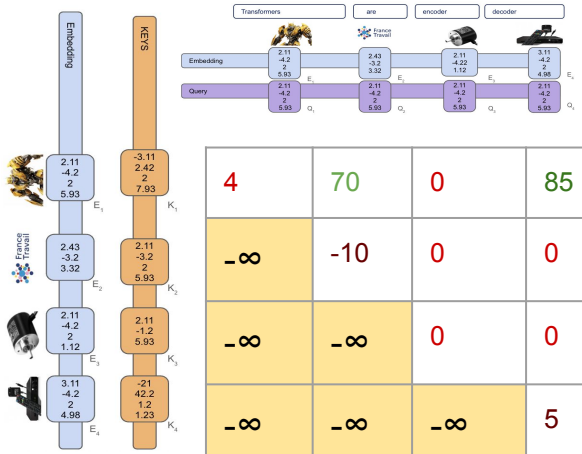
1x V_1	0.97 x V_1	0.33 x V_2	0.85 x V_1
0 x V_2	0.03 x V_2	0.33 x V_2	0.02 x V_2
0 x V_3	0 x V_3	0.33 x V_3	0.02x V_3
0 x V_4	0 x V_4	0 x V_4	0.11 x V_4



Values: What do I reveal to others ?
$|E|$: Embedding (1, 1512)
$|W_V|$: Value matrix (12 288, 12 288)

Attention pattern

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



4	70	0	85
$-\infty$	-10	0	0
$-\infty$	$-\infty$	0	0
$-\infty$	$-\infty$	$-\infty$	5

This step allows numerical stability

4/128	70/128	0	85/128
$-\infty$	-10	0	0
$-\infty$	$-\infty$	0	0
$-\infty$	$-\infty$	$-\infty$	5/128

1	0.97	0.33	0.85
0	0.03	0.33	0.02
0	0	0.33	0.02
0	0	0	0.11

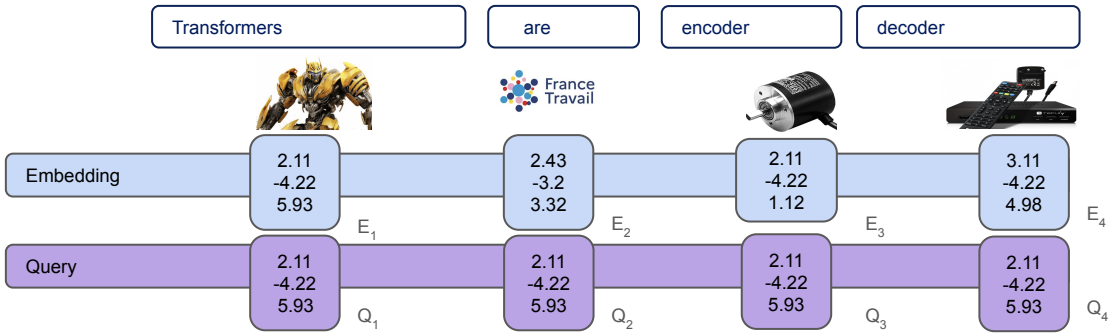
$$Q \times K^T \rightarrow \frac{Q \times K^T}{\sqrt{d_k}} \rightarrow \text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right)$$

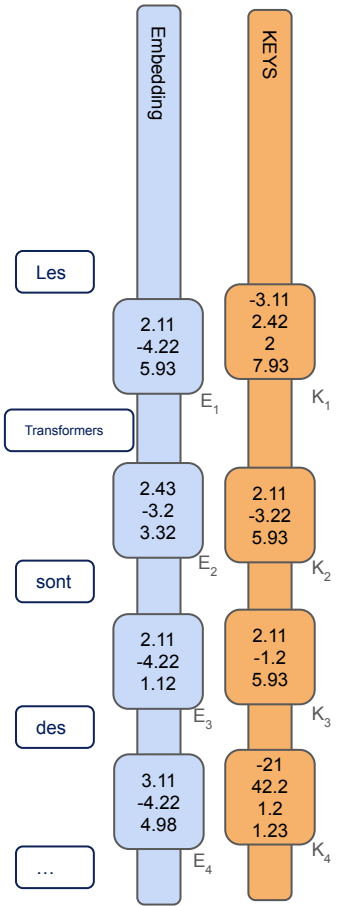
	Q_1	Q_2	Q_3	Q_4
K_1	$Q_1.K_1$	$Q_2.K_1$	$Q_3.K_1$	$Q_4.K_1$
K_2	$Q_1.K_2$	$Q_2.K_2$	$Q_3.K_2$	$Q_4.K_2$
K_3	$Q_1.K_3$	$Q_2.K_3$	$Q_3.K_3$	$Q_4.K_3$
K_4	$Q_1.K_4$	$Q_2.K_4$	$Q_3.K_4$	$Q_4.K_4$

	Q_1	Q_2	Q_3	Q_4
K_1	$Q_1.K_1/\sqrt{d_k}$	$Q_2.K_1/\sqrt{d_k}$	$Q_3.K_1/\sqrt{d_k}$	$Q_4.K_1/\sqrt{d_k}$
K_2	$Q_1.K_2/\sqrt{d_k}$	$Q_2.K_2/\sqrt{d_k}$	$Q_3.K_2/\sqrt{d_k}$	$Q_4.K_2/\sqrt{d_k}$
K_3	$Q_1.K_3/\sqrt{d_k}$	$Q_2.K_3/\sqrt{d_k}$	$Q_3.K_3/\sqrt{d_k}$	$Q_4.K_3/\sqrt{d_k}$
K_4	$Q_1.K_4/\sqrt{d_k}$	$Q_2.K_4/\sqrt{d_k}$	$Q_3.K_4/\sqrt{d_k}$	$Q_4.K_4/\sqrt{d_k}$

Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$





70	70	0	85
-4	35	0	0
2	0	30	0
3	-3	4	18
…	…	…	…

French to english translation example

No masking

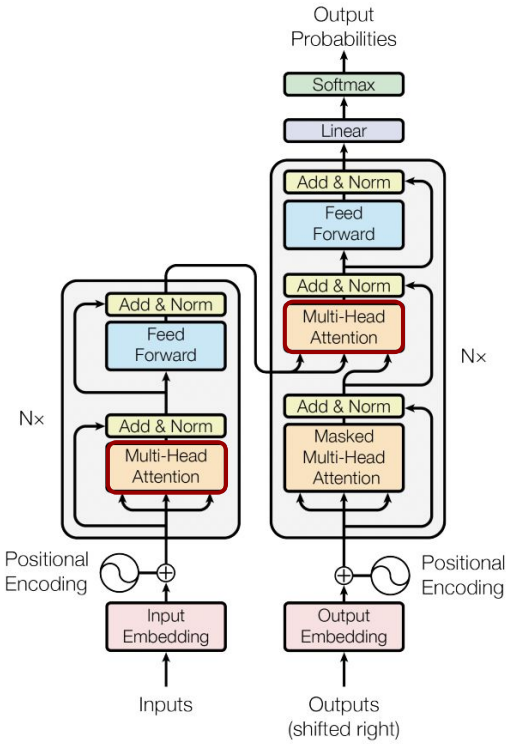


Figure 1: The Transformer - model architecture.

GPT 3 dimension for one attention head

Query: What am I looking for ?
|E| : Embedding (1, 12 288)
$|W_Q|$: Query matrix (12 288, 128)

Key: What do I have ?
|E| : Embedding (1, 12 288)
$|W_K|$: Query matrix (12 288, 128)

Values: What do I reveal to others ?
|E| : Embedding (1, 1512)
$|W_V|$: Value matrix (12 288, 12 288)

Embedding	(Embedding Dimension, N words)	(12 288, 50 257)	49 152 params
Key	(Key size, Embedding Dimension)	(128, 12 288)	1 572 864 params per head
Query	(Query size, Embedding Dimension)	(128, 12 288)	1 572 864 params per head
Value up	(Value size, Embedding Dimension)	(128, 12 288)	1 572 864 params per head
Value down	(Embedding Dimension, Value size)	(12 288, 128)	1 572 864 params per head

GPT 3 dimension for all attention heads: *603 979 776 parameters*

Query: What am I looking for ?
|E| : Embedding (1, 12 288)
$|W_Q|$: Query matrix (12 288, 128)

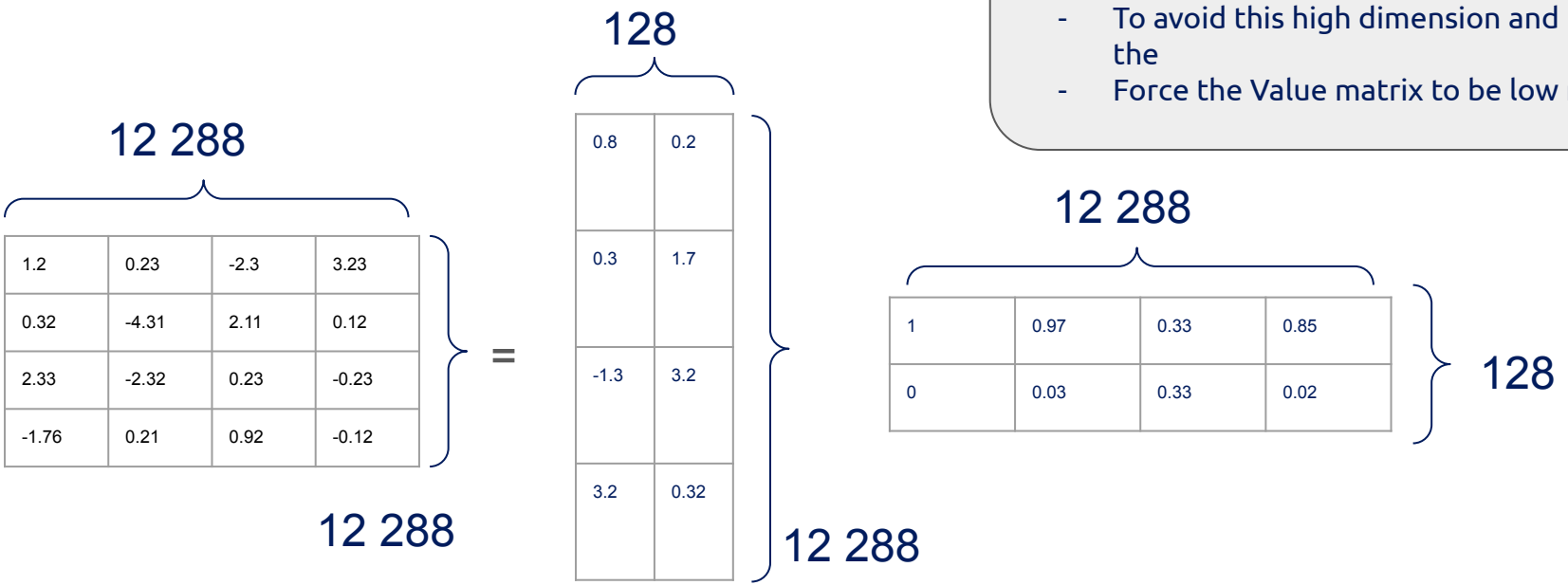
Key: What do I have ?
|E| : Embedding (1, 12 288)
$|W_K|$: Query matrix (12 288, 128)

Values: What do I reveal to others ?
|E| : Embedding (1, 1512)
$|W_V|$: Value matrix (12 288, 12 288)

Embedding	(Embedding Dimension, N words)	(12 288, 4)	49 152 params
Key	(Key size, Embedding Dimension)	(128, 12 288) x 96 heads	150 994 944 params
Query	(Query size, Embedding Dimension)	(128, 12 288) x 96 heads	150 994 944 params
Value up	(Value size, Embedding Dimension)	(128, 12 288) x 96 heads	150 994 944 params
Value down	(Embedding Dimension, Value size)	(12 288, 128) x 96 heads	150 994 944 params

Value Matrix (12 288, 12 288) decomposition≈

Idea:
- The number of # is 150m for the Value matrix
- To avoid this high dimension and respects the
- Force the Value matrix to be low rank



Value Matrix (12 288, 12 288) decomposition

Idea:

- The number of # is 150m for the Value matrix
- To avoid this high dimension and respects the
- Force the Value matrix to be low rank

12 288

1.2	0.23	-2.3	3.23
0.32	-4.31	2.11	0.12
2.33	-2.32	0.23	-0.23
-1.76	0.21	0.92	-0.12

12 288

=

128

0.8	0.2
0.3	1.7
-1.3	3.2
3.2	0.32

12 288

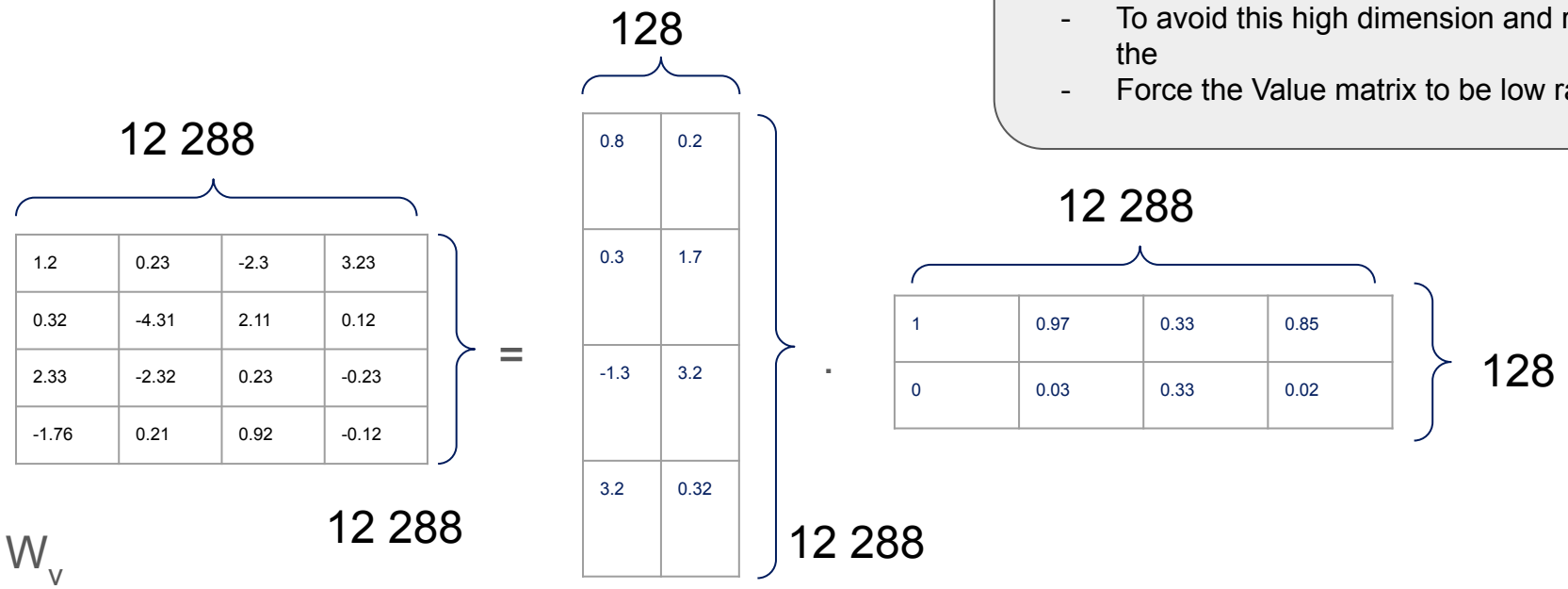
12 288

1	0.97	0.33	0.85
0	0.03	0.33	0.02

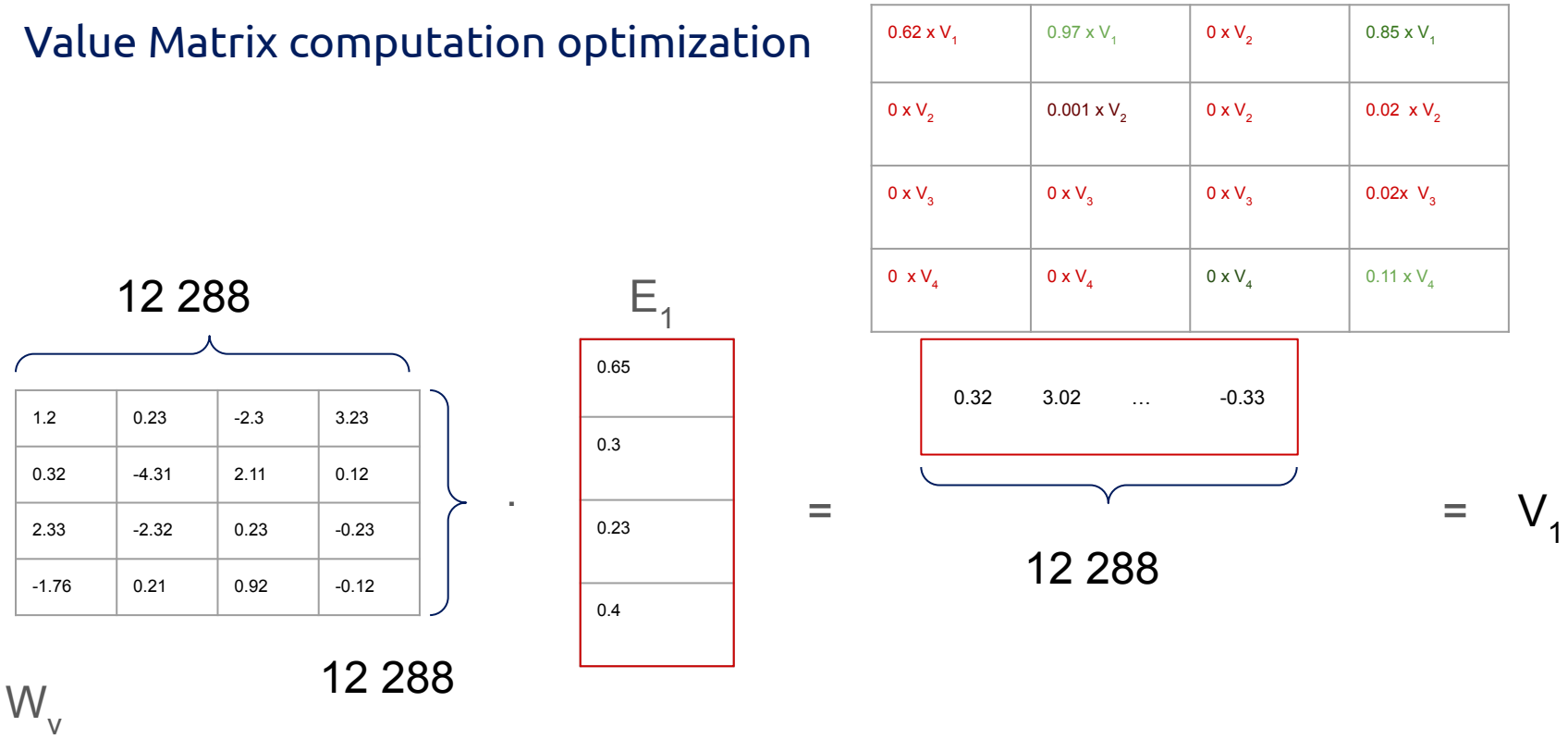
128

Value Matrix (12 288, 12 288) decomposition

Idea:
- The number of # is 150m for the Value matrix
- To avoid this high dimension and respects the
- Force the Value matrix to be low rank



Value Matrix computation optimization



Method 1 W_V (12 288, 12 288)

Number of computations:
- N words x 12 288 multiplications
- N words x 12 288 additions

0.62 x V_1	0.97 x V_1	0 x V_2	0.85 x V_1
0.1 x V_2	0.001 x V_2	0 x V_2	0.02 x V_2
0 x V_3	0 x V_3	0 x V_3	0.02x V_3
0 x V_4	0 x V_4	0 x V_4	0.11 x V_4

Method 1 W_V (12 288, 128)

Step 2:

Number of computations:
- N words x 128 multiplications
- N words x 128 additions

Matrix multiplication between value up matrix and result:
- 128 multiplication + 128 addition for each row
- 12 288 times

0.62 x V_1	0.97 x V_1	0 x V_2	0.85 x V_1
0.1 x V_2	0.001 x V_2	0 x V_2	0.02 x V_2
0 x V_3	0 x V_3	0 x V_3	0.02x V_3
0 x V_4	0 x V_4	0 x V_4	0.11 x V_4

As presented so far, the value vectors inside an attention head would have the same dimension as the embedding space. This would involve taking weighted sums of these large vectors, weighted by the attention pattern.

You can save on computation by instead running the weighted sums on the smaller intermediate outputs produced by the value-down matrices.

That is, inside each head, use only the value-down matrix to produce a sequence of 128-dimensional vectors. You take weighted sums of these with the attention pattern, meaning the head outputs a single 128-dimensional vector for each position in the context.

You could then multiply each of those by the head's value-up matrix to get a 12,288-dimensional vector that can be added to the embedding in that position.

However! All those up-projections are usually bundled together (more confusion!). For each position in the context, you concatenate all the 128-dimension vectors produced by each head at that position, creating a big vector with length (96*128). Now multiply this by the output matrix described on the previous slide.

This is mathematically equivalent to applying all the up-projections and adding the results (exercise). Given the chance, ML people *love* to compress things into a single matrix multiplication, even if it comes at the cost of conceptual clarity.

II.B. Transformers Architecture

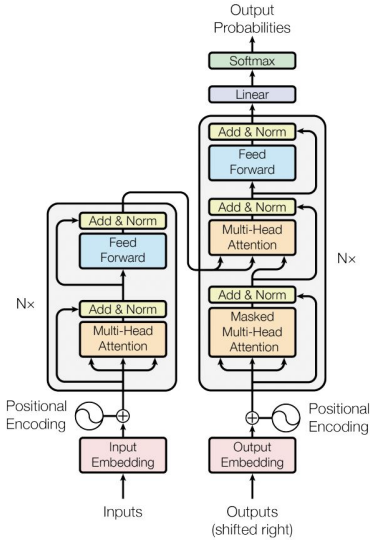


Figure 1: The Transformer - model architecture.

Introduction

1. Self Attention / Cross Attention
2. **Multi-Head Attention**
3. Residual connection & Layer normalization
4. Feed forward layer
5. Softmax Layer
6. Positional Embeddings

Dauphine | PSL TUNIS

II.B.2 Multi-Head Attention

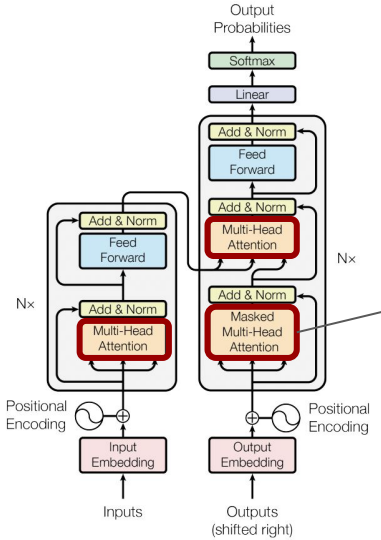
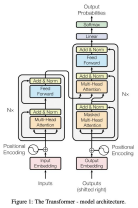


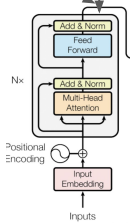
Figure 1: The Transformer - model architecture.

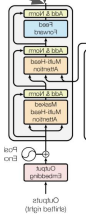
Transformers | are | encoder | decoder

France Travail

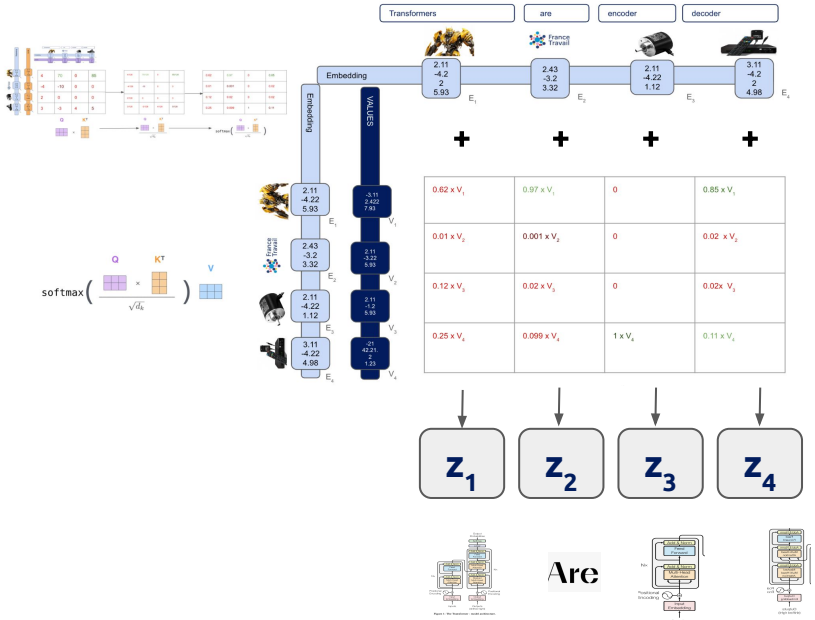
Attention mechanism

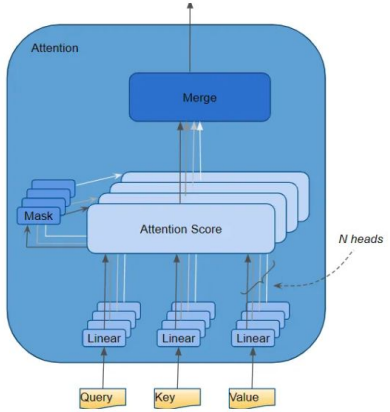


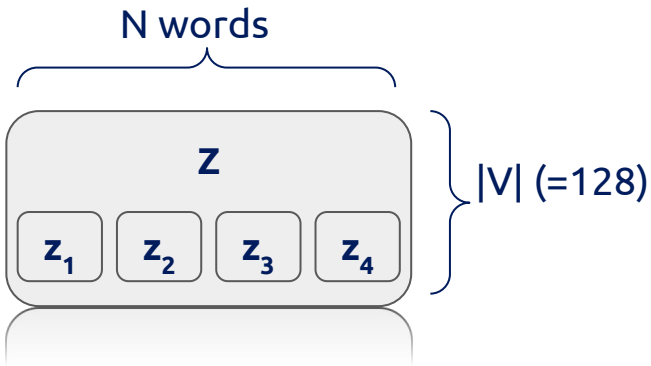


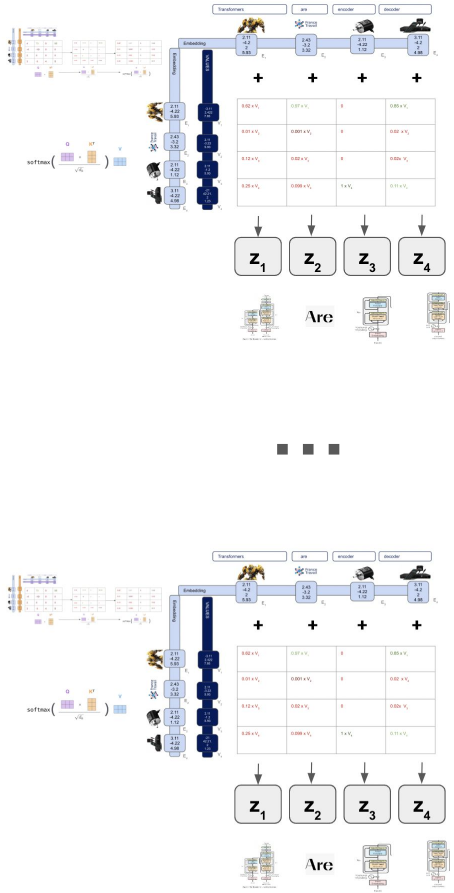


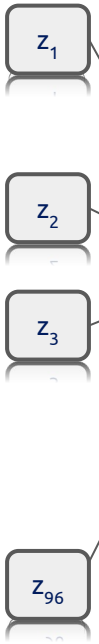
Dauphine | PSL TUNIS

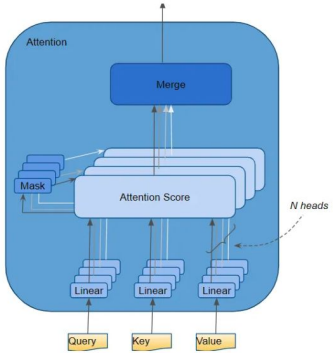






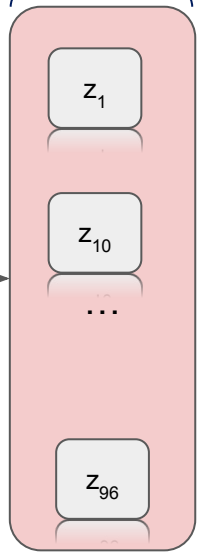




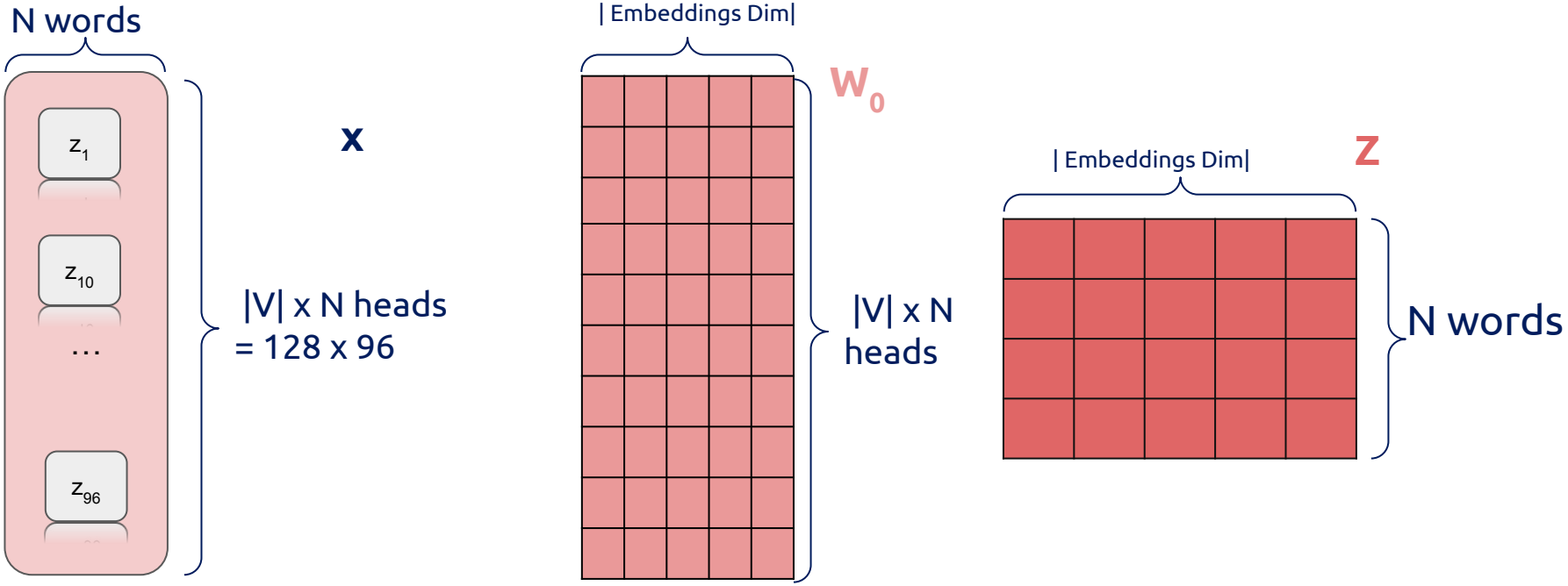


Concatenation

N words

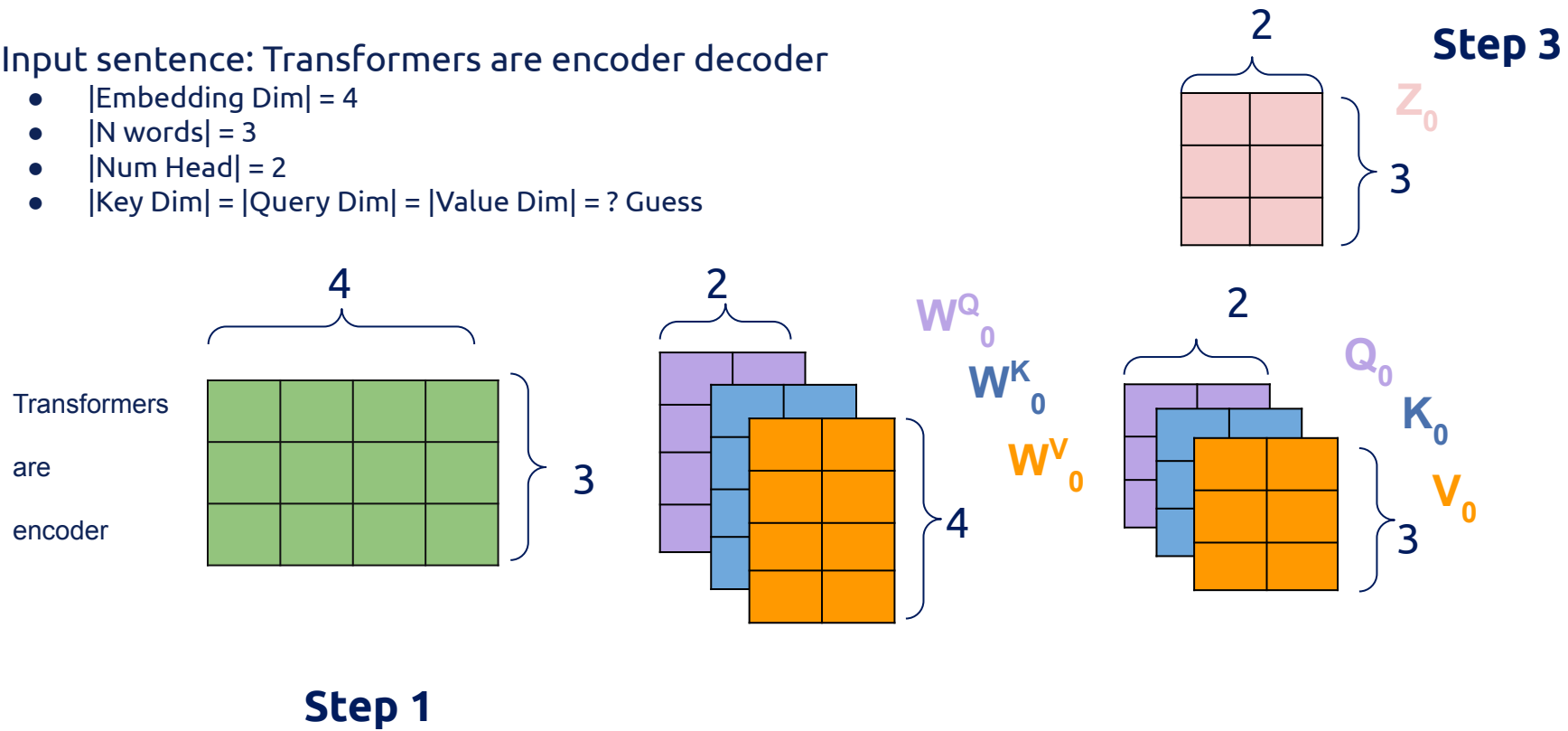


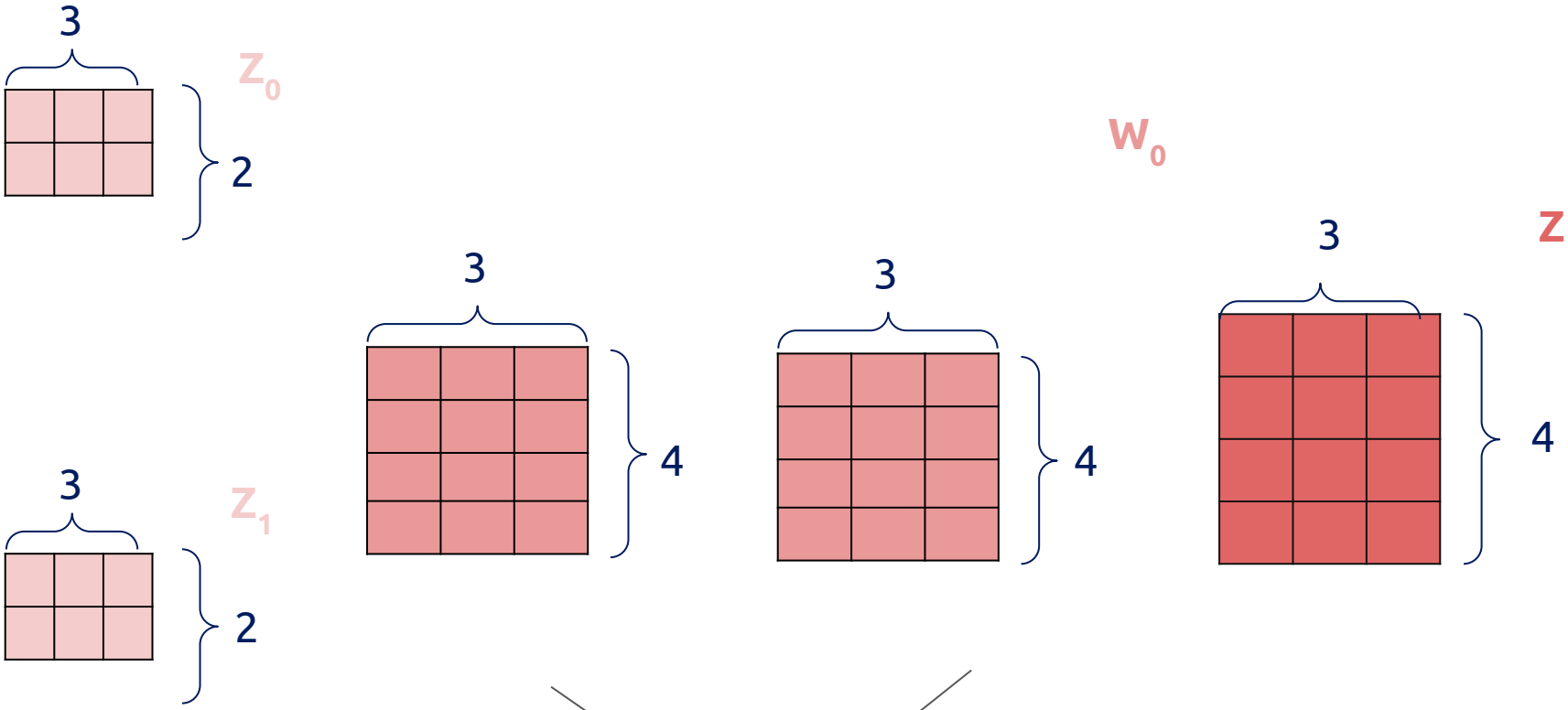
|V| x N heads
= 128 x 96
= 12 288
= embedding dimension



Input sentence: Transformers are encoder decoder

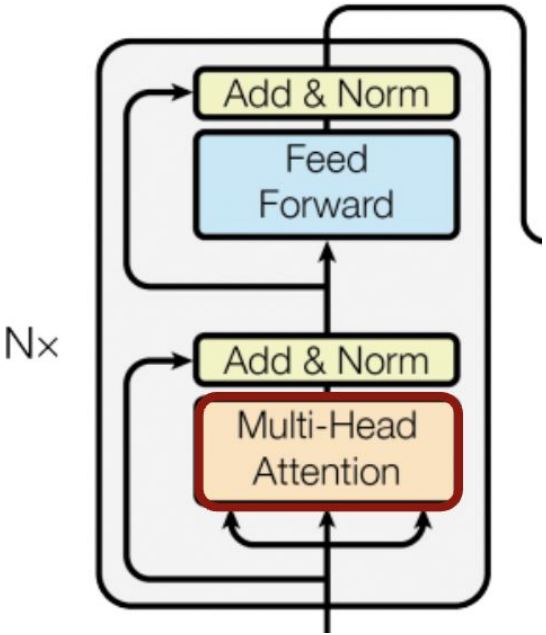
- |Embedding Dim| = 4
- |N words| = 3
- |Num Head| = 2
- |Key Dim| = |Query Dim| = |Value Dim| = ? Guess

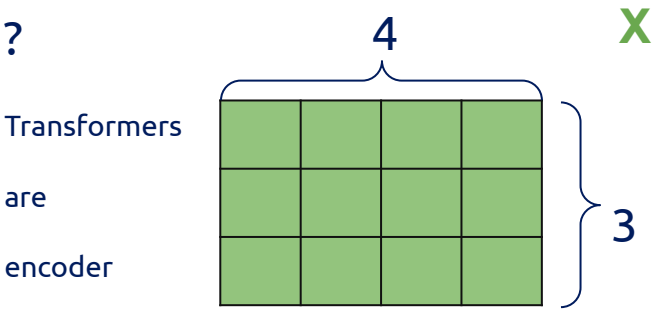


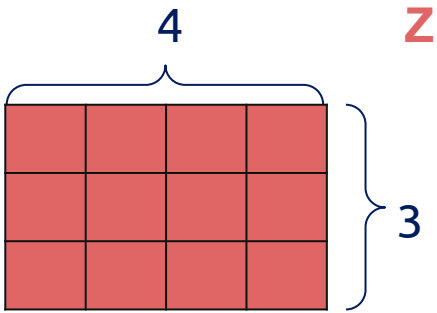


Dauphine | PSL TUNIS

Why do Z and X have the same dimension ?







II.B. Transformers Architecture

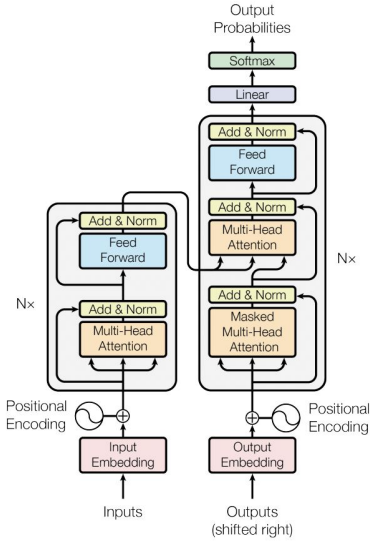


Figure 1: The Transformer - model architecture.

Introduction

1. Self Attention / Cross Attention
2. Multi-Head Attention
3. Residual connection & Layer normalization
4. Feed forward layer
5. Softmax Layer
6. Positional Embeddings

Dauphine | PSL TUNIS

II.B.3. Residual connections & Layer normalization

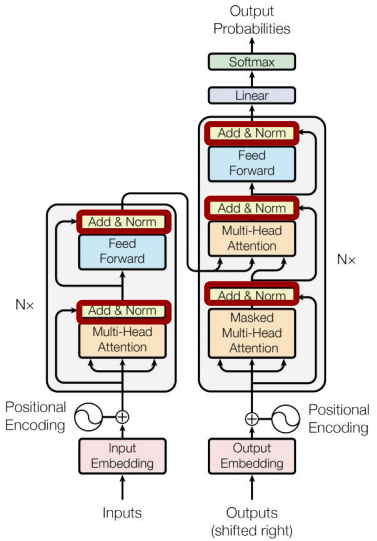


Figure 1: The Transformer - model architecture.

Dauphine | PSL TUNIS

Residual connections

- Residual connections mainly help mitigate the vanishing gradient problem
- Another effect of residual connections is that the information stays local in the Transformer layer stack

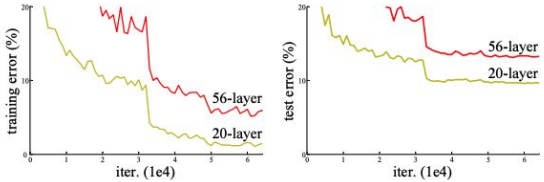
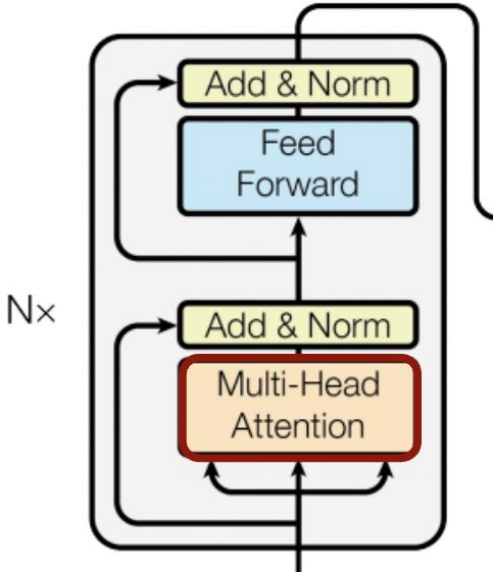
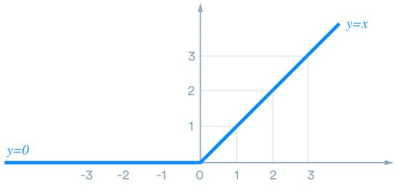


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

He & Al, 2015, Deep Residual Learning for Image Recognition

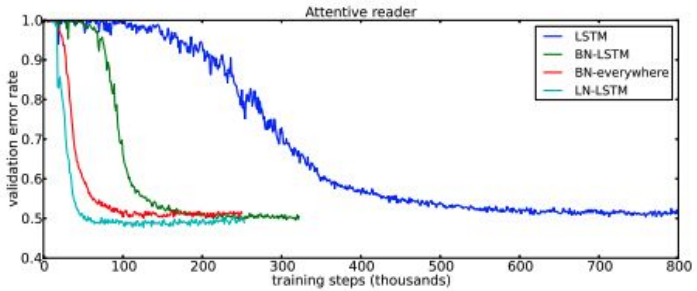


ReLU
y = max(x, 0)

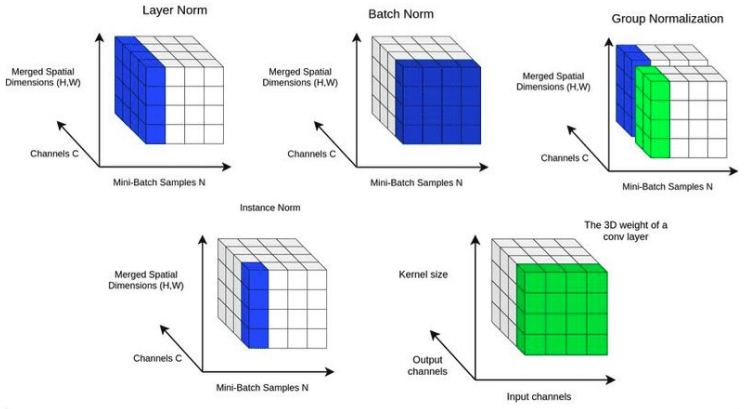


Layer normalization

- “Layer normalization is very effective at stabilizing the hidden state dynamics in recurrent networks.
- Empirically, we show that layer normalization can substantially reduce the training time compared with previously published techniques.”



Hinton & Al, 2016, Layer Normalization



2020, In-layer normalization techniques for training very deep neural networks [Blog]

So far ...

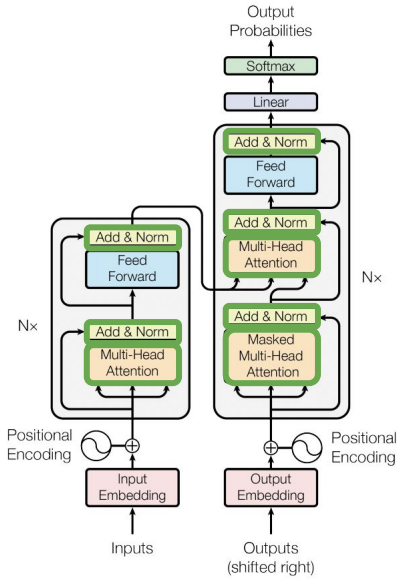


Figure 1: The Transformer - model architecture.

Dauphine | PSL TUNIS

II.B. Transformers Architecture

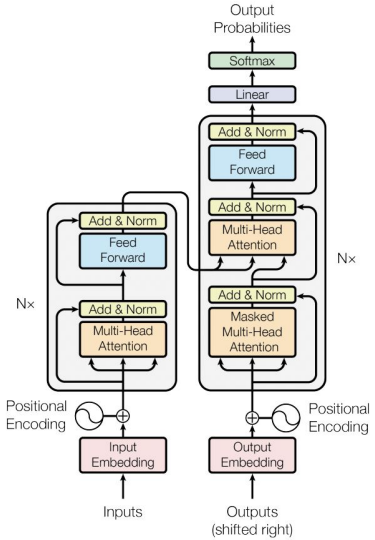


Figure 1: The Transformer - model architecture.

Introduction

1. **Self Attention / Cross Attention**
2. **Multi-Head Attention**
3. **Residual connection & Layer normalization**
4. **Feed forward layer**
5. **Softmax Layer**
6. **Positional Embeddings**
7. **Optimization**

Dauphine | PSL TUNIS

II.B.4. Feed forward layer

"In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between."

$$\mathrm{FFN}(x) = \underbrace{\max(0, xW_1 + b_1)}_{\text{'Up projection'}}\underbrace{W_2 + b_2}_{\text{'Down projection'}}$$

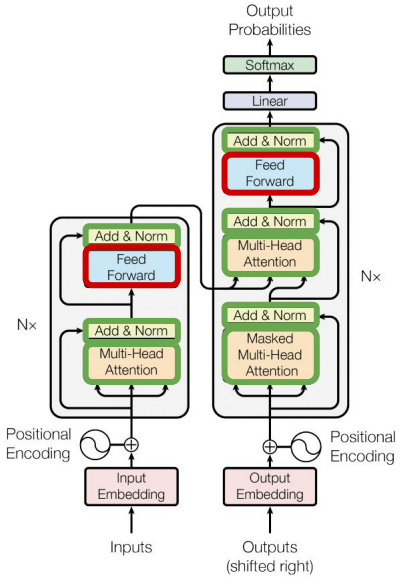
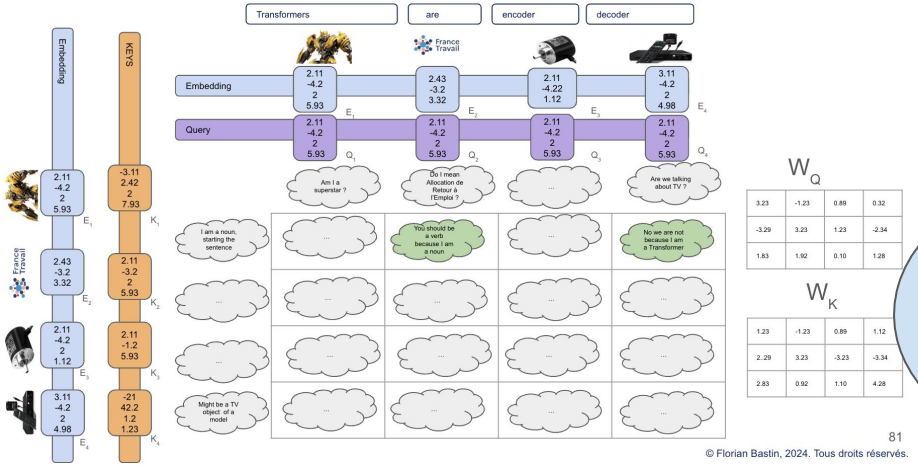


Figure 1: The Transformer - model architecture.

Dauphine | PSL TUNIS

Feed forward layer



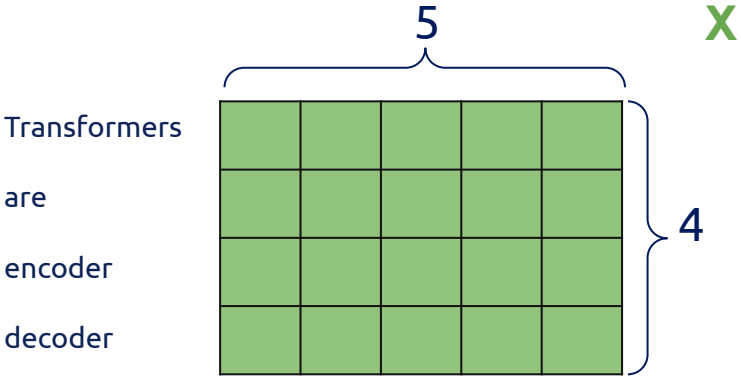
W_Q

3.23	-1.23	0.89	0.32
-3.29	3.23	1.23	-2.34
1.83	1.92	0.10	1.28

W_K

1.23	-1.23	0.89	1.12
2.29	3.23	-3.23	-3.34
2.83	0.92	1.10	4.28

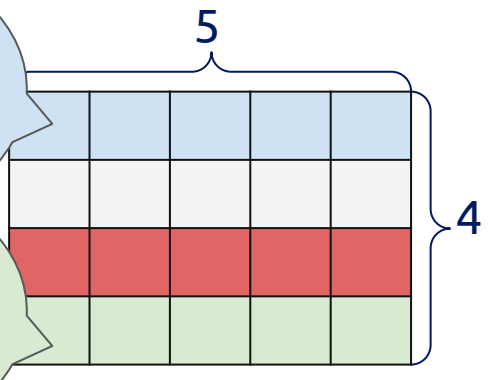
X



Transformers: "I am related to maths stuff, a plural noun, at the beginning of the sentence"

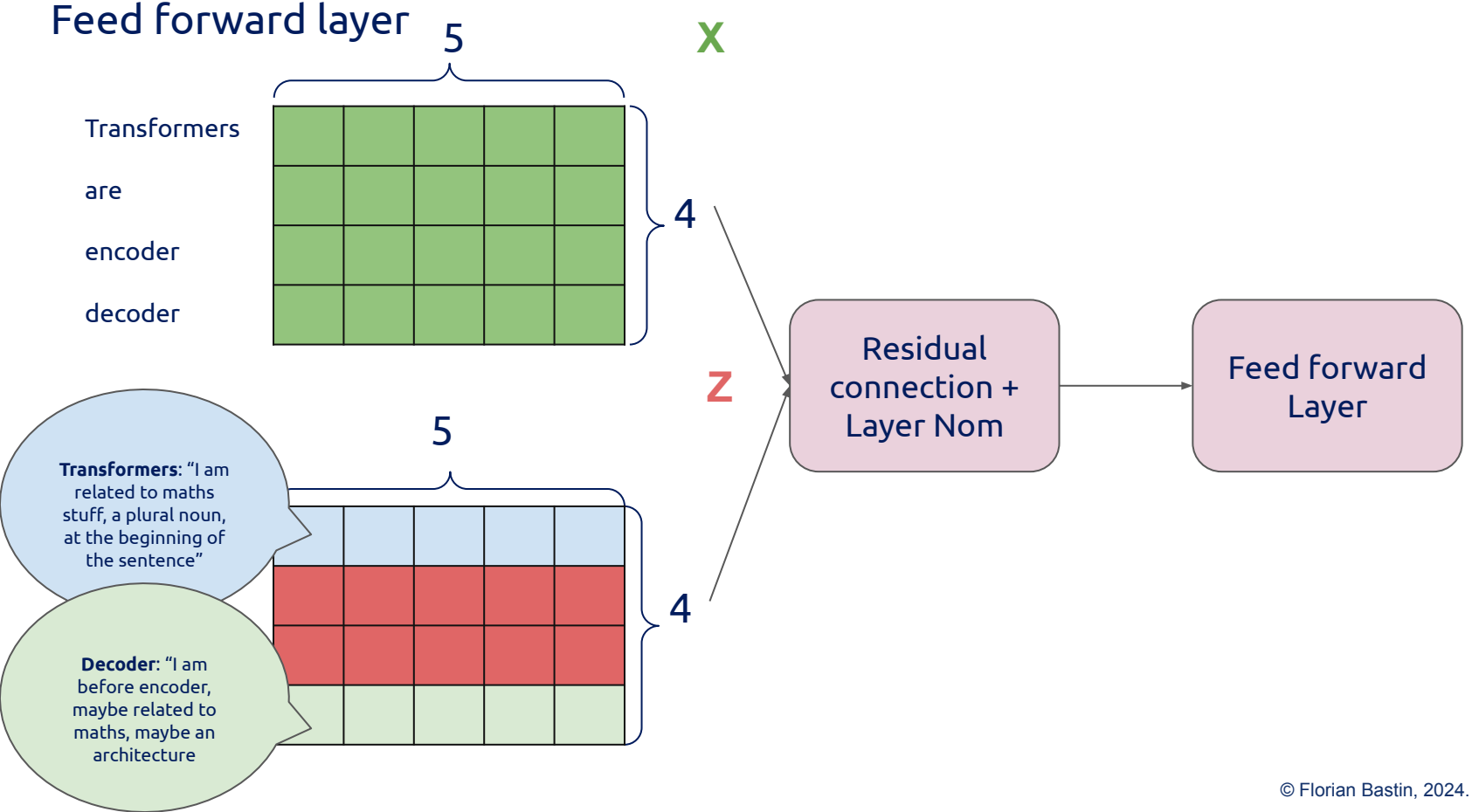
Decoder: "I am before encoder, maybe related to maths, maybe an architecture

Z

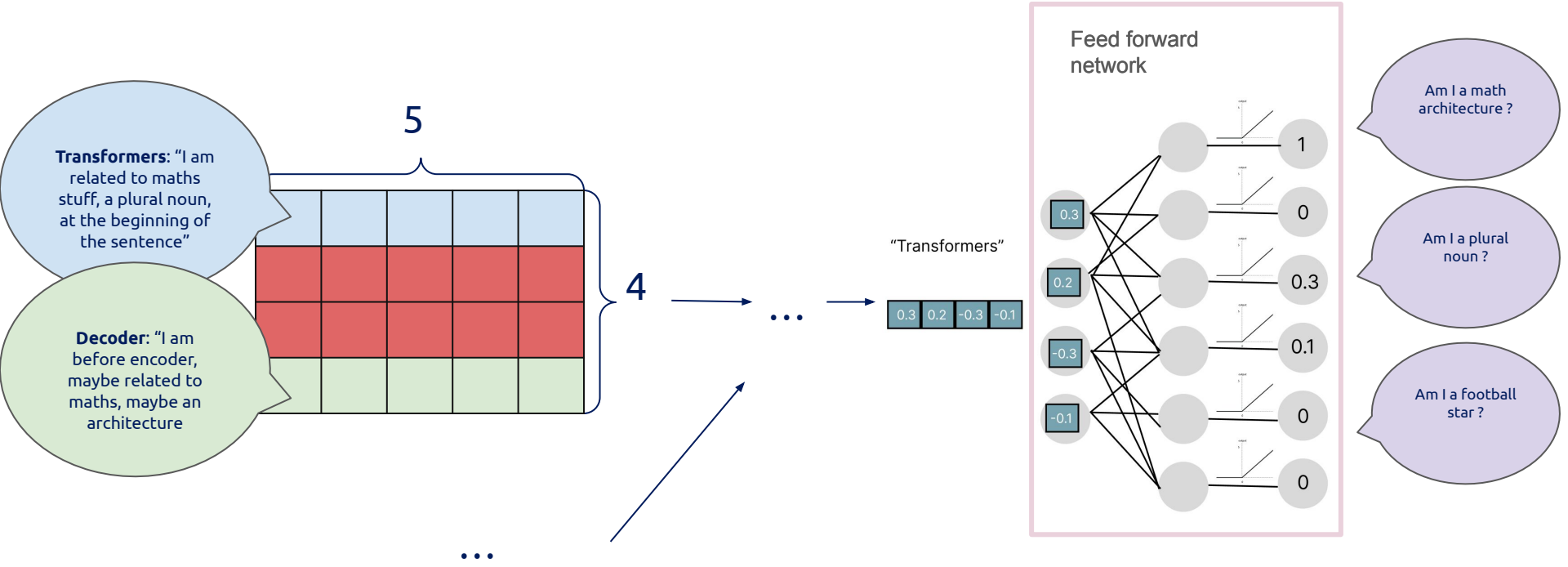


Dauphine | PSL TUNIS

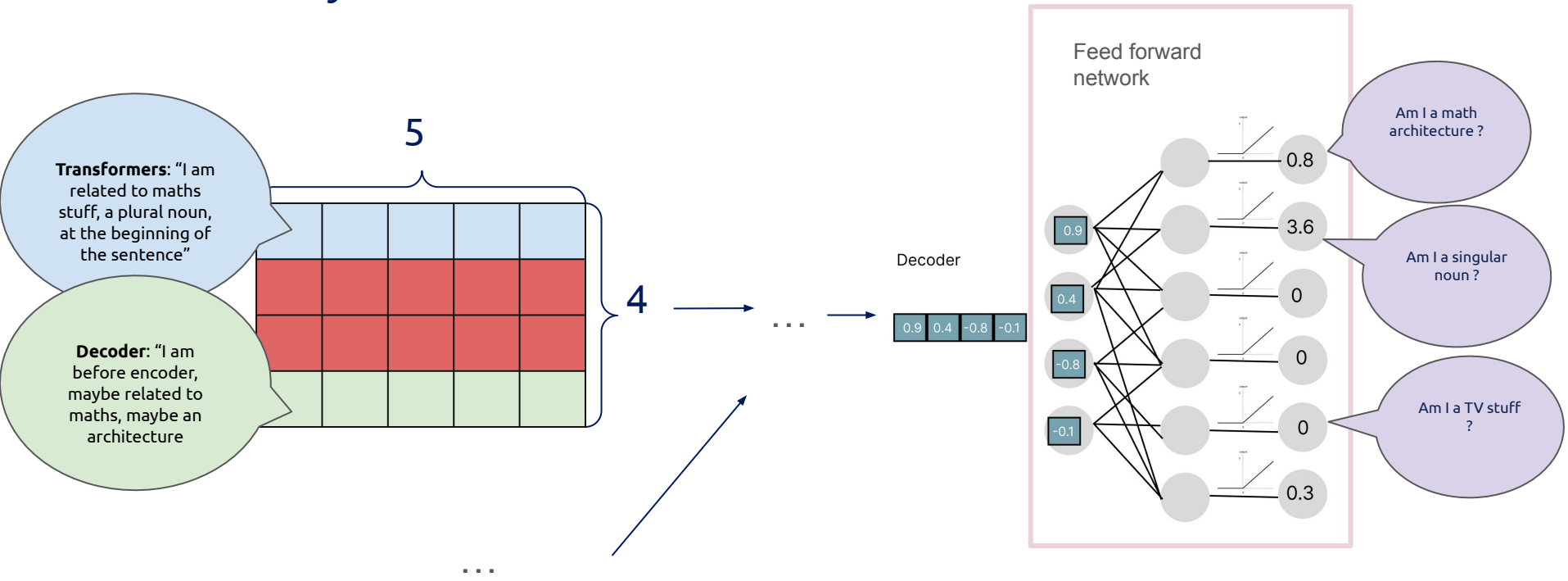
Feed forward layer



Feed forward layer

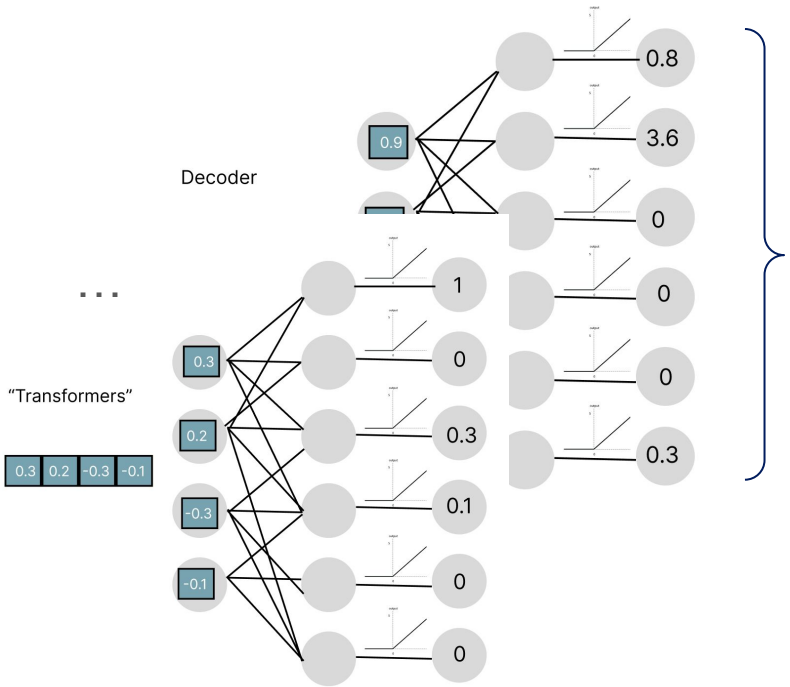


Feed forward layer



Feed forward layer : “Up projection”

Feed forward network

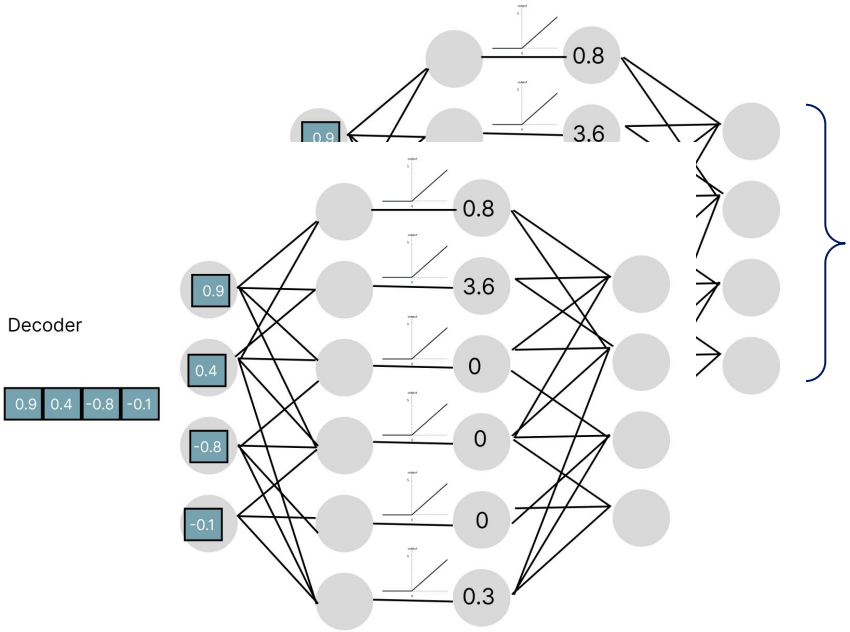


Up projection dimension = 49 152 x embedding dimension. In the paper, this value is called d_{ff}

49 152

Feed forward layer : “Down projection”

Another layer



Down projection dimension = embedding dim x 49 152

12 288

GPT 3 dimension **per layer**

Embedding	(Embedding Dimension, N words)	(12 288, 50 257)	617 558 016 params
Key	(Key size, Embedding Dimension)	(128, 12 288) x 96 heads	150 994 944 params
Query	(Query size, Embedding Dimension)	(128, 12 288) x 96 heads	150 994 944 params
Value up	(Value size, Embedding Dimension)	(128, 12 288) x 96 heads	150 994 944 params
Value down	(Embedding Dimension, Value size)	(12 288, 128) x 96 heads	150 994 944 params
Up Projection	(Neuron Dim, Embedding Dimension)	(49 152, 12 288)	603 979 776 params
Down Projection	(Embedding Dimension, Neuron Dim)	(12 288, 49 152)	603 979 776 params
Unembedding	(N words, Embedding Dimension)	(50 257, 12 288)	617 558 016

GPT 3 dimension (96 layers):

Embedding	(Embedding Dimension, N words)	(12 288, 50 257)	617 558 016 params
Key	(Key size, Embedding Dimension)	(128, 12 288) x 96 heads x 96 layers	14 495 514 624 params
Query	(Query size, Embedding Dimension)	(128, 12 288) x 96 heads x 96 layers	14 495 514 624 params
Value up	(Value size, Embedding Dimension)	(128, 12 288) x 96 heads x 96 layers	14 495 514 624 params
Value down	(Embedding Dimension, Value size)	(12 288, 128) x 96 heads x 96 layers	14 495 514 624 params
Up Projection	(Neuron Dim, Embedding Dimension)	(49 152, 12 288) x 96 layers	57 982 058 496 params
Down Projection	(Embedding Dimension, Neuron Dim)	(12 288, 49 152) x 96 layers	57 982 058 496 params
Unembedding	(N words, Embedding Dimension)	(50 257, 12 288)	617 558 016 params

So far ...

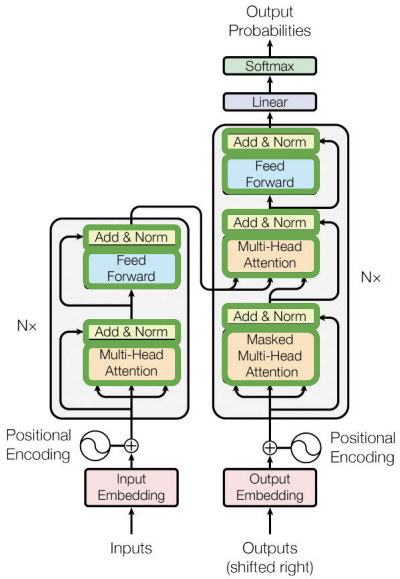


Figure 1: The Transformer - model architecture.

Dauphine | PSL TUNIS

II.B. Transformers Architecture

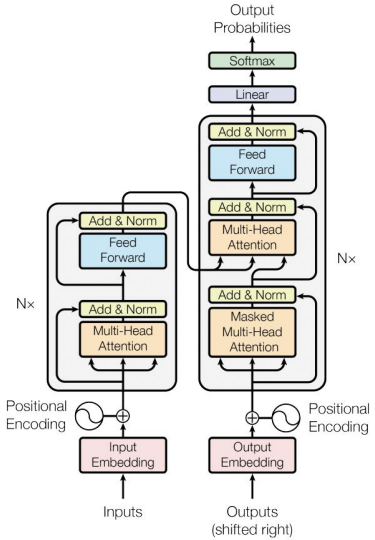


Figure 1: The Transformer - model architecture.

Introduction

1. Self Attention / Cross Attention
2. Multi-Head Attention
3. Residual connection & Layer normalization
4. Feed forward layer
5. Softmax Layer
6. Positional Embeddings

Dauphine | PSL TUNIS

II.B.5. Softmax Layer

"Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model} *.*

We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities.

In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by $\sqrt{d_{model}}$*."*

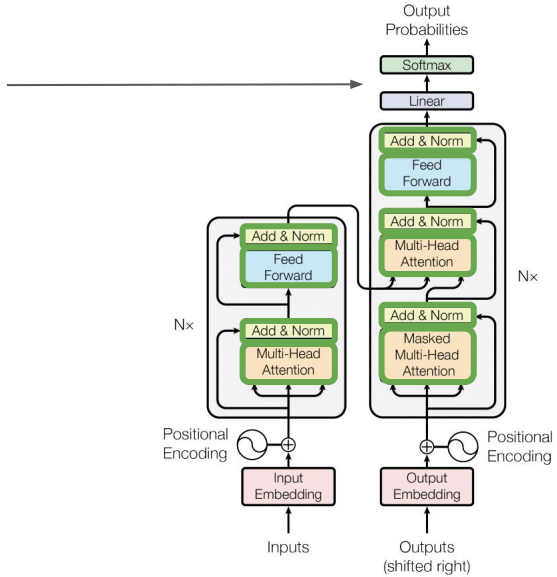
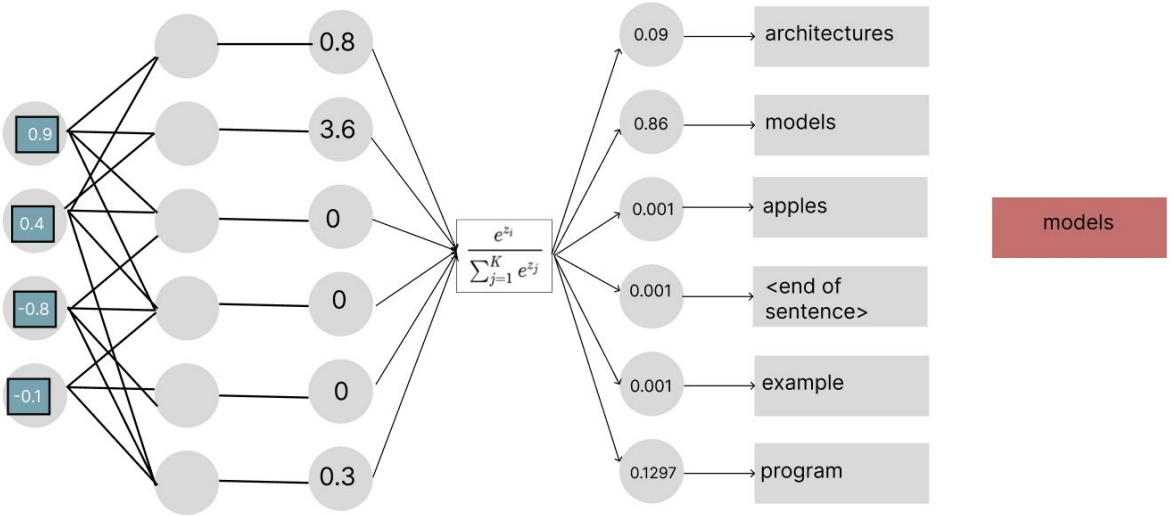


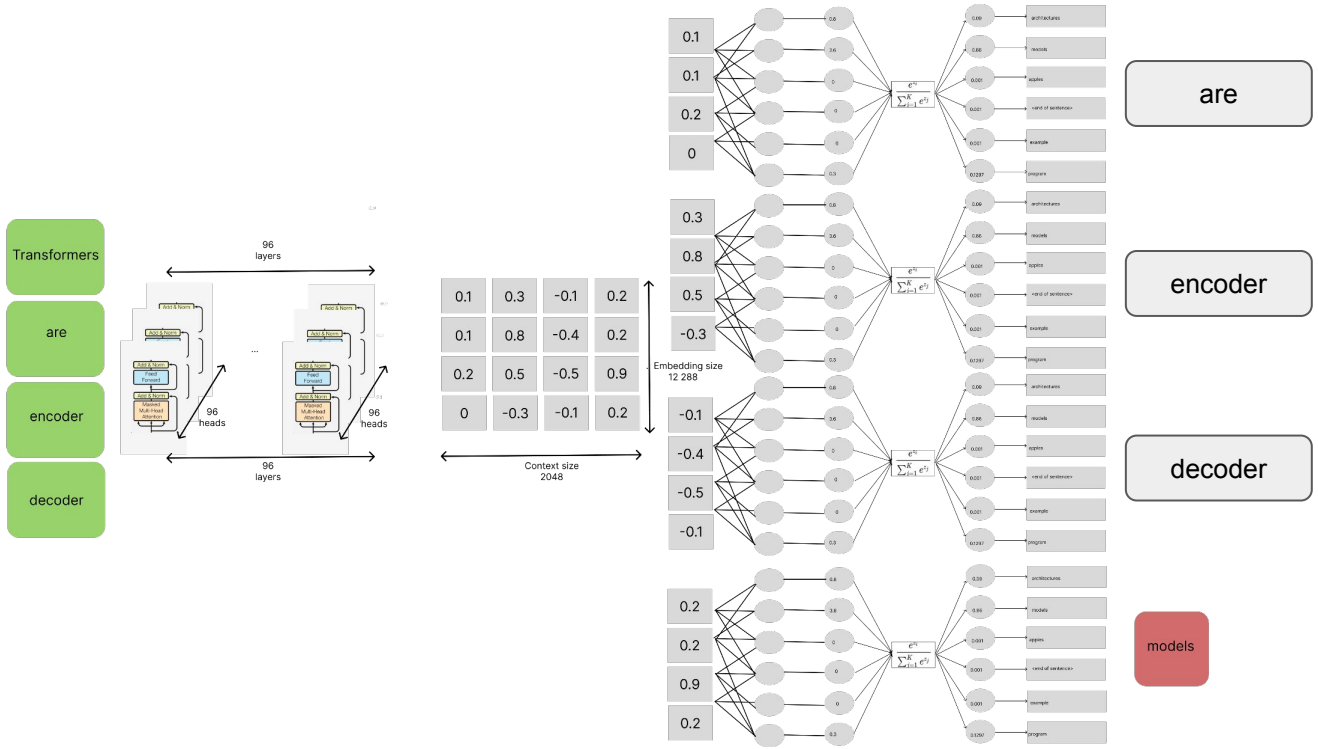
Figure 1: The Transformer - model architecture.

Dauphine | PSL TUNIS

- The input is he vector from the **last word** of the sentence
- The output is the probability distribution over all words in the dictionary (50k words for GPT 3-

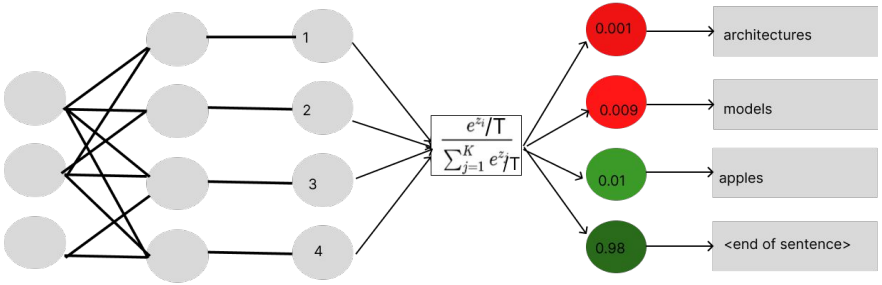


Q. Why don't we take all the previous representations of the other vectors for the inference ?

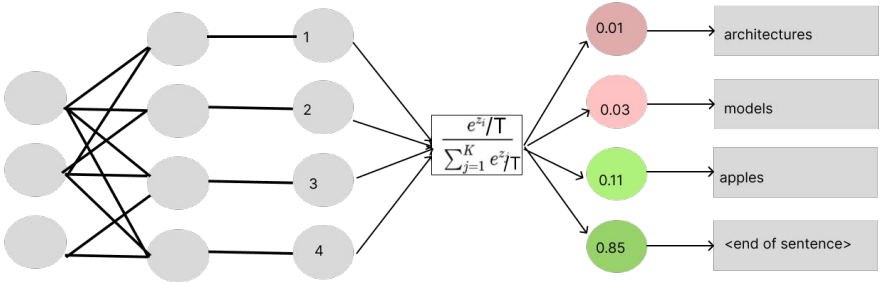


- For training, each word/token is used for next word prediction. The model is trained to predict next word from only its previous word.
- Of course, the last word context is learned with attention

Temperature
T = 0.1



Temperature
T = 0.5



Dauphine | PSL TUNIS

GPT 3 dimension (96 layers):
175 181 291 520 trainable parameters

Embedding	(Embedding Dimension, N words)	(12 288, 50 257)	617 558 016 params
Key	(Key size, Embedding Dimension)	(128, 12 288) x 96 heads x 96 layers	14 495 514 624 params
Query	(Query size, Embedding Dimension)	(128, 12 288) x 96 heads x 96 layers	14 495 514 624 params
Value up	(Value size, Embedding Dimension)	(128, 12 288) x 96 heads x 96 layers	14 495 514 624 params
Value down	(Embedding Dimension, Value size)	(12 288, 128) x 96 heads x 96 layers	14 495 514 624 params
Up Projection	(Neuron Dim, Embedding Dimension)	(49 152, 12 288) x 96 layers	57 982 058 496 params
Down Projection	(Embedding Dimension, Neuron Dim)	(12 288, 49 152) x 96 layers	57 982 058 496 params
Unembedding	(N words, Embedding Dimension)	(50 257, 12 288)	617 558 016 params

So far ...

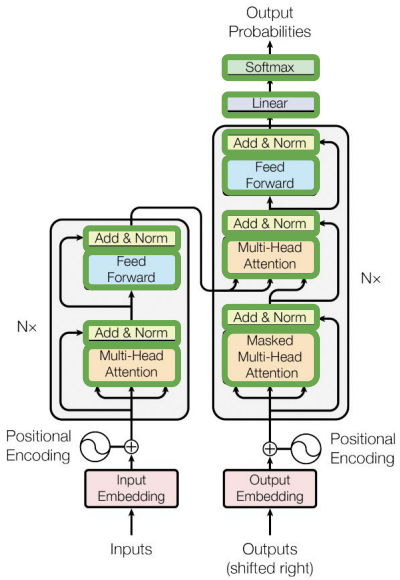


Figure 1: The Transformer - model architecture.

Dauphine | PSL TUNIS

II.B. Transformers Architecture

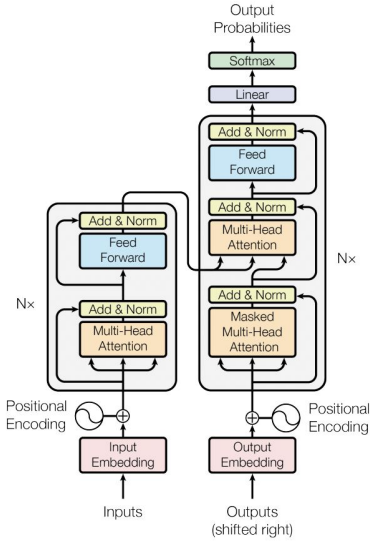


Figure 1: The Transformer - model architecture.

Introduction

1. Self Attention / Cross Attention
2. Multi-Head Attention
3. Residual connection & Layer normalization
4. Feed forward layer
5. Softmax Layer
6. **Positional Embeddings**

Dauphine | PSL TUNIS

II.B.6. Positional Embeddings

"Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence.
To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed."

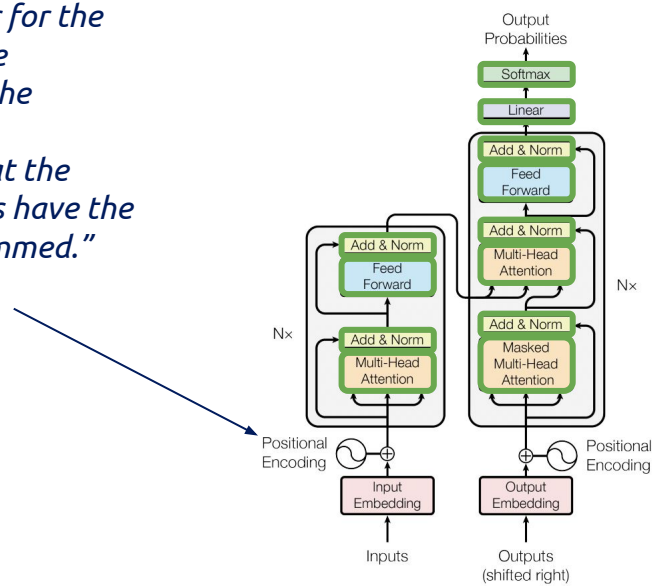


Figure 1: The Transformer - model architecture.

Dauphine | PSL TUNIS

- Computations are not done sequentially (unlike RNN and LSTM)
- How to compare “A B C” and “C A B” ?

Beneficial to find a method that satisfy the **following points:**

1. Unambiguous (each position have its own value)
2. Deterministic
3. Allows to estimate distance between tokens
4. Works with longer sequence than seen during training

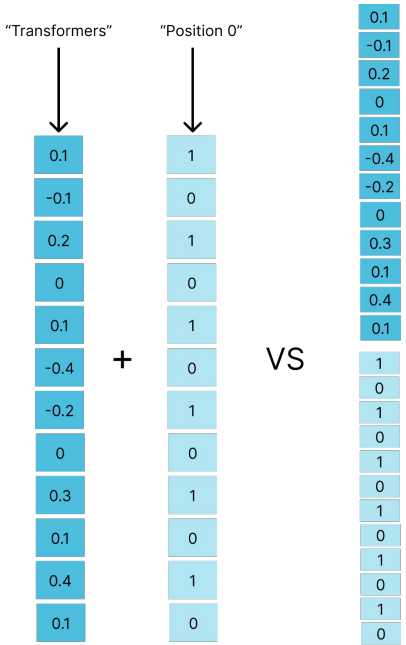
- Same size embedding to represent position (computationnaly for efficient that concatenation and model size increase)
- Don’t want to allow the model to extract information about positional informations only. Have to be coupled with word meaning

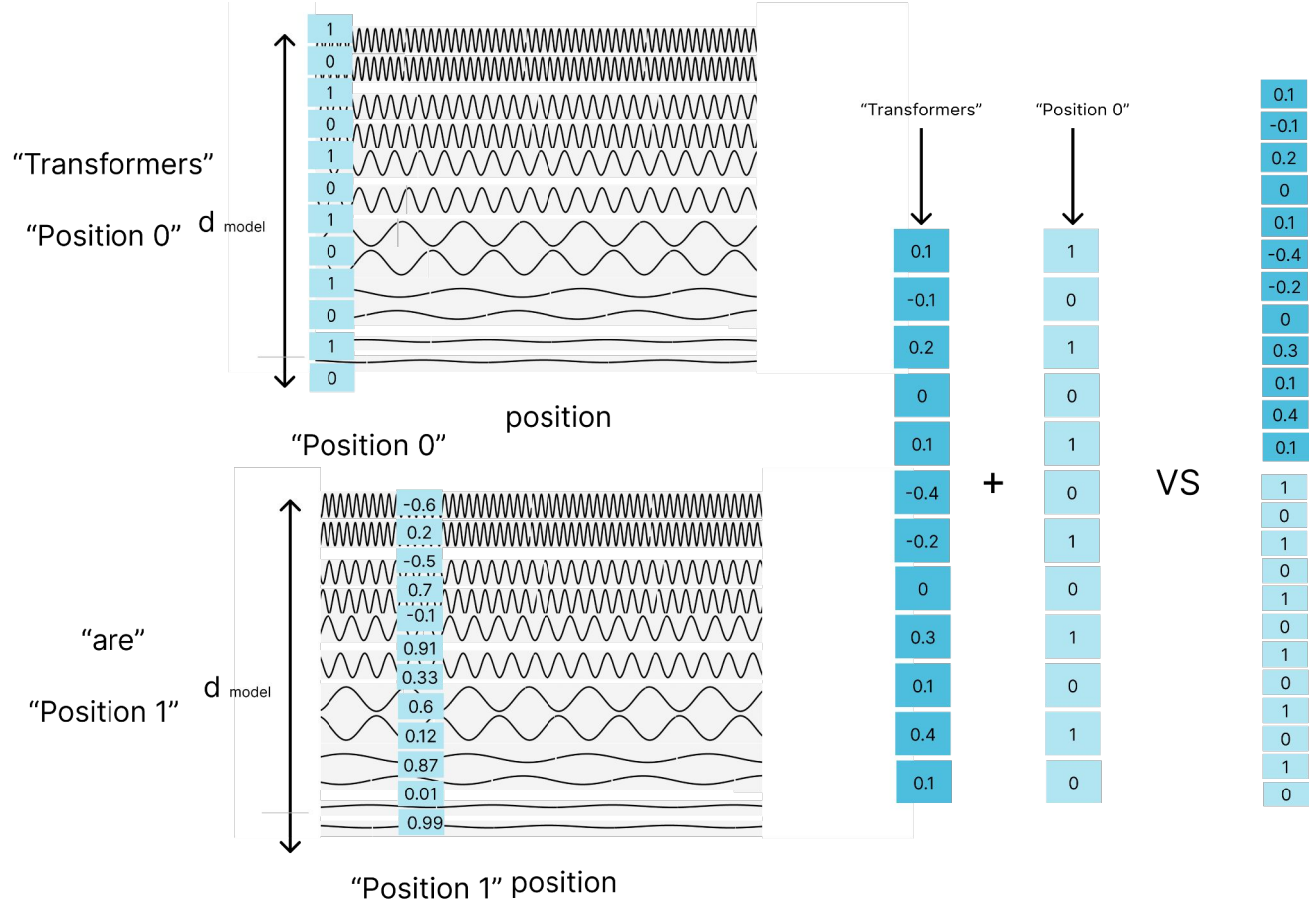
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Where:

- *pos* is the position {1, …, context length}
- *i* is the dimension = {1, …, d_{model}}
- d_{model} is the embedding dimension (*GPT 3 = 12 288*)





Properties

- The positional values are unique if at least one function has maximum size of this sequence (context window) 👌
- The positional values are not random, created using two equations 👌
- Looking at frequencies, we can estimate distance between positions. For positions near to each other, we can use high frequency functions. For long distance position, we can use function with larger periods. 👌
- Since *sin* and *cosine* are periodic functions, the model can generalize for longer sequences 👌

"We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}."

For every sine-cosine pair corresponding to frequency ω_k, there is a linear transformation $M \in \mathbb{R}^{2\times 2}$ (independent of t) where the following equation holds:

$$M.\begin{bmatrix}\sin(\omega_k.t)\\ \cos(\omega_k.t)\end{bmatrix} = \begin{bmatrix}\sin(\omega_k.(t+\phi))\\ \cos(\omega_k.(t+\phi))\end{bmatrix}$$

Proof:

Let M be a 2×2 matrix, we want to find u_1, v_1, u_2 and v_2 so that:

$$\begin{bmatrix}u_1 & v_1\\ u_2 & v_2\end{bmatrix}.\begin{bmatrix}\sin(\omega_k.t)\\ \cos(\omega_k.t)\end{bmatrix} = \begin{bmatrix}\sin(\omega_k.(t+\phi))\\ \cos(\omega_k.(t+\phi))\end{bmatrix}$$

By applying the addition theorem, we can expand the right hand side as follows:

$$\begin{bmatrix}u_1 & v_1\\ u_2 & v_2\end{bmatrix}.\begin{bmatrix}\sin(\omega_k.t)\\ \cos(\omega_k.t)\end{bmatrix} = \begin{bmatrix}\sin(\omega_k.t)\cos(\omega_k.\phi)+\cos(\omega_k.t)\sin(\omega_k.\phi)\\ \cos(\omega_k.t)\cos(\omega_k.\phi)-\sin(\omega_k.t)\sin(\omega_k.\phi)\end{bmatrix}$$

Which result in the following two equations:

$$u_1\sin(\omega_k.t)+v_1\cos(\omega_k.t) = \cos(\omega_k.\phi)\sin(\omega_k.t)+\sin(\omega_k.\phi)\cos(\omega_k.t) \quad (1)$$
$$u_2\sin(\omega_k.t)+v_2\cos(\omega_k.t) = -\sin(\omega_k.\phi)\sin(\omega_k.t)+\cos(\omega_k.\phi)\cos(\omega_k.t) \quad (2)$$

By solving above equations, we get:

$$u_1 = \cos(\omega_k.\phi) \quad v_1 = \sin(\omega_k.\phi)$$
$$u_2 = -\sin(\omega_k.\phi) \quad v_2 = \cos(\omega_k.\phi)$$

So the final transformation matrix M is:

$$M_{\phi,k} = \begin{bmatrix}\cos(\omega_k.\phi) & \sin(\omega_k.\phi)\\ -\sin(\omega_k.\phi) & \cos(\omega_k.\phi)\end{bmatrix}$$

Kazemnejads, 2019, Transformer Architecture: The Positional Encoding [Blog]

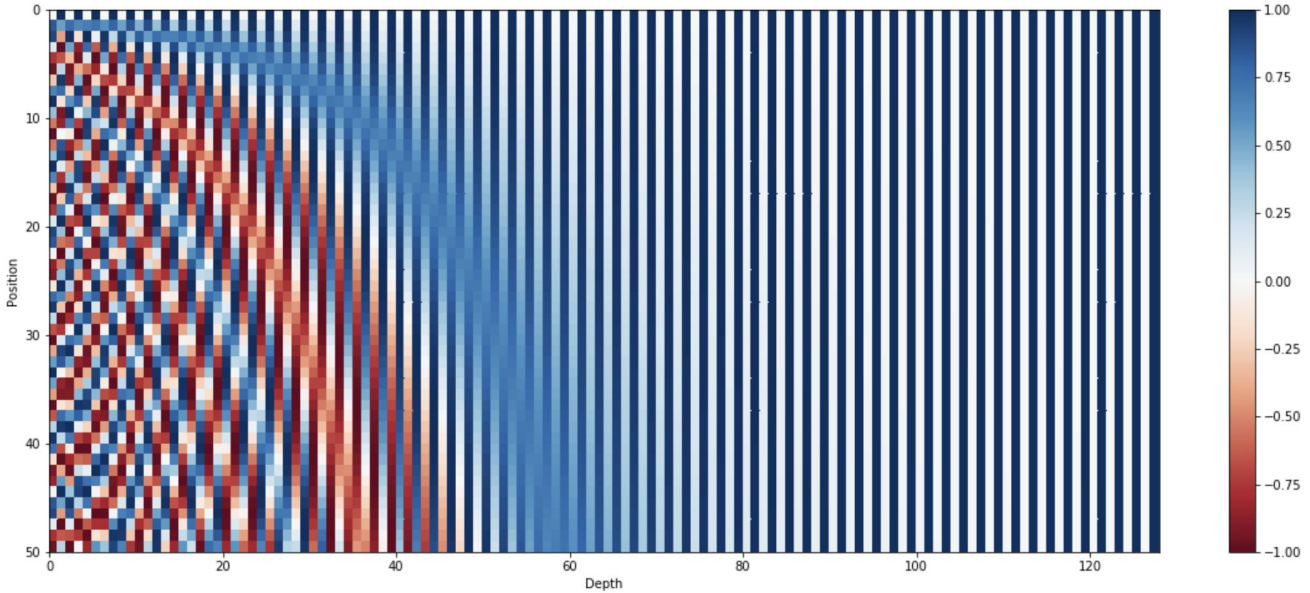


Figure: Positional encoding representation
Each row represent a positional vector for a given token

Kazemnejads, 2019, Transformer Architecture: The Positional Encoding [Blog]

So far ...

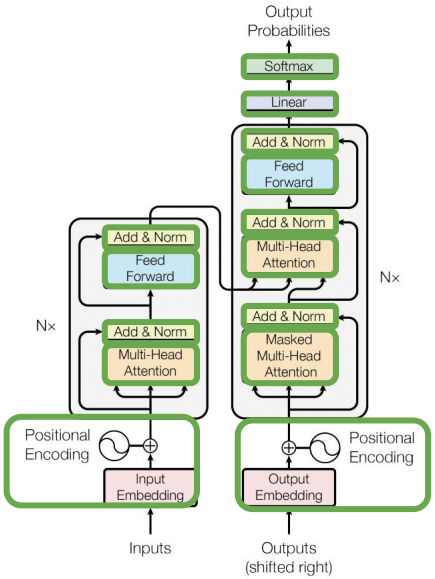


Figure 1: The Transformer - model architecture.

Dauphine | PSL TUNIS

Blogs:

- The Illustrated Transformer
- Transformers Explained Visually (Part 3): Multi-head Attention, deep dive

Papers:

- Attention is All You Need

Videos:

- Visual introduction to Transformers (part 1)
- Transformers visualized (part 2)
- How might LLMs store fact (part 3)
- Residual Network and skip connections
- Stanford CS25: V2 I Introduction to Transformers w/ Andrej Karpathy

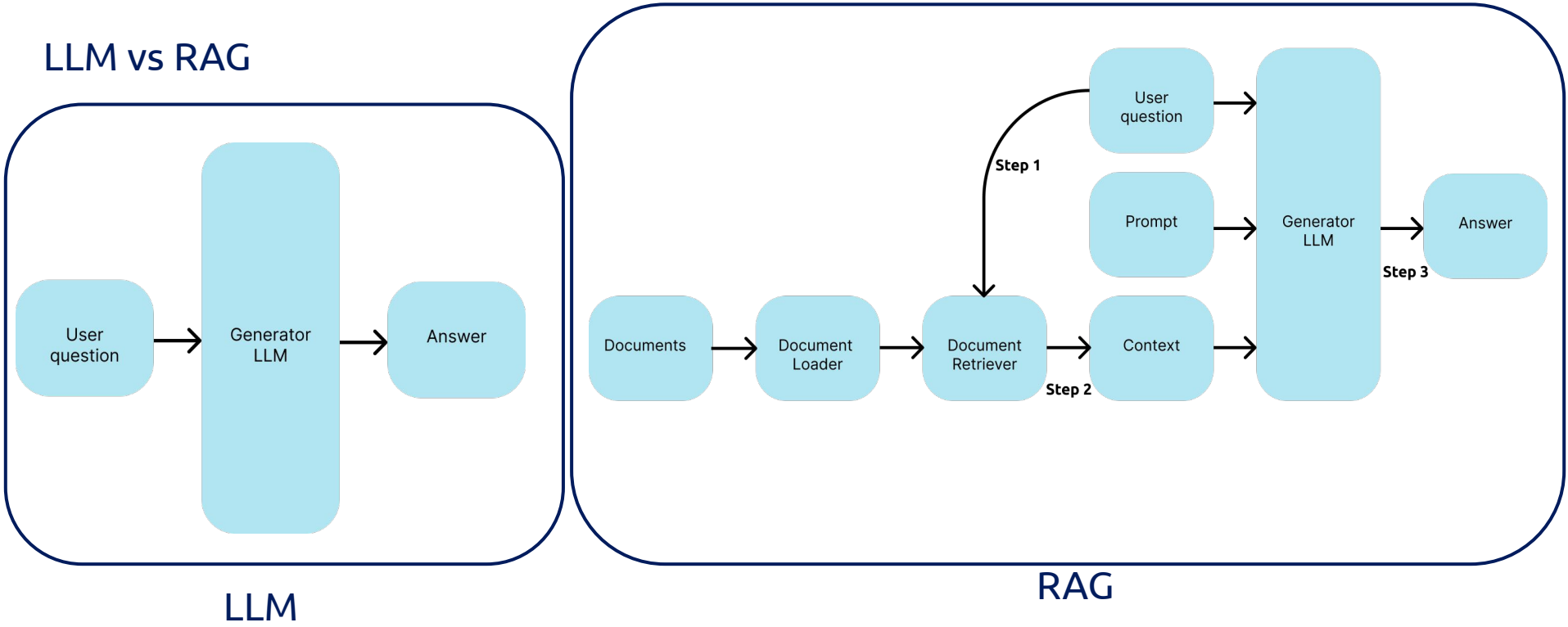
Go to : https://kahoot.com/
Click Play
Enter code: XXX XXX
Enter your name

III. Retrieval Augmented Generation

1. Basic Architecture
2. Information retrieval
3. Vectorstore & Search optimization
4. RAG Techniques
5. Evaluation
6. Multimodal RAG
7. SOTA RAG architectures

Dauphine | PSL
TUNIS

LLM vs RAG



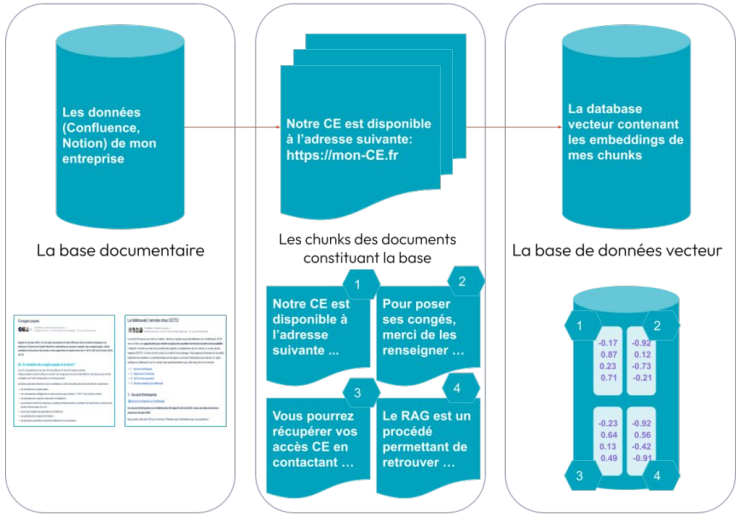
RAG Definition

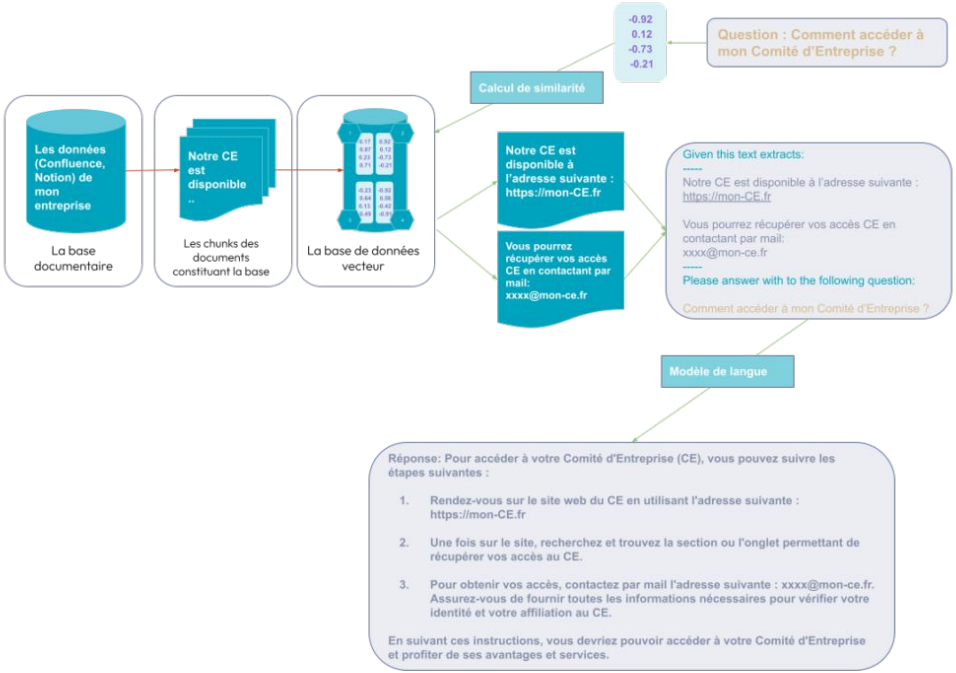
Definition: Retrieval-Augmented Generation (RAG) is a framework that combines retrieval-based and generation-based models. It enhances the capabilities of language models by providing them with access to external knowledge bases or documents during the generation process. This allows the model to generate more accurate and up-to-date information by retrieving relevant data instead of relying solely on its internal parameters.

Benefits:

- Produces more informed and factual responses.
- Can handle queries about recent events not present in the training data.
- Reduces hallucinations common in language models.

RAG Architecture



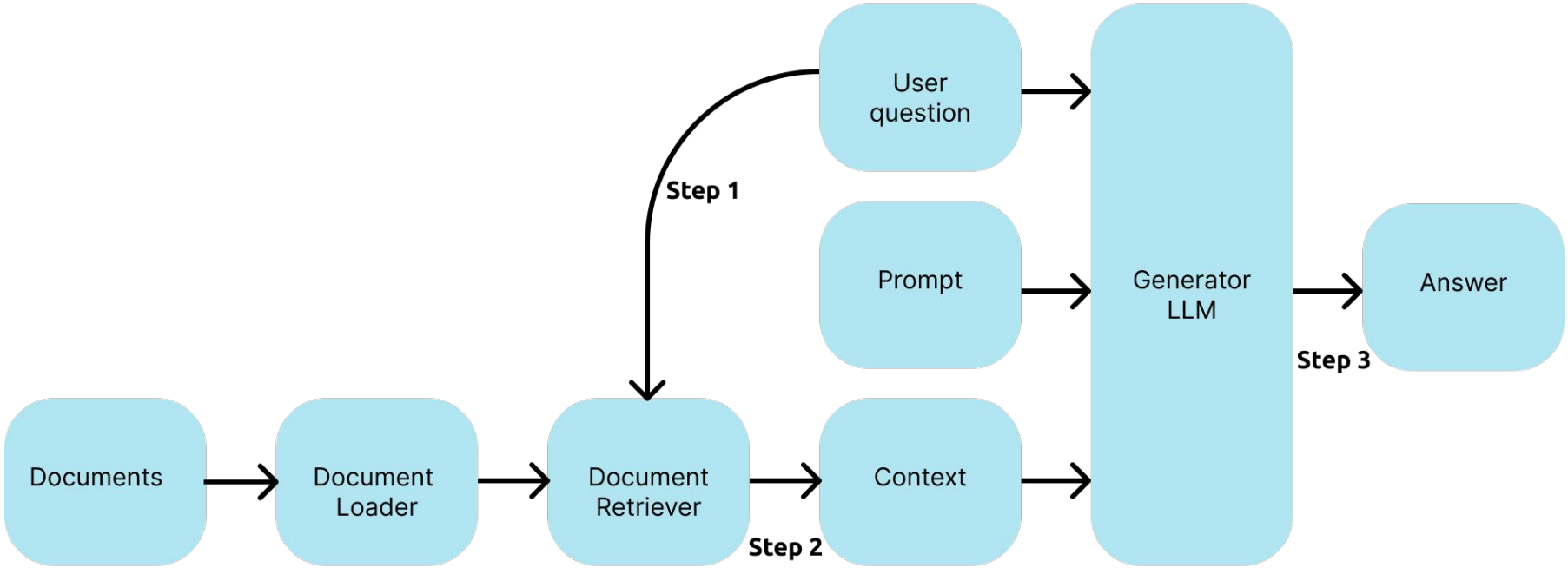


Step 1: Document ingestion

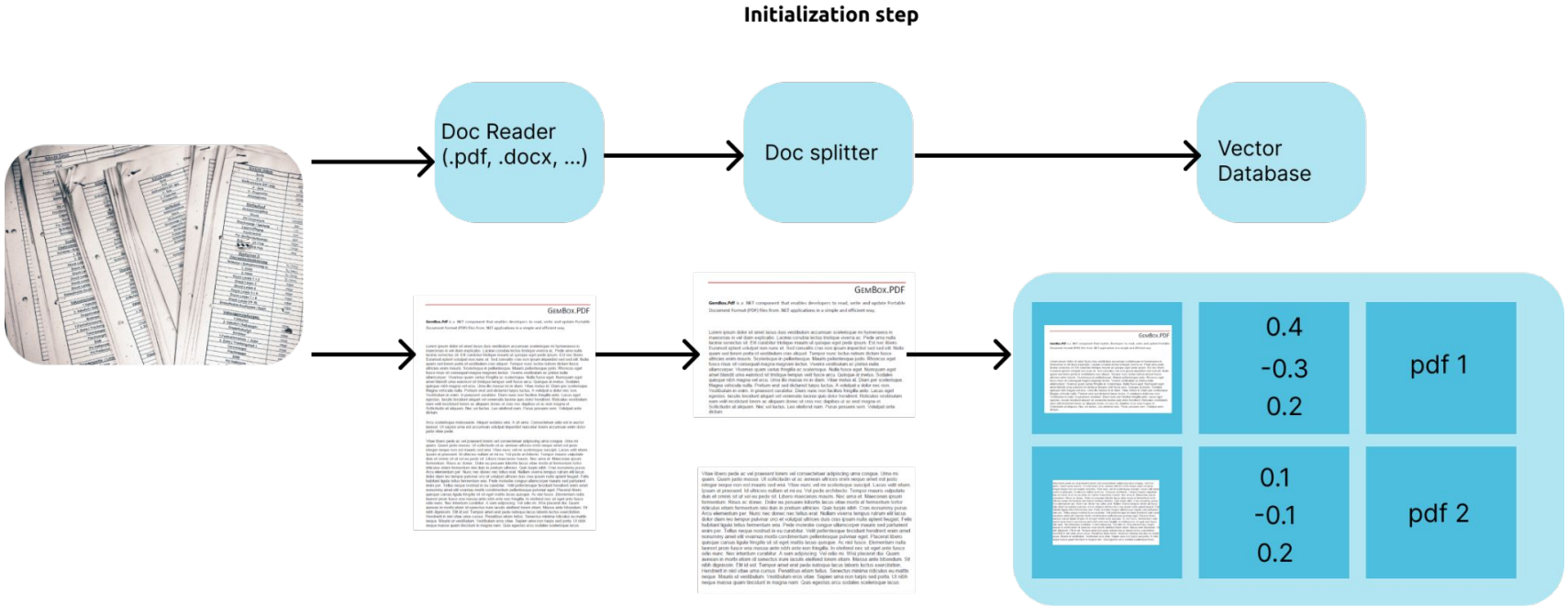
Step 2: Contextualized answering

[Construire son RAG (Retrieval Augmented Generation) grâce à langchain: L'exemple de l'Helpdesk d'OCTO]

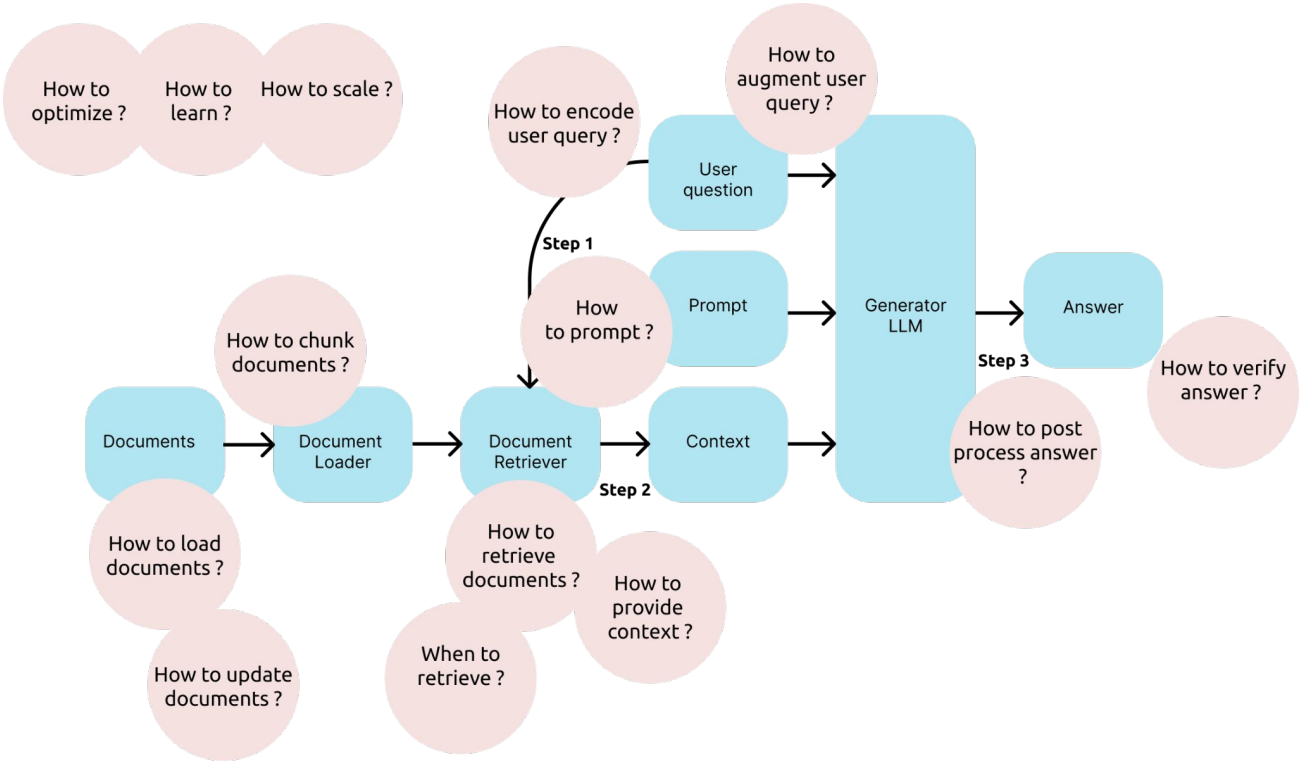
RAG Architecture



RAG Architecture



How to ?



How to retrieve documents ?

TF-IDF

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

$\mathrm{tf}(t,d)$

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

- TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents
- Most important word for each document is highlighted

TF-IDF score computation. [Image Source]

x

$\mathrm{idf}(t,D)$

blue	bright	can	see	shining	sky	sun	today
0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$

	blue	bright	can	see	shining	sky	sun	today
1	**0.301**	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	**0.201**
3	0	0.0417	0	0	0	**0.100**	0.0417	0
4	0	0.0209	**0.100**	**0.100**	**0.100**	0	0.0417	0

BM 25

Given a query Q, containing keywords $\{q_1, \ldots, q_n\}$, the BM25 score of a document D is:

$$\mathrm{score}(D,Q) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\mathrm{avgdl}}\right)}$$

$$\mathrm{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

- f(qi,D) is the number of times that the keyword qi occurs in the document *D*,
- |D| is the length of the document *D* in words
- avgdl is the average document length in the text collection from which documents are drawn.
- K_1 and *b* are free parameters, usually chosen, in absence of an advanced optimization, as $K_1 \in [1.2,2.0]$ and b=0.75
- *N* is the total number of documents in the collection, and
- n(qi) s the number of documents containing q_i

Text Search using TF-IDF and Elasticsearch

How to retrieve documents ?

Cosine Similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Euclidean distance

$$d(p, q) = \|p - q\|.$$

How to retrieve documents ?

Maximal Marginal Relevance

$$MMR \stackrel{def}{=} Arg \max_{D_i \in R \setminus S} \left[\lambda (Sim_1(D_i, Q) - (1-\lambda) \max_{D_j \in S} Sim_2(D_i, D_j)) \right]$$

The goal of this metric is to retrieve dissimilar documents and increase diversity

- D is the set of all candidate documents, R is the set of already selected documents, q is the query
- Sim_1 is the similarity function between a document and the query
- Sim_2 is the similarity function between two documents.
- di and dj are documents in D and R respectively

The parameter λ (mmr_threshold) controls the trade-off between relevance (the first term) and diversity (the second term). If mmr_threshold is close to 1, more emphasis is put on relevance, while a mmr_threshold close to 0 puts more emphasis on diversity.

The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries

How to retrieve documents ?

	big	count	create	dataset	differnt	features	hello	is	james	my	name	notebook	of	python	this	to	try	trying	vectorizer	words
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	0	0	0	0	0
2	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0
3	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	1
4	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0

Sparse vs Dense retrieval

Sparse embedding
(lots of 0)

Sparse Retrieval (TF IDF, BM 25, …) are methods to retrieve similar documents based on **keywords only.**
Dense Retrieval (Cos Sim, Euclidean distance) allows to retrieve document using semantic embedding representation of documents and query.

Hybrid Search is a method involving both sparse and dense retrievers to provide both advantages of the two approaches

Q. If I want to retrieve document based on the user query '*LeCun Meta*', what kind of retriever do I use ?
Q. If I want to retrieve document based on the user query '*What are the most wonderful shots of Lebron James ?*', what kind of retriever do I use ?
Q. If I want to retrieve document based on the user query '*What is the capital city of the biggest city in the world ?*', what kind of retriever do I use ?

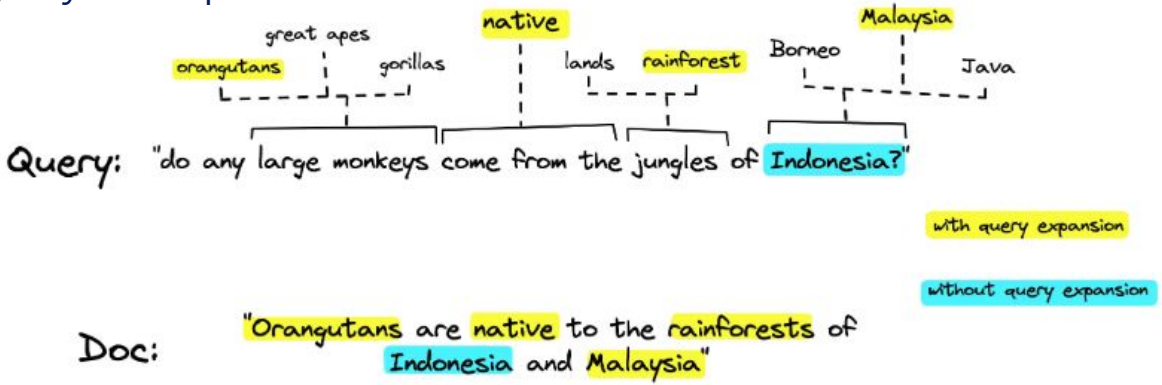
How to retrieve documents ?

Sparse Lexical and Expansion (SPLADE)

Vector database are super efficient compare to Splade at the moment
With sparse methods, you cannot get synonyms from a words.

SPLADE:

- Use Bert to get similar words like synonyms
- Provide these synonyms to a sparse methods



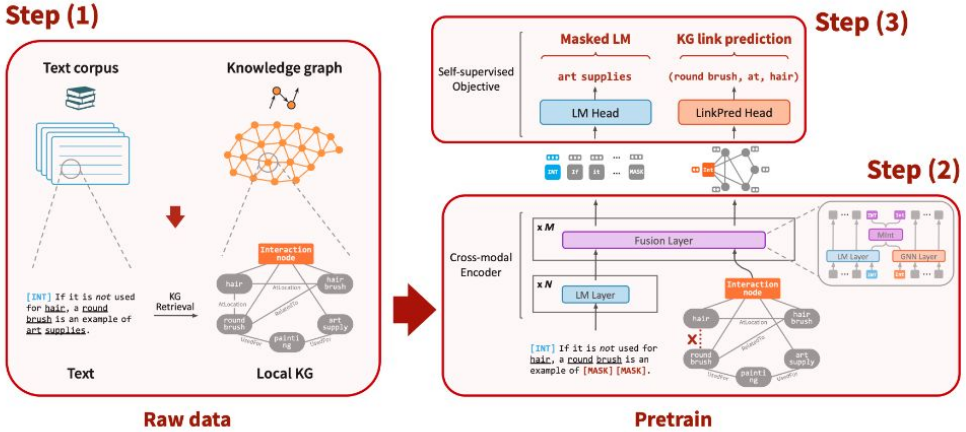
SPLADE for Sparse Vector Search Explained

Formal & AI, 2021, SPLADE V2

How to retrieve documents ?

Deep Bidirectional Language-Knowledge Graph Pretraining (DRAGON)

Dense retriever
Progressive Data Augmentation strategy for training sampling very difficult negatives



Lin & Al, 2023, How to Train Your DRAGON

Yasunaga, 2023, DRAGON: Training a Foundation Model from Text and Knowledge Graph [Blog]

How to retrieve documents ?

Best retrieval methods

Leaderboard for best Information Retrieval methods:
https://eval.ai/web/challenges/challenge-page/1897/leaderboard/4475

B - Baseline * - Private V - Verified Visible Metrics

Rank	Participant team	Avg (↑)	TREC-COVID (↑)	BioASQ (↑)	NFCorpus (↑)	NQ (↑)	HotpotQA (↑)	FiQA (↑)	Signal-1M (↑)	TREC-NEWS (↑)	Robust04 (↑)	ArguAna (↑)	Touche-2020 (↑)
1	nle ((BM25+SPLADE) RANKT5 top 50)	0.553	0.839	0.593	0.382	0.637	0.769	0.451	0.337	0.520	0.596	0.554	0.361
2	ZA+NM+Unicamp (InParsv2)	0.546	0.846	0.595	0.385	0.638	0.791	0.509	0.308	0.490	0.632	0.369	0.291
3	MetaAI+UW*2 (DRAGON+)	0.474	0.759	0.433	0.339	0.537	0.662	0.356	0.301	0.444	0.479	0.469	0.263
4	BEIR (SPLADE) V B	0.474	0.711	0.504	0.345	0.544	0.686	0.351	0.296	0.394	0.458	0.521	0.244
5	BEIR (BM25 multifield) B V	0.429	0.656	0.465	0.325	0.329	0.603	0.236	0.330	0.398	0.407	0.414	0.367

How to retrieve documents ?

Vector Database

Definition: A vector database is a specialized database designed to store, manage, and query high-dimensional vector embeddings of data such as text, images, or other content types.
These embeddings are numerical representations produced by machine learning models that capture the semantic meaning of the data.

A comparison of leading vector databases

	Pinecone	Weaviate	Milvus	Qdrant	Chroma	Elasticsearch	PGvector
Is open source	✗	✓	✓	✓	✓	✗	✓
Self-host	✗	✓	✓	✓	✓	✓	✓
Cloud management	✓	✓	✓	✓	✗	✓	(✓)
Purpose-built for Vectors	✓	✓	✓	✓	✓	✗	✗
Developer experience	👍👍👍	👍👍	👍👍	👍👍	👍👍	👍	👍
Community	Community page & events	8k☆ github, 4k slack	23k☆ github, 4k slack	13k☆ github, 3k discord	9k☆ github, 6k discord	23k slack	6k☆ github
Queries per second (using text nytimes-256-angular)	150 *for p2, but more pods can be added	791	2406	326	?	700-100 *from various reports	141
Latency, ms (Recall/Percentile 95 (millis), nytimes-256-angular)	1 *batched search, 0.99 recall, 200k SBERT	2	1	4	?	?	8
Supported index types	?	HNSW	Multiple (11 total)	HNSW	HNSW	HNSW	HNSW/IVFFlat
Hybrid Search (i.e. scalar filtering)	✓	✓	✓	✓	✓	✓	✓
Disk index support	✓	✓	✓	✓	✓	✗	✓
Role-based access control	✓	✗	✓	✗	✗	✓	✗
Dynamic segment placement vs. static data sharding	?	Static sharding	Dynamic segment placement	Static sharding	Dynamic segment placement	Static sharding	-
Free hosted tier	✓	✓	✓	(free self-hosted)	(free self-hosted)	(free self-hosted)	(varies)
Pricing (50k vectors @1536)	$70	fr. $25	fr. $65	est. $9	Varies	$95	Varies
Pricing (20M vectors, 20M req. @768)	$227 ($2074 for high performance)	$1536	fr. $309 ($2291 for high performance)	fr. $281 ($820 for high performance)	Varies	est. $1225	Varies

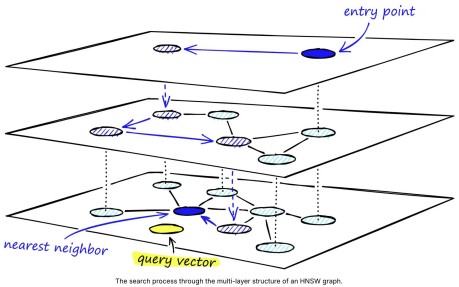
Vector DB Comparison

How to optimize ?

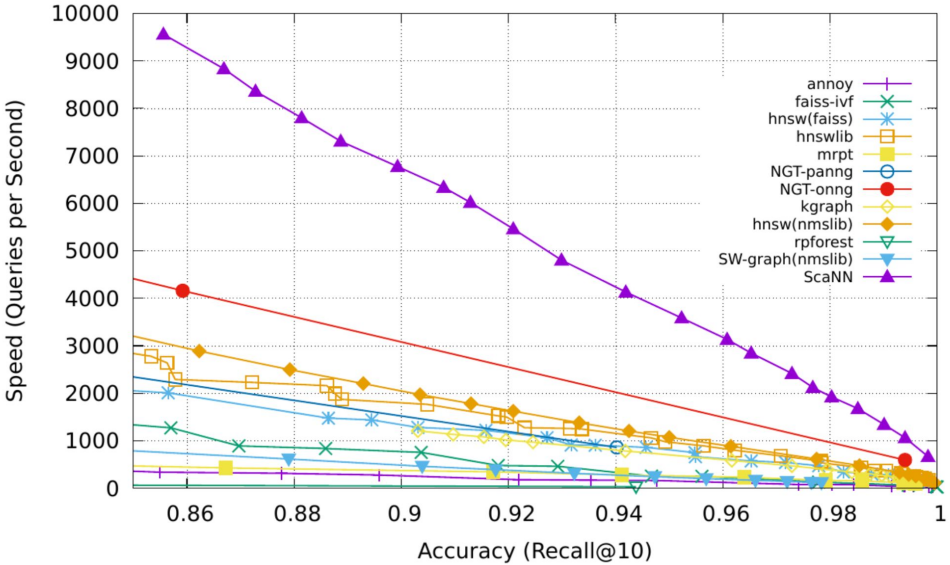
Efficient similarity search

- Scalable Nearest Neighbors (ScaNN) - Google
- Facebook AI Similarity Search (FAISS)
- Hierarchical Navigable Small Worlds (HNSW)

Definition:
ScaNN, FAISS and HNSW are methods for retrieving similar embeddings based on vector quantization and ANN search instead of full scan search.



Hierarchical Navigable Small Worlds (HNSW) [Blog]



Announcing ScaNN: Efficient Vector Similarity Search

How to provide context ?

Reciprocal Rank Fusion (RRF)

Definition:
RRF allows to merge results of different retrievers

$$\mathrm{RRF}(d) = \sum_{s=1}^{S} \frac{1}{k + r_s(d)}$$

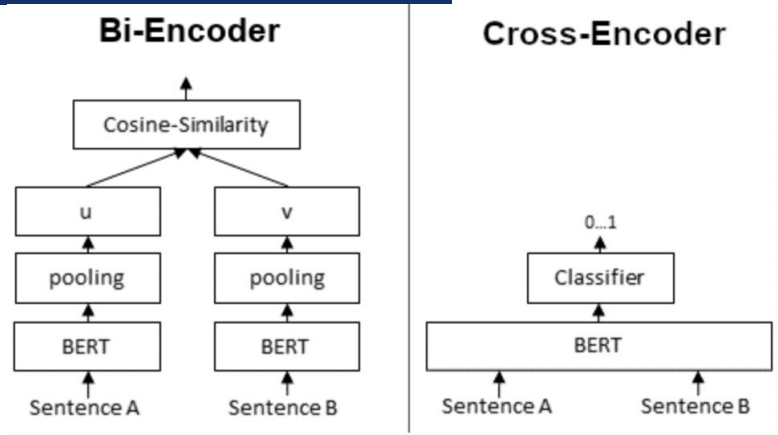
$\mathrm{RRF}(d)$: Denotes the RRF score for document d

$\sum_{s=1}^{S}$: Summation over all systems from s = 1 to S

$\frac{1}{k + r_s(d)}$: The reciprocal rank for document d in system s , adjusted by the constant k

How to provide context ?

Reranker



A reranker ranks retrieved documents after a first similarity search

Reranker type:

- **Cross-Encoders**
- **Neural Rerankers**

Benefits of Using a Reranker:

- Increased Accuracy: Improves the likelihood that the most relevant information is used in generating the response.
- Better Contextual Understanding: Helps the system understand subtle nuances in the query.

Challenges:

- Computational Overhead: Additional processing can increase response time.
- Resource Intensive: Advanced models require significant computational resources.

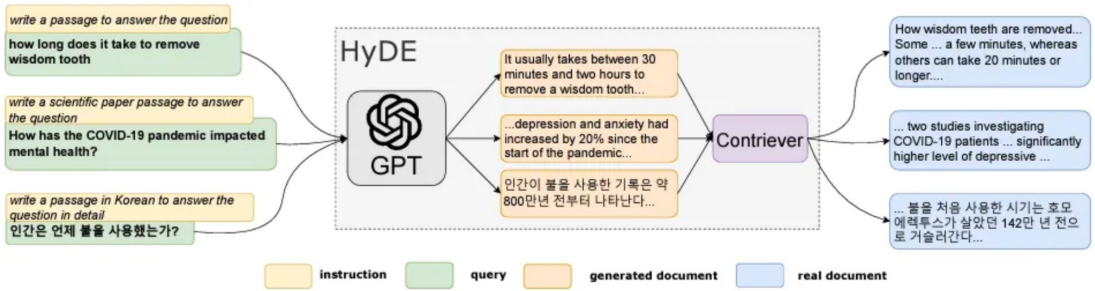
Bi Encoder vs Cross Encoder

How to augment user query ?

Query augmentation

Definition: query augmentation refers to the process of enhancing or expanding the user's original query to improve the retrieval of relevant documents or information from a knowledge base.
By augmenting the query, the system aims to retrieve more comprehensive and pertinent data, which can then be used to generate more accurate and informative responses.

HyDE: Generate a fake answer from a query to improve information retrieval

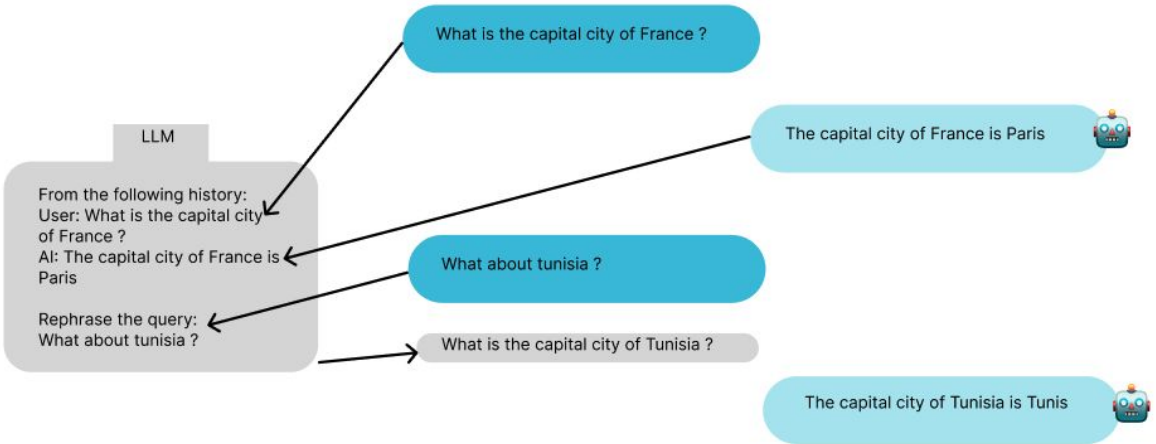


Luyu Gao & Al, 2022 Precise Zero-Shot Dense Retrieval without Relevance Labels

How to augment user query ?

Query rephrasing

Query rephrasing can be used to rephrase the query from the conversation history



How to provide context ?

Lost In the Middle

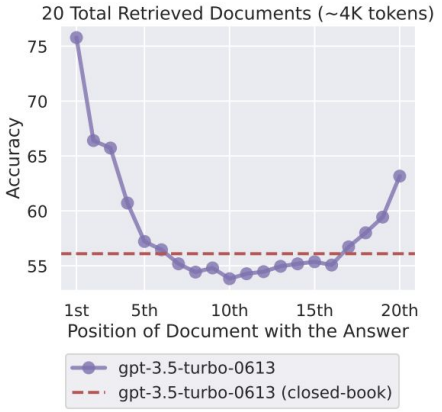
Retrieved context provided at the beginning or the end of the prompt have more impact on the answer

```python
from langchain.chains import LLMChain, StuffDocumentsChain
from langchain_chroma import Chroma
from langchain_community.document_transformers import (
    LongContextReorder,
)
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_core.prompts import PromptTemplate
from langchain_openai import OpenAI

# Get embeddings.
embeddings = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")

texts = [
    "Basquetball is a great sport.",
    "Fly me to the moon is one of my favourite songs.",
    "The Celtics are my favourite team.",
    "This is a document about the Boston Celtics",
    "I simply love going to the movies",
    "The Boston Celtics won the game by 20 points",
    "This is just a random text.",
    "Elden Ring is one of the best games in the last 15 years.",
    "L. Kornet is one of the best Celtics players.",
    "Larry Bird was an iconic NBA player.",
]

# Create a retriever
retriever = Chroma.from_texts(texts, embedding=embeddings).as_retriever(
    search_kwargs={"k": 10}
)
query = "What can you tell me about the Celtics?"

# Get relevant documents ordered by relevance score
docs = retriever.invoke(query)
docs
```

```
[Document(page_content='This is a document about the Boston Celtics'),
 Document(page_content='The Celtics are my favourite team.'),
 Document(page_content='L. Kornet is one of the best Celtics players.'),
 Document(page_content='The Boston Celtics won the game by 20 points'),
 Document(page_content='Larry Bird was an iconic NBA player.'),
 Document(page_content='Elden Ring is one of the best games in the last 15 years.'),
 Document(page_content='Basquetball is a great sport.'),
 Document(page_content='I simply love going to the movies'),
 Document(page_content='Fly me to the moon is one of my favourite songs.'),
 Document(page_content='This is just a random text.')]
```

```python
# Reorder the documents:
# Less relevant document will be at the middle of the list and more
# relevant elements at beginning / end.
reordering = LongContextReorder()
reordered_docs = reordering.transform_documents(docs)

# Confirm that the 4 relevant documents are at beginning and end.
reordered_docs
```

```
[Document(page_content='The Celtics are my favourite team.'),
 Document(page_content='The Boston Celtics won the game by 20 points'),
 Document(page_content='Elden Ring is one of the best games in the last 15 years.'),
 Document(page_content='I simply love going to the movies'),
 Document(page_content='This is just a random text.'),
 Document(page_content='Fly me to the moon is one of my favourite songs.'),
 Document(page_content='Basquetball is a great sport.'),
 Document(page_content='Larry Bird was an iconic NBA player.'),
 Document(page_content='L. Kornet is one of the best Celtics players.'),
 Document(page_content='This is a document about the Boston Celtics')]
```



LangChain, Long Context Reorder [Blog]

F. Liu & Al, 2023, Lost in the Middle: How Language Models Use Long Contexts

How to prompt ?

Prompt Engineering

- Write clear instructions
- Provide reference text
- Split complex tasks into simpler subtasks
- Give the model time to "think"
- Use external tools (RAG)

Tactic:

- Ask the model to adopt a persona
- Use delimiters to clearly indicate distinct parts of the input
- Specify the steps required to complete a task
- Provide examples
- Specify the desired length of the output

Open AI, Prompt engineering

How to load documents ?

Document Loader

Load any type of document (PDF, PPT(x), DOC(x), XLS(x)

Unstructured: https://unstructured.io/
LLama Parse: https://llamahub.ai/l/readers/llama-index-readers-llama-parse?from=readers

Context

Definition: The context size refers to the maximum number of tokens (words or subword units) that the model can process in a single input sequence. It determines how much textual information the model can consider at once when generating responses or predictions.

- A larger context size allows the model to capture longer dependencies and understand more extensive context within the input, leading to more coherent and relevant outputs.
- A smaller context size limits the amount of information the model can utilize from the input text.

Variable Sequence Length Training for Long-Context Large Language Models

How to chunk documents ?

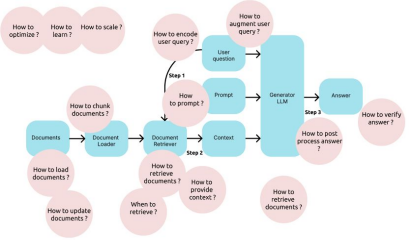
Chunking

To avoid context limitations, we can do document chunking:

- Chunk by document if the document is small
- Chunk by title or header if the document is big

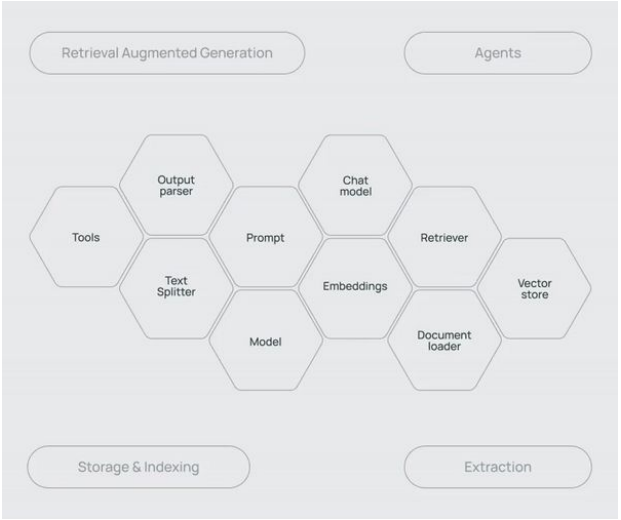
```
from langchain.text_splitter import CharacterTextSplitter
text_splitter = CharacterTextSplitter(
    separator = "\n\n",
    chunk_size = 1000,
    chunk_overlap  = 200,
    length_function = len,
    is_separator_regex = False,
)
```

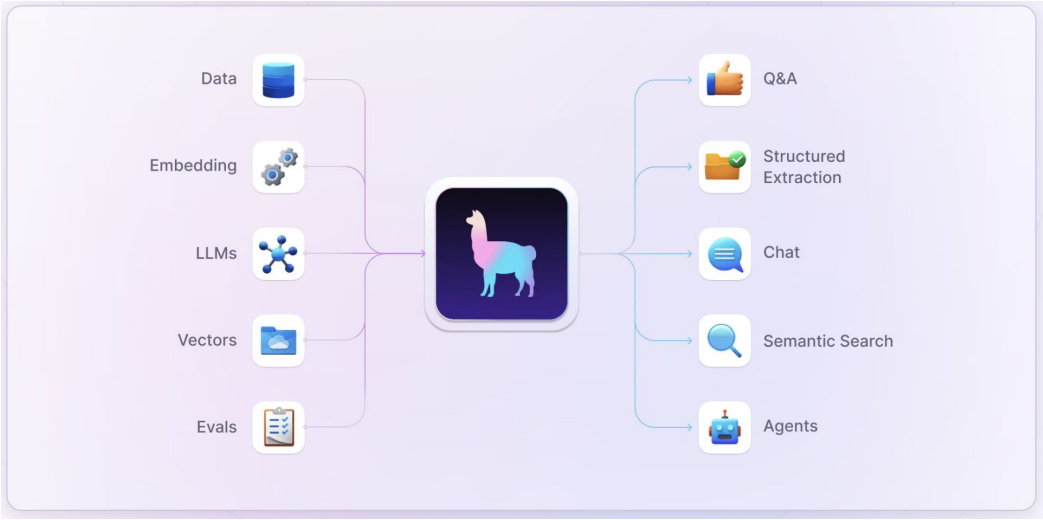
Announcing ScaNN: Efficient Vector Similarity Search



RAG Frameworks in Python

LangChain





Langchain

LlamaIndex

How to scale ?

Cloud services

Cloud Services provide:

- A Secure environment
- Enough compute to train big models
- Product as a Service (PaaS)
  - LLMs APIs
  - Vector Store management
  - Efficient Retrieval
- Monitoring tools
- …

How to verify answer ?

Retriever Evaluation: Precision & Recall @ k

Precision @k
How many **retrieved** documents are **relevant** ?

$$\text{Precision at } K = \frac{\text{Number of relevant items in } K}{\text{Total number of items in } K}$$

Recall @k
How many **relevant** documents are **retrieved** ?

$$\text{Recall at } K = \frac{\text{Number of relevant items in } K}{\text{Total number of relevant items}}$$

How to verify answer ?

Retriever Evaluation : NDCG

NDCG can take values from 0 to 1.

- NDCG equals 1 in the case of ideal ranking when items are perfectly sorted by relevance.
- NDCG equals 0 when there are no relevant objects in top-K.
- NDCG can be between 0 and 1 in all other cases. The higher the NDCG, the better.

$$\text{DCG@K} = \sum_{k=1}^{K} \frac{rel_i}{\log_2(i+1)}$$

$$\text{NDCG@K} = \frac{DCG@K}{IDCG@K}$$

How to verify answer ?

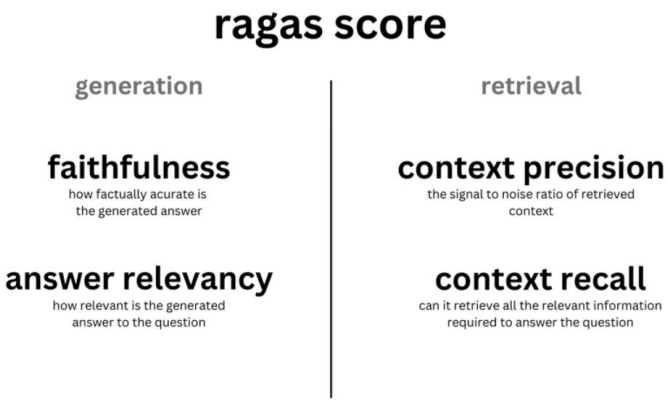
Answer Evaluation: LLM As a judge

Ask an LLM to evaluate answer quality:

- Does my answer answer to the question ?
- Does my answer used information from the context ?
- Does my answer give enough facts ?
- …

BLEU, ROUGE, Perplexity are not ideal for RAG use case.

Evaluating answers in RAG is not easy

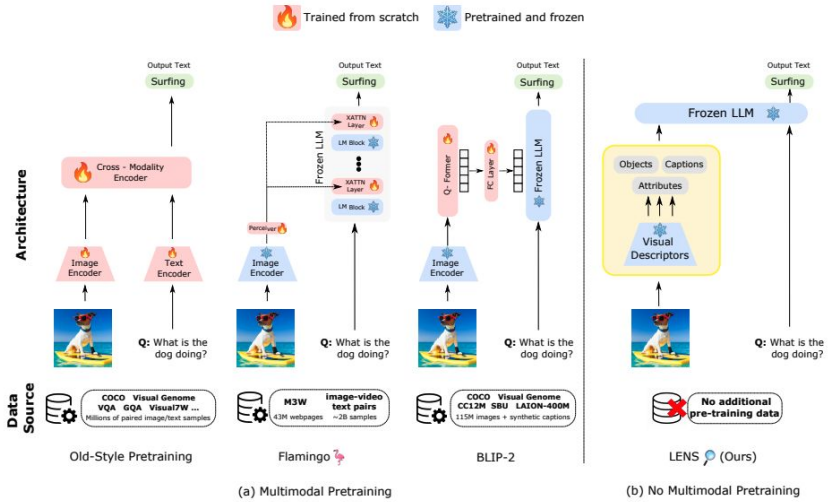


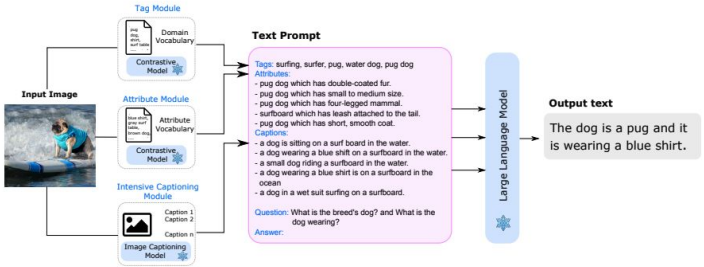
RAGAS

Multimodal

Multimodal LLM: LLM capable of processing and understanding multiple types (or “modes”) of input data, such as **text**, **images**, **audio**, **video**, and other sensory inputs, in a unified manner.

Exemple: GPT 4o, Gemini 1.5, Qwen2- VL





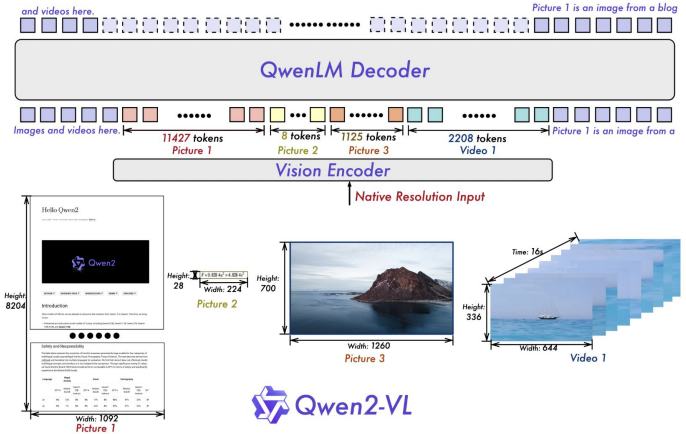
Berrios & Al, 2023, Towards Language Models That Can See: Computer Vision Through the LENS of Natural Language

Multimodal: Qwen2- VL

Outperform GPT 4o on most of the benchmarks

Multimodal Rotary Position (M-ROPE): "By deconstructing the original rotary embedding into three parts representing temporal and spatial (height and width) information, M-ROPE enables LLM to concurrently capture and integrate 1D textual, 2D visual, and 3D video positional information."

		Qwen2-VL-72B	GPT-4o-0513	Claude3.5-Sonnet	Other Best Model
College-level Problems	MMMU	64.5	**69.2**	68.3	66.1 (Gemini-2)
Mathematical Reasoning	MathVista	**70.5**	63.8	67.7	69.0 (Gemini-2)
	MATH-Vision	25.9	**30.4**	-	30.3 (GPT-4 Turbo)
Document and Diagrams Reading	DocVQA	**96.5**	92.8	95.2	94.1 (InternVL2-76B)
	ChartQA	88.3	85.7	**90.8**	88.4 (InternVL2-76B)
	OCRBench	**855**	736	788	852 (MiniCPM-V 2.6)
	MTVQA	**32.6**	27.8	25.7	23.2 (Qwen-VL-1.0-Ultra)
	InfoVQA	**84.5**	-	-	82.0 (InternVL2-76B)
	TextVQA	**85.5**	-	-	84.4 (InternVL2-76B)
General Visual Question Answering	RealWorldQA	**77.8**	75.4	60.1	72.2 (InternVL2-76B)
	MMStar	**68.3**	63.9	62.2	67.1 (InternVL2-76B)
	MMVet	**74.0**	69.1	66.0	67.5 (GPT-4o)
	MMT-Bench	**71.7**	65.5	-	63.4 (InternVL2.5)
	MMBench-1.1	**85.9**	82.2	78.5	85.5 (InternVL2-76B)
	MME	**2482.7**	2328.7	1920.0	2414.7 (InternVL2-76B)
	HallBench	**58.1**	55.0	49.9	55.2 (InternVL2-76B)
Video Understanding	MVBench	**73.6**	-	-	69.6 (InternVL2-76B)
	EgoSchema	**77.9**	72.2	-	72.2 (Gemini 1.5 Pro)
	PerceptionTest	**68.0**	-	-	66.9 (LLaVA-OneVision-72B)
	Video-MME w/o subs	71.2	71.9	60.0	**75.0** (Gemini 1.5 Pro)
	Video-MME w/ subs	77.8	77.2	-	**81.3** (Gemini 1.5 Pro)
Visual Agent	FnCall	**93.1**	90.2	-	-
	AITZ	**89.6**	70.0	-	83.0 (AITZ)
	Gym Cards	**61.7**	53.6	-	45.5 (BLAVA)
	ALFRED	**67.8**	-	-	67.7 (Claude3.5-Sonnet)



Berrios & Al, 2023, Towards Language Models That Can See: Computer Vision Through the LENS of Natural Language

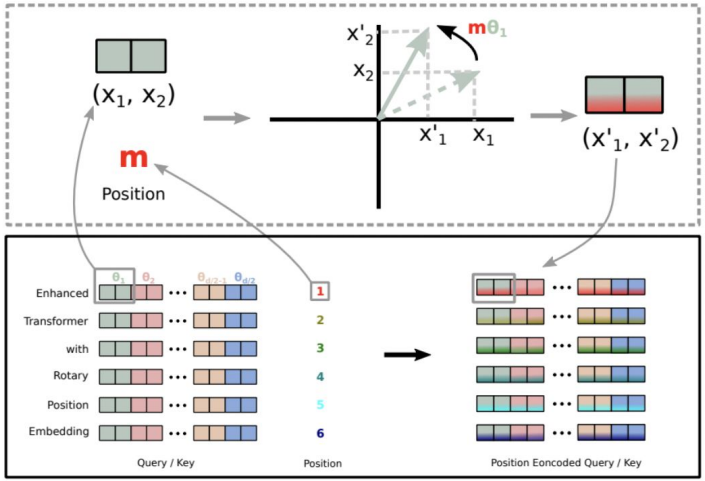
Rotary Embedding (ROPE)

Rotary Position Embedding, or RoPE, is a type of position embedding which encodes absolute positional information with rotation matrix and naturally incorporates explicit relative position dependency in self-attention formulation.

Unlike traditional position embeddings, which add fixed vectors to represent positions, RoPE encodes positional information directly into the attention mechanism by rotating the query and key vectors in the Transformer architecture. This approach allows the model to better handle long-range dependencies while maintaining the flexibility of the attention mechanism.

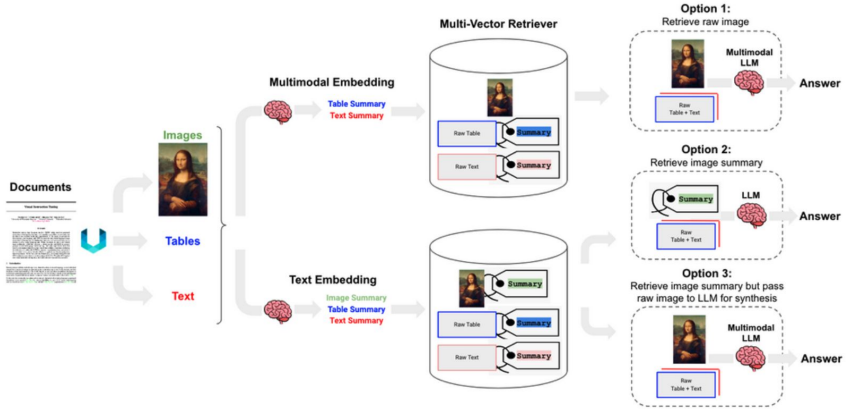
Properties:

- Flexibility of being expand to any sequence lengths
- Decaying inter-token dependency with increasing relative distances
- Capability of equipping the linear self-attention with relative position encoding.



Su & AL. 2023. RoFormer: Enhanced Transformer with Rotary Position Embedding

Multimodal RAG



Option 1: Store raw image using multimodal embedding. Retrieve images based on multimodal embedding similarity. Provide the raw image to the generator.

Option 2: Store the summary of the image using a multimodal LLM. Retrieve images based on its summary embedding. Provide the **summary** to the generator.

Option 3: Store the image and its summary using a multimodal LLM. Retrieve images based on its summary embedding. Provide the **image** to the generator.

LangChain, Multi-Vector Retriever for RAG on tables, text, and images

Multimodal RAG

Michihiro Yasunaga,[1 *] Armen Aghajanyan,[2] Weijia Shi,[3] Rich James,[2] Jure Leskovec,[1] Percy Liang [1]
Mike Lewis,[2] Luke Zettlemoyer,[3 2] Wen-tau Yih [2]

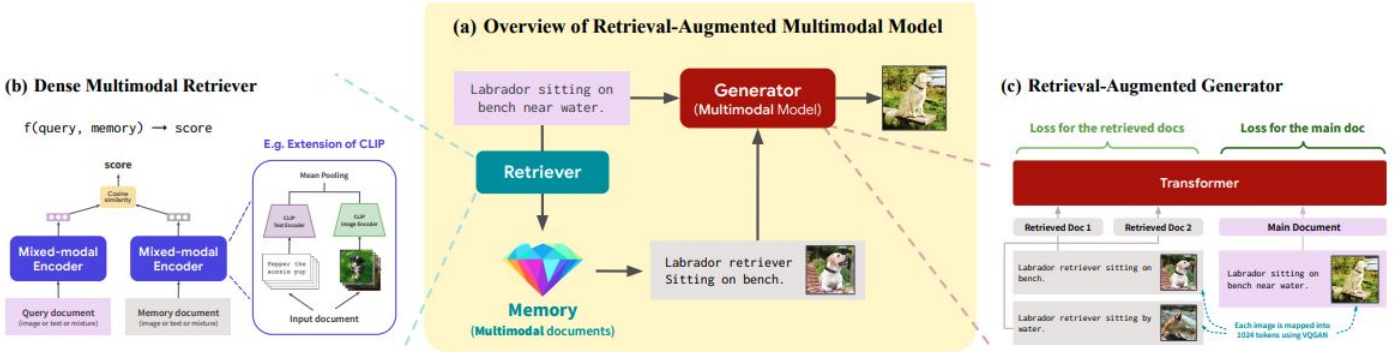
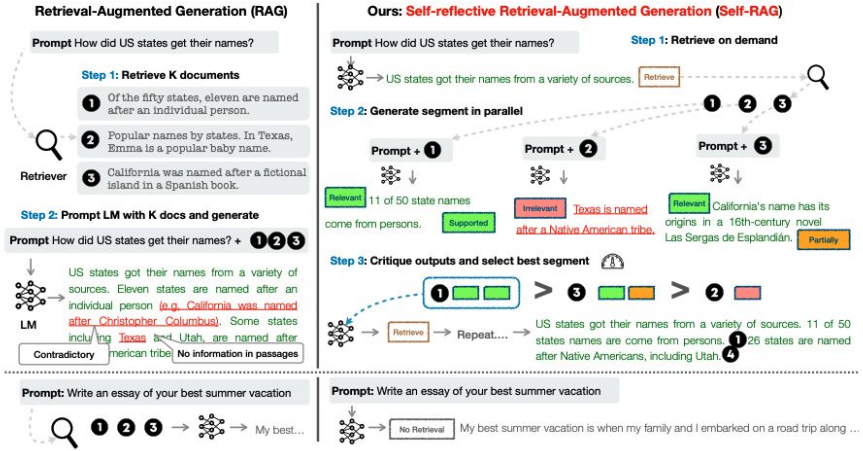


Figure 1. **Our approach, retrieval-augmented multimodal modeling**. **(a)** Overview: given an input multimodal document, we use a **retriever** to retrieve relevant multimodal documents from an external memory, and use the **generator** to refer to the retrieved documents and make predictions for the input (e.g., generate the continuation). **(b)** The multimodal retriever is a dense retriever with a mixed-modal encoder that can encode mixture of text and images (§3.2). **(c)** The retrieval-augmented generator uses the CM3 Transformer architecture, and we prepend the retrieved documents to the main input document that we feed into the model (§3.3).

Yasunaga & Al, 2023, Retrieval-Augmented Multimodal Language Modeling

Self-RAG

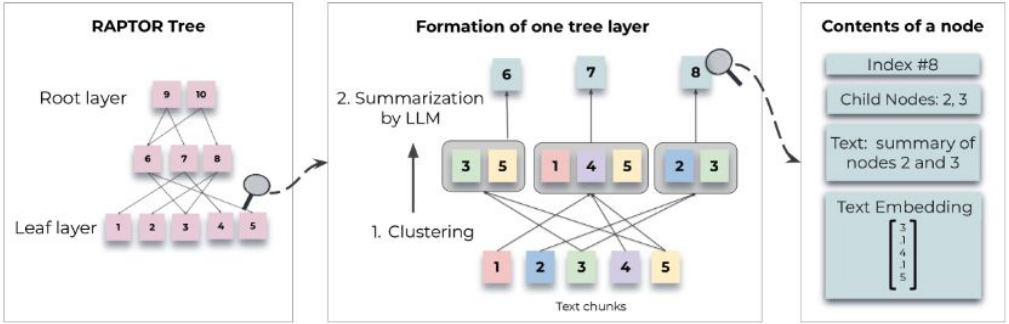
- Generates multiple possible response segments in parallel, utilizing the retrieved documents as context.
- The model ranks the generated segments based on their critique scores, selecting the most accurate and relevant segment as the final output.
- This selection process ensures that the response is both factually correct and contextually appropriate.



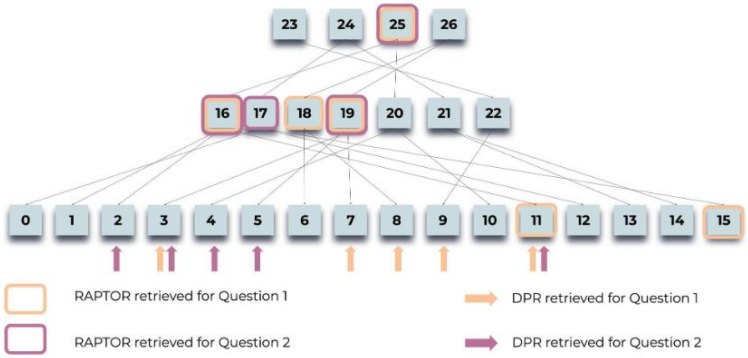
Asai & Al, 2023, Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection

Ahmed, 2024, SELF-RAG (Self-Reflective Retrieval-Augmented Generation): The Game-Changer in Factual AI Generation [Blog]

RAPTOR



- RAPTOR recursively summarizes retrieved documents. Instead of processing the full text of multiple documents directly, it creates concise summaries that retain the most important information at each recursive step.
- This hierarchy of summaries reduces the amount of information that needs to be processed while preserving the context and key facts from the original documents.

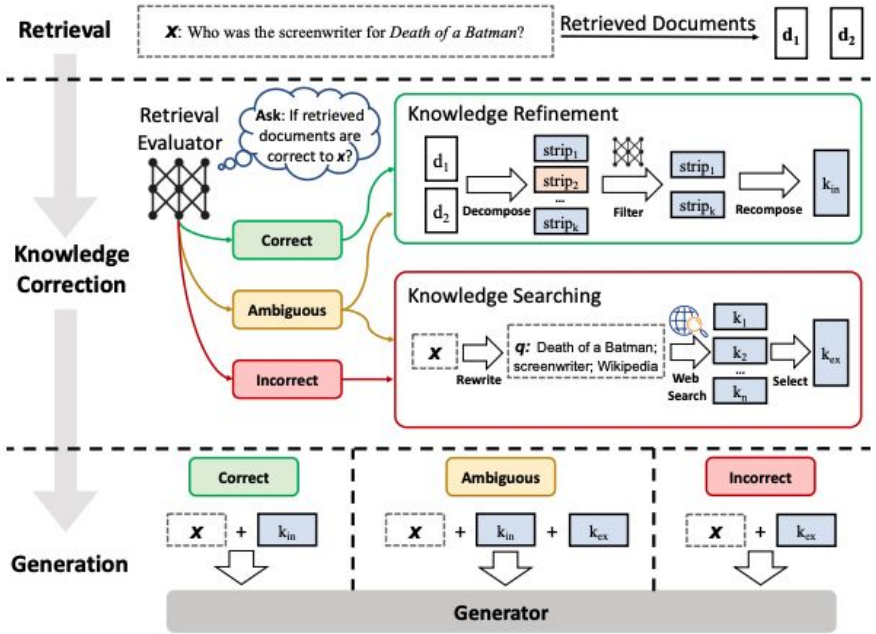


Sarthi & AL, 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval

Ahmed, 2024. SELF-RAG (Self-Reflective Retrieval-Augmented Generation): The Game-Changer in Factual AI Generation [Blog]

Corrective RAG (CRAG)

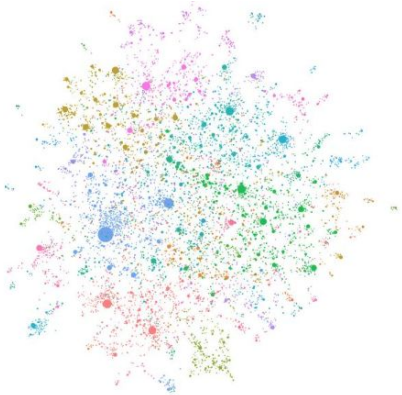
- Add a retrieval evaluator based on the quality of retrieved sources
- If sources are considered as incorrect, or ambiguous, augment or replace the context by a Web Search query



Yan & Al, 2024, Corrective Retrieval Augmented Generation

GraphRAG

- The LLM processes the entire private dataset, creating references to all entities and relationships within the source data, which are then used to create an LLM-generated knowledge graph.
- This graph is then used to create a bottom-up clustering that organizes the data hierarchically into semantic clusters. This partitioning allows for pre-summarization of semantic concepts and themes, which aids in holistic understanding of the dataset.
- At query time, both of these structures are used to provide materials for the LLM context window when answering a question.



Microsoft, 2024, GraphRAG: Unlocking LLM discovery on narrative private data

Conclusion

Simple RAG: Encodes document content into a vector store, enabling quick retrieval of relevant information to enhance model responses.
Context Enrichment: Adds surrounding context to each retrieved chunk, improving the coherence and completeness of the returned information.
Multi-faceted Filtering: Applies various filtering techniques (metadata, similarity thresholds etc.) to refine and improve the quality of retrieved results.
Fusion Retrieval: Combines vector-based similarity search with keyword-based retrieval to improve document retrieval.
Intelligent Reranking: Reassesses and reorders initially retrieved documents to ensure that the most pertinent information is prioritized for subsequent processing.
Query Transformation: Modifies or expands the original query with query rewriting, step-back prompting and sub-query decomposition.
Hierarchical Indices: First identifies relevant document sections through summaries, then drills down to specific details within those sections.
Hypothetical Questions: HyDE transforms queries into hypothetical documents that contain answers, bridging the gap between query and document distributions in vector space.
Choose Chunk Size: Selects an appropriate fixed size for text chunks to balance context preservation and retrieval efficiency.
Semantic Chunking: Unlike traditional methods that split text by fixed character/word counts, semantic chunking creates more meaningful, context-aware segments.
Context Compression: Compresses and extracts the most pertinent parts of documents in the context of a given query.
Explainable Retrieval: Not only retrieves relevant documents based on a query but also provides explanations for why each retrieved document is relevant.
Retrieval w/ Feedback: Utilizes user feedback on the relevance and quality of retrieved documents and generated responses to fine-tune retrieval and ranking models.

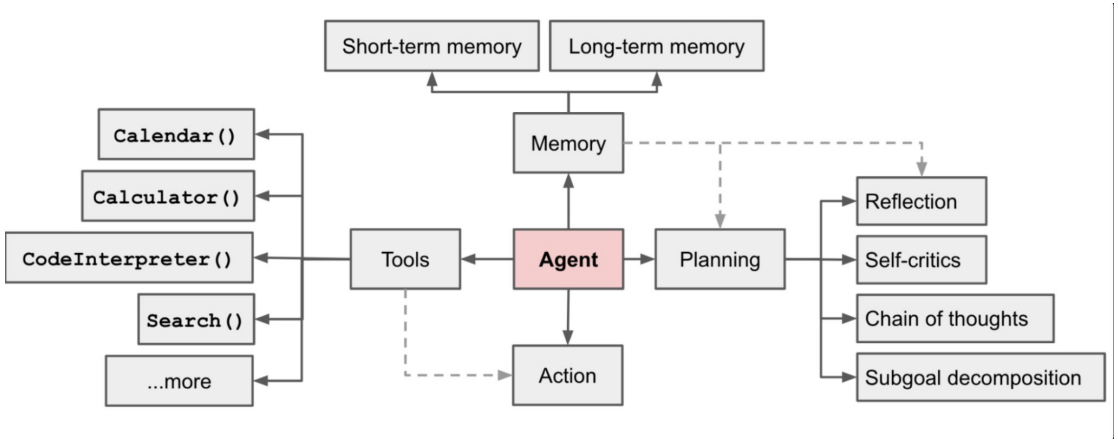
Conclusion

Adaptive Retrieval: Classifies queries into different categories and uses tailored retrieval strategies (factual, analytical, contextual etc.) for each, considering query context and preferences.
Iterative Retrieval: Analyzes initial results and generates follow-up queries to fill in gaps or clarify information.
Ensemble Retrieval: Applies different embedding models or retrieval algorithms and uses voting or weighting mechanisms to determine the final set of retrieved documents.
Graph RAG= Retrieves entities and their relationships from a knowledge graph relevant to the query, combining with unstructured text for more informative responses.
Multimodal: Integrates models that can retrieve and understand different data modalities, combining insights from text, images, and more.
RAPTOR: Uses abstractive summarization to recursively process and summarize retrieved documents, organizing the information in a tree structure for hierarchical context.
Self RAG: Multi-step process integrating decision, document retrieval, relevance filtering and generative feedback for more powerful model responses.
Corrective RAG: Dynamically evaluates and corrects the retrieval process, combining vector databases, feedback, and models to improve responses.
Few shot examples: Provides a few examples in the prompt to help the LLM understand the desired output

Go to : https://kahoot.com/
Click Play
Enter code: XXX XXX
Enter your name

IV. Beyond LLM, Tools and (Multi)-Agents

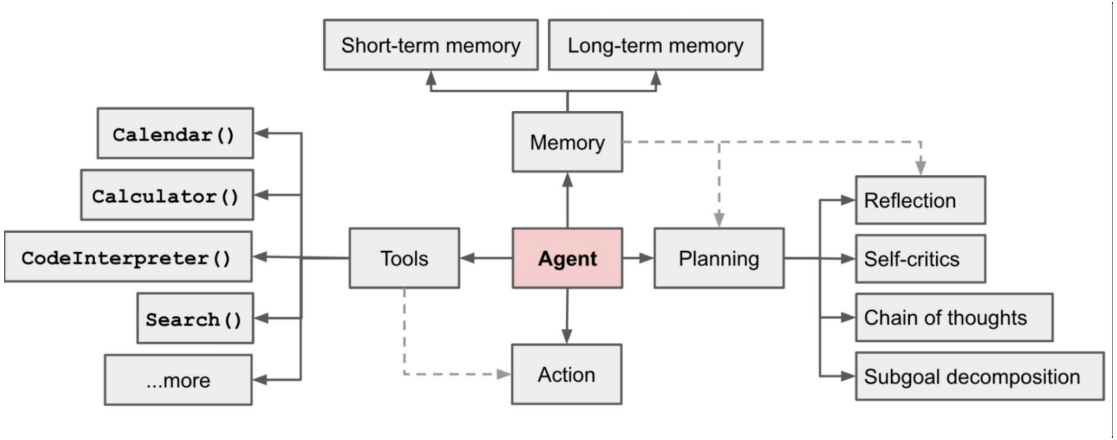
A. **Beyond LLMs**
B. **Agents & Tools**
C. **Multi Agents Systems**



Dauphine | PSL TUNIS

IV.A. Beyond LLMs

A. **Beyond LLMs**
B. **Agents & Tools**
C. **Multi Agents Systems**

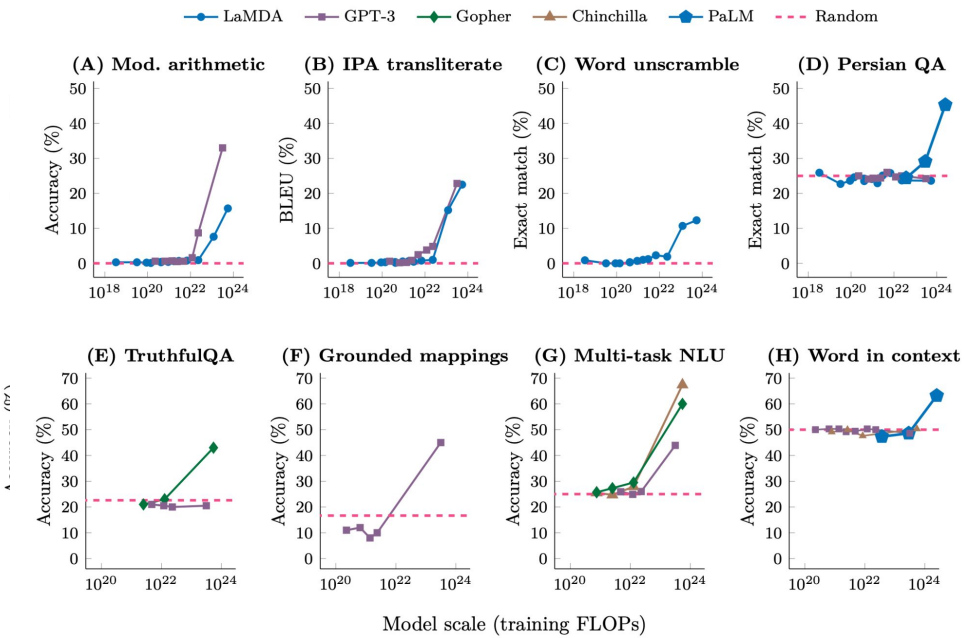


Dauphine | PSL TUNIS

Back in the past

Emergent abilities definition:
An ability is emergent if it is present in larger but no smaller models

Performance is near random until a certain critic threshold
Know as “phase transition”



Wei & Al, 2022, Emergent Abilities of Large Language Models

Table 1: List of emergent abilities of large language models and the scale (both training FLOPs and number of model parameters) at which the abilities emerge.

	Emergent scale			
	Train. FLOPs	Params.	Model	Reference
Few-shot prompting abilities				
• Addition/subtraction (3 digit)	2.3E+22	13B	GPT-3	Brown et al. (2020)
• Addition/subtraction (4-5 digit)	3.1E+23	175B		
• MMLU Benchmark (57 topic avg.)	3.1E+23	175B	GPT-3	Hendrycks et al. (2021a)
• Toxicity classification (CivilComments)	1.3E+22	7.1B	Gopher	Rae et al. (2021)
• Truthfulness (Truthful QA)	5.0E+23	280B		
• MMLU Benchmark (26 topics)	5.0E+23	280B		
• Grounded conceptual mappings	3.1E+23	175B	GPT-3	Patel & Pavlick (2022)
• MMLU Benchmark (30 topics)	5.0E+23	70B	Chinchilla	Hoffmann et al. (2022)
• Word in Context (WiC) benchmark	2.5E+24	540B	PaLM	Chowdhery et al. (2022)
• Many BIG-Bench tasks (see Appendix E)	Many	Many	Many	BIG-Bench (2022)
Augmented prompting abilities				
• Instruction following (finetuning)	1.3E+23	68B	FLAN	Wei et al. (2022a)
• Scratchpad: 8-digit addition (finetuning)	8.9E+19	40M	LaMDA	Nye et al. (2021)
• Using open-book knowledge for fact checking	1.3E+22	7.1B	Gopher	Rae et al. (2021)
• Chain-of-thought: Math word problems	1.3E+23	68B	LaMDA	Wei et al. (2022b)
• Chain-of-thought: StrategyQA	2.9E+23	62B	PaLM	Chowdhery et al. (2022)
• Differentiable search index	3.3E+22	11B	T5	Tay et al. (2022b)
• Self-consistency decoding	1.3E+23	68B	LaMDA	Wang et al. (2022b)
• Leveraging explanations in prompting	5.0E+23	280B	Gopher	Lampinen et al. (2022)
• Least-to-most prompting	3.1E+23	175B	GPT-3	Zhou et al. (2022)
• Zero-shot chain-of-thought reasoning	3.1E+23	175B	GPT-3	Kojima et al. (2022)
• Calibration via P(True)	2.6E+23	52B	Anthropic	Kadavath et al. (2022)
• Multilingual chain-of-thought reasoning	2.9E+23	62B	PaLM	Shi et al. (2022)

Wei & Al, 2022, Emergent Abilities of Large Language Models

Increase model abilities

- Further scaling to train larger model with new emergent abilities
- New architecture, higher quality data, new training procedure could enable emergent abilities
- Further research to make the abilities available for small models

Emergent Abilities limitations

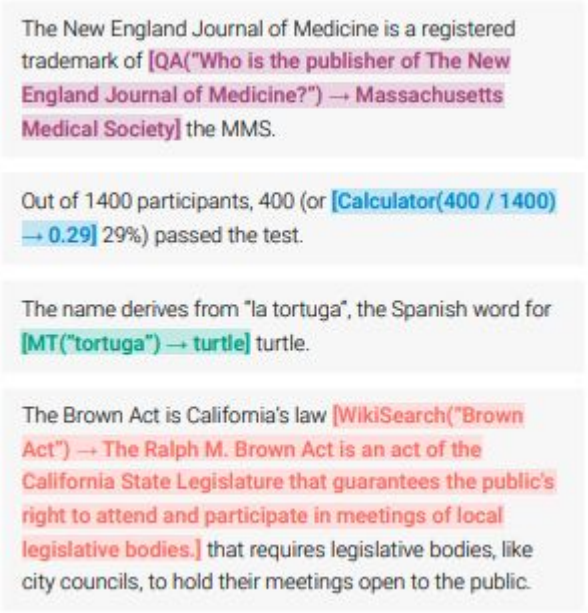
- Societal risk with toxicity, bias and truthfulness
- Limitations with scaling LLM: Training data memorization (“overfitting”), toxicity and bias
- Other limitations we don’t have discovered yet

Sociological limitations

- Adoption of LLM, use of these models
- LLM used for a large range of tasks
  - Text
  - Image
  - Audio
  - Classification
  - …

The future of Large Language Models:

- Generalist Agents
- Longer video understanding and generation
- Incredibly long sequence modeling (GPT authors a novel)
- Domain specific foundation models (Doctor GPT, Lawyer GPT)
- Real world impacts
  - Personalized education and tutoring systems
  - Advanced healthcare diagnostics
  - Real time multilingual communication



Shick & Al, 2023, Toolformer: Language Models Can Teach Themselves to Use Tools

The future (What is missing):

Missing fonctionnalites to Artificial General Intelligence (AGI)

- Reducing computation complexity
- Enhance human controllability
- Adaptive Learning
- Infinite external memory
- Long horizon decision making
- Multi-sensory embodiment (physics and commonsense)

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Shick & Al, 2023, Toolformer: Language Models Can Teach Themselves to Use Tools

Chain-of-Thought (CoT)

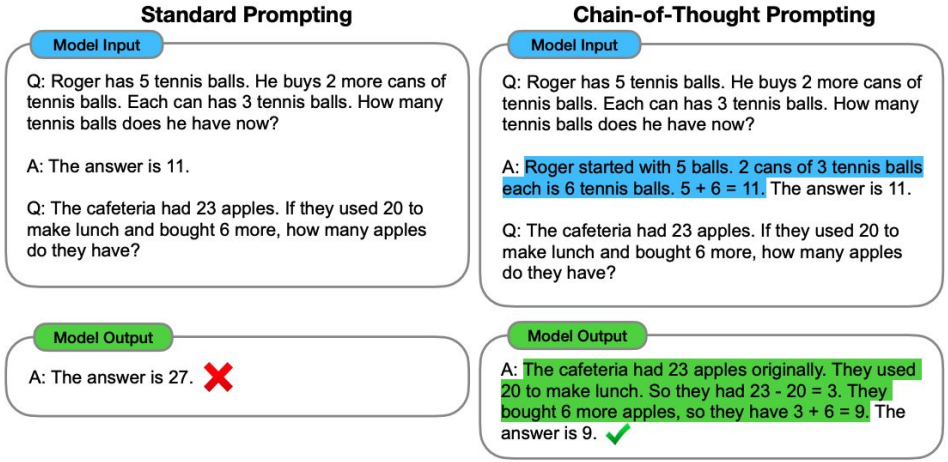
Definition: 'a series of intermediate reasoning steps significantly improves the ability of large language models to perform complex reasoning'

Example with one shot prompting:

"If Alice has 5 apples, and she buys 3 more, how many apples does she have in total?"

Without Chain-of-Thought (direct answer):
"8"

With Chain-of-Thought:
"Alice starts with 5 apples. She buys 3 more apples. So, 5 + 3 equals 8. Therefore, Alice has 8 apples."



Wei & Al, 2023, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Chain-of-Thought (CoT)

CoT results in performance gains for larger LMs, but still have a non negligible fraction of errors

CoT works for 100B + params models
CoT fail for smaller models: “one step missing” or “semantic understanding” CoT errors are the most commons on smaller models

3 potentials reasons:

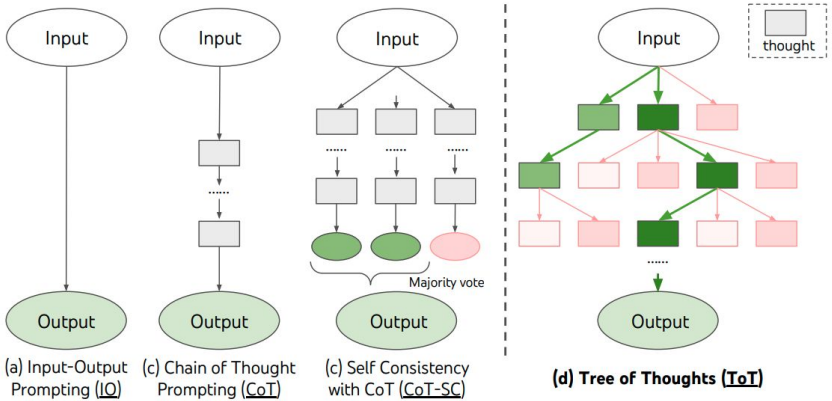
- Weaker arithmetic abilities
- Logical loopholes and do not arrive at final answer

New research area: Generalizing CoT for small models

Wei & Al, 2023, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Tree of Thought (ToT)

Definition: 'Consider multiple different reasoning paths and self-evaluation choices to decide the next course of action, as well as looking ahead or backtracking when necessary to make global choices'



Yao & Al, 2024, Tree of Thoughts: Deliberate Problem Solving with Large Language Models

Program-Aided Language Model (PAL)

PAL add an intermediate reasoning step to transform a problem into Python code.

PAL distinguishes **reasoning** and **solving** in two separate steps

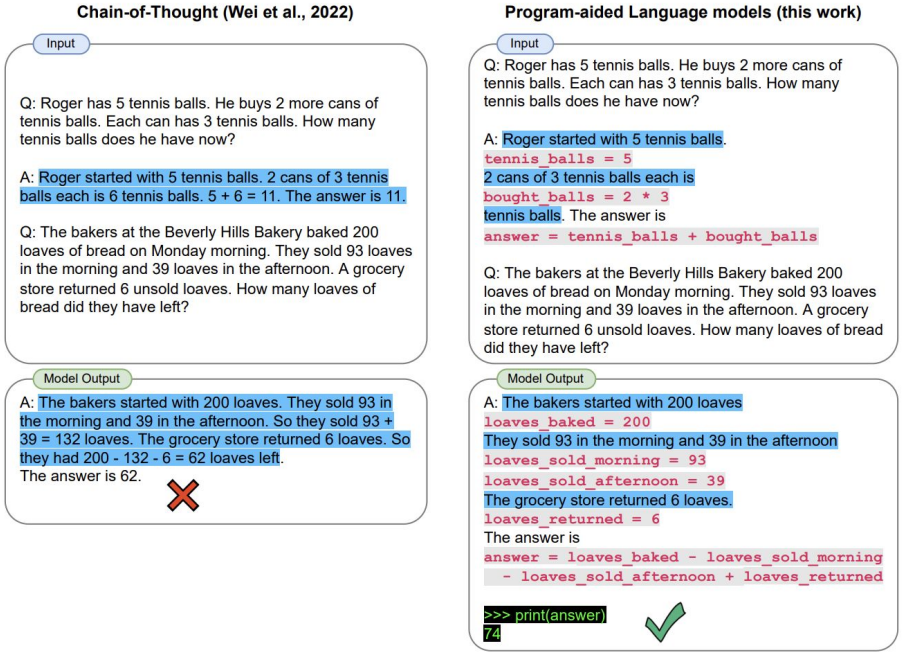


Figure 1: A diagram illustrating PAL: Given a mathematical reasoning question, Chain-of-thought (left) generates intermediate reasoning steps of free-form text. In contrast, Program-aided Language models (PAL, right) generate intermediate steps *and* Python code. This shifts the role of *running* the reasoning steps from the language model to the Python interpreter. The final answer is obtained by running the generated reasoning chain. Chain-of-thought reasoning is highlighted in blue; PAL steps are `highlighted in gray and pink`; the Python interpreter run is highlighted in black and green.

Gao & Al, 2022, PAL: Program-aided Language Models

Program-Aided Language Model (PAL)

```
Q: Jane was born on the last day of Feburary in 2001. Today is her 16-year-old
↪  birthday. What is the date 24 hours later in MM/DD/YYYY?

A: The last day of February is the 28th, so Jane was born on 02/28/2001. Today is her
↪  16-year old birthday, so today is 02/28/2017. So 24 hours later is 02/29/2017. So
↪  the answer is 02/29/2017.
```

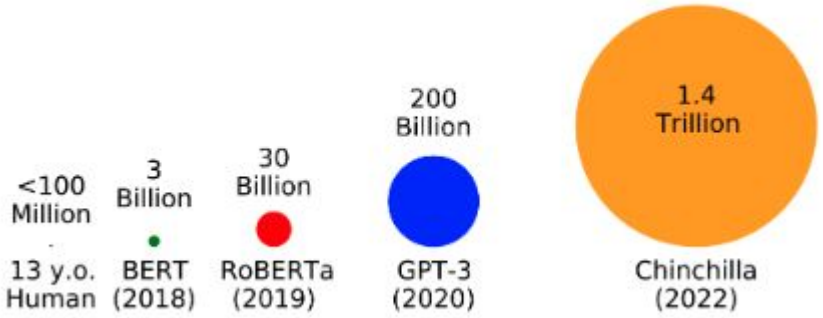
```
# Q: Jane was born on the last day of Feburary in 2001. Today is her 16-year-old
    birthday. What is the date 24 hours later in MM/DD/YYYY?
# If Jane was born on the last day of Feburary in 2001 and today is her 16-year-old
    birthday, then today is 16 years later.
today = datetime(2001, 2, 28) + relativedelta(years=16)
# 24 hours later,
later = today + relativedelta(hours=24)
# The answer formatted with %m/%d/%Y is
later.strftime('%m/%d/%Y')
```

Figure 25: Example model generation on Date Understanding.

Gao & Al, 2022, PAL: Program-aided Language Models

Baby LM: Children vs LLMs

- A LM sees 10^{4-5} more words than a 4 years old child
- Children learn in a smarter, more explicit compositional/hierarchical manners learning abstraction/generalization and reasoning more easily
- Baby LM Challenge: train a powerful model with just a limited amount of data
- LM on device (edge computing, i.e smartphones



Baby LM Challenge

Baby LM: Children vs LLMs

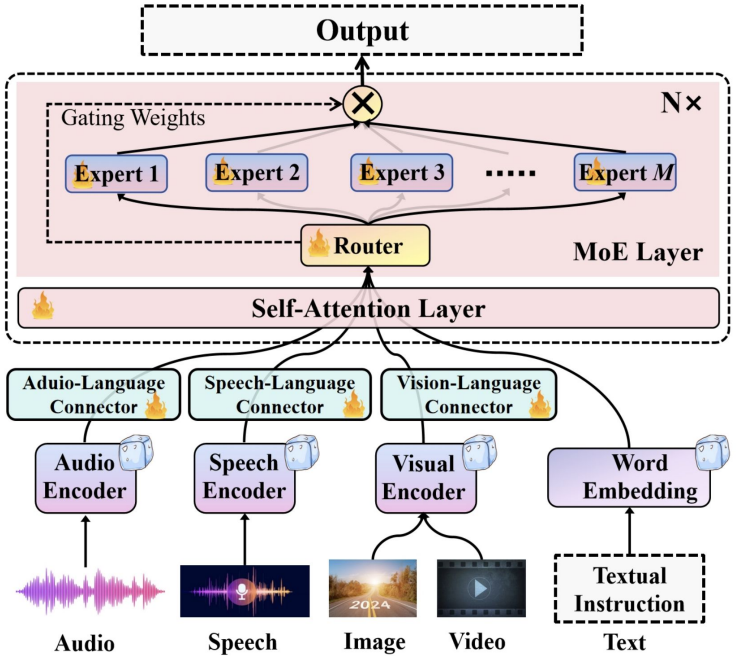
Developmentally plausible pre training dataset inspired by the input to children

- Under 100M words: Children are exposed to 2M-7M words per year (Gilkerson et al., 2017). Choosing the beginning of adolescence (age 13) as a cutoff, children are exposed to at maximum 91M words, which they round up to 100M.
- Mostly transcribed speech: Most of the input to children is spoken; thus, the dataset focuses on transcribed speech.
- Mixed domain, consisting of the following sources: CHILDES (child-directed speech), OpenSubtitles (speech), BNC (speech), TED talks (speech), children's books (simple written language), Wikipedia, and Simple Wikipedia.

Baby LM Challenge

Mixture of Expert (MoE) model

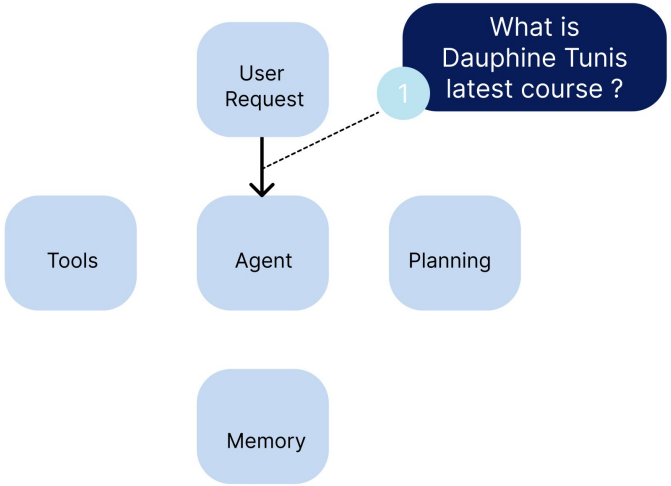
- Combination of multiple small Neural networks knowns as expert which are trained and capable of handling particular data and performing specialised task
- Gatin network predicts which response is best suited to address the request



Li & Al, 2024.5 Uni-MoE: Scaling Unified Multimodal LLMs with Mixture of Experts

IV.B. Agents & Tools

A. **Beyond LLMs**
B. **Agents & Tools**
C. **Multi Agents Systems**



Dauphine TUNIS | PSL

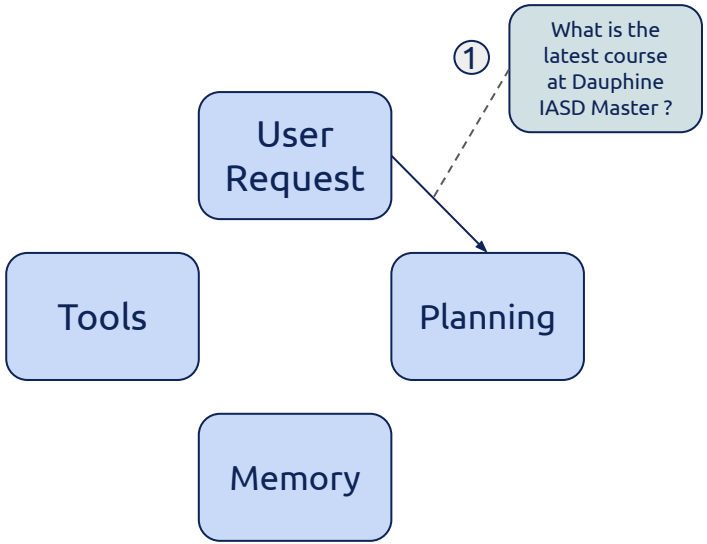
Example

https://www.youtube.com/watch?v=Ubr7FWmeIyM&t=75s

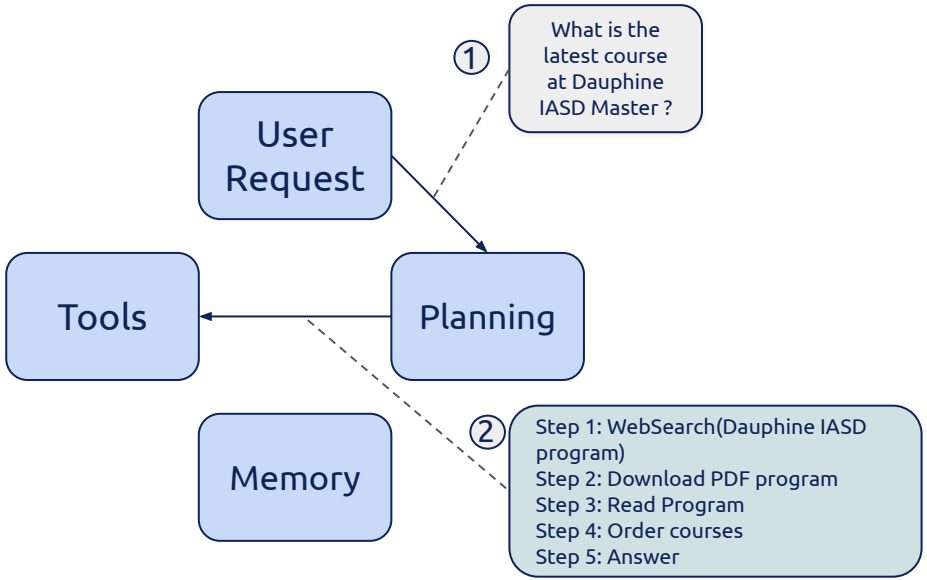
Why Agents ?

1. **Why ?** A single LLM call is not enough to do complex tasks
2. **How ?** Using model chaining, reflection, chain of thought …
3. **Tools ?** Memory, context length, personalization, actions, internet access…

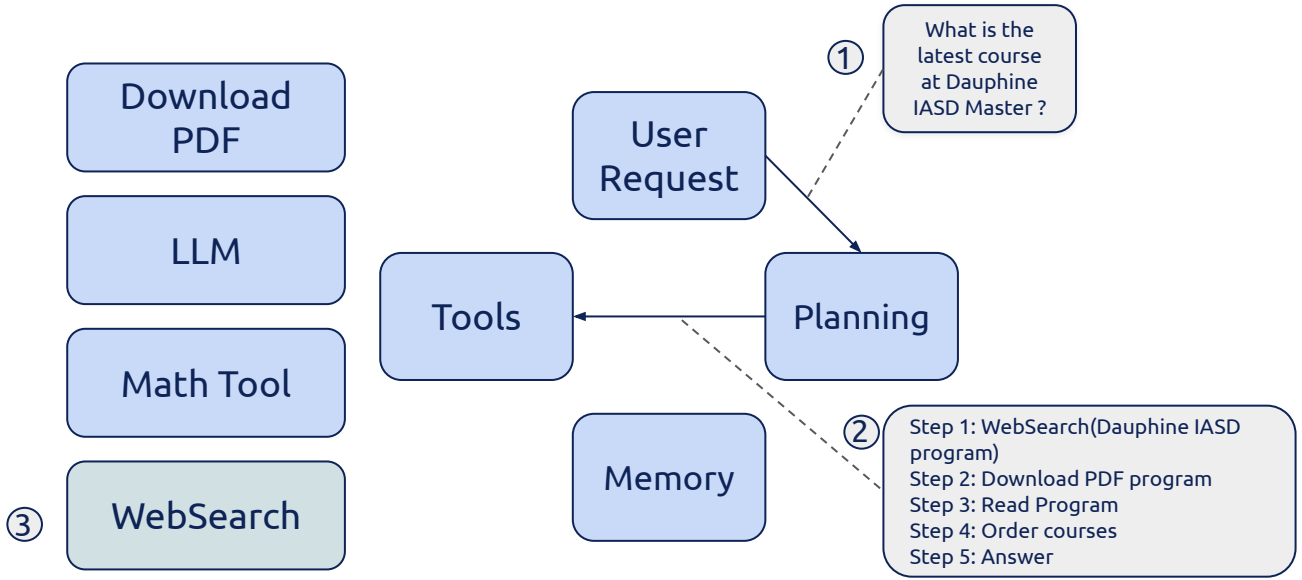
Agents example



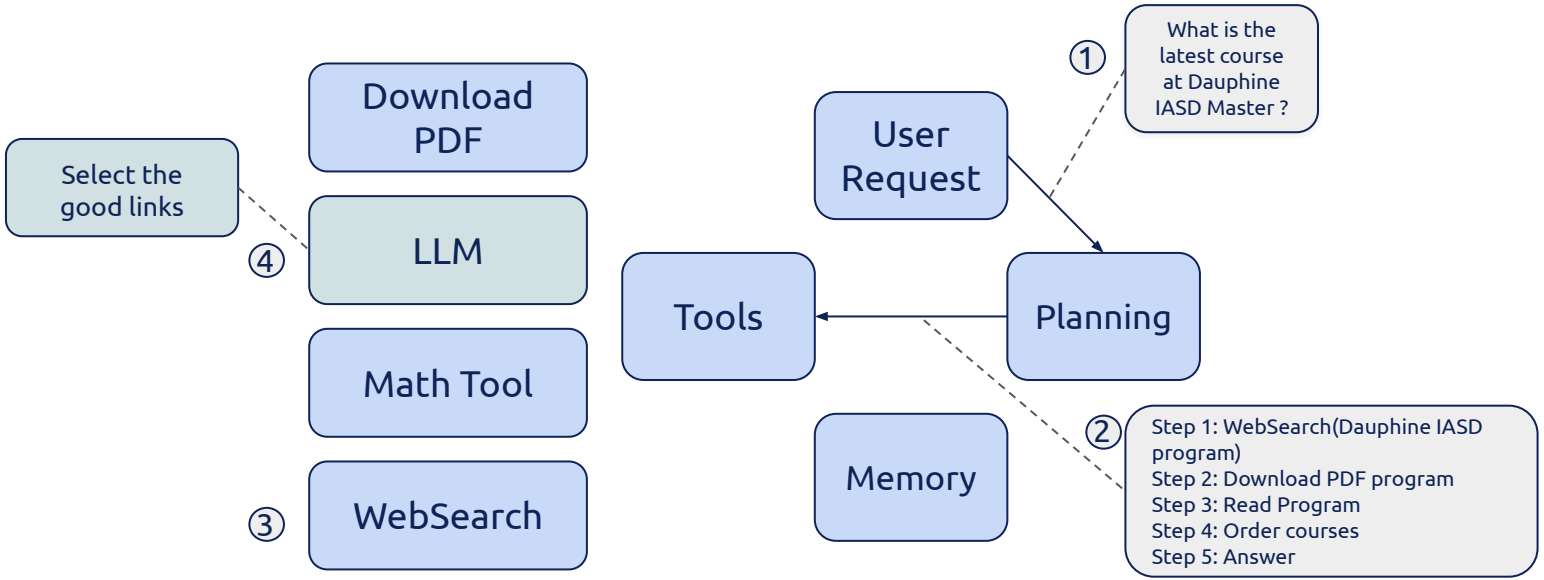
Agents example



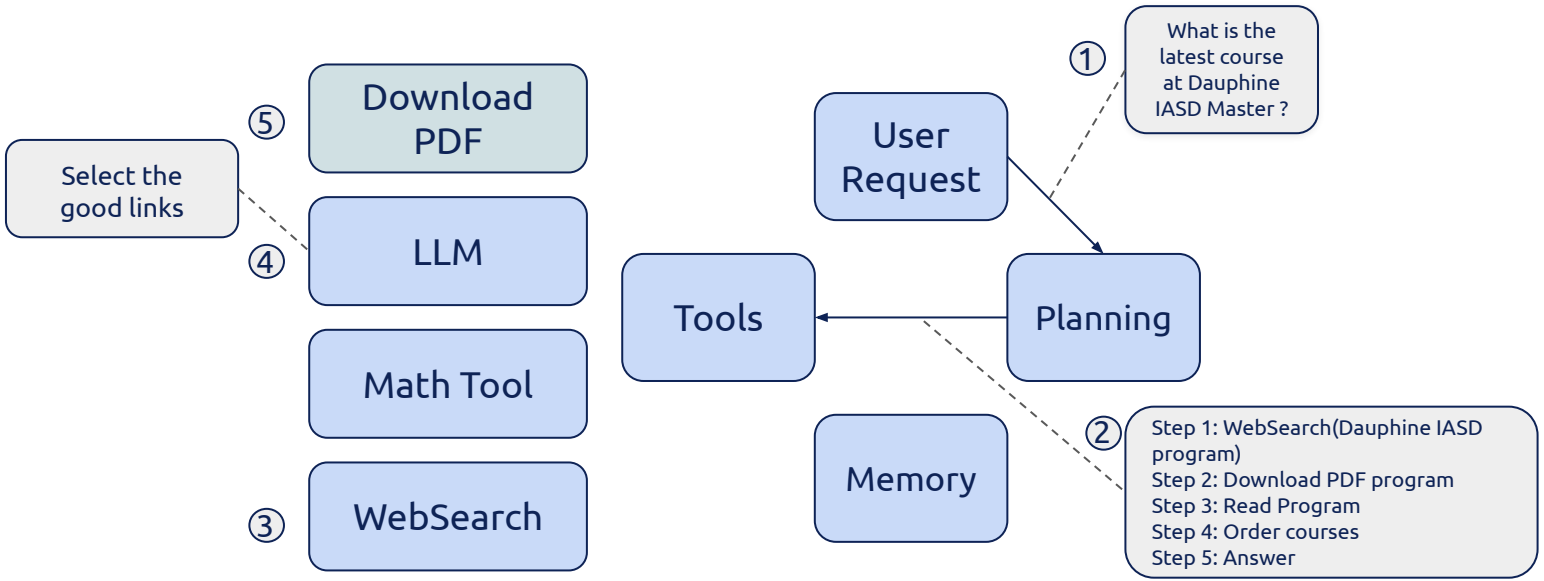
Agents example



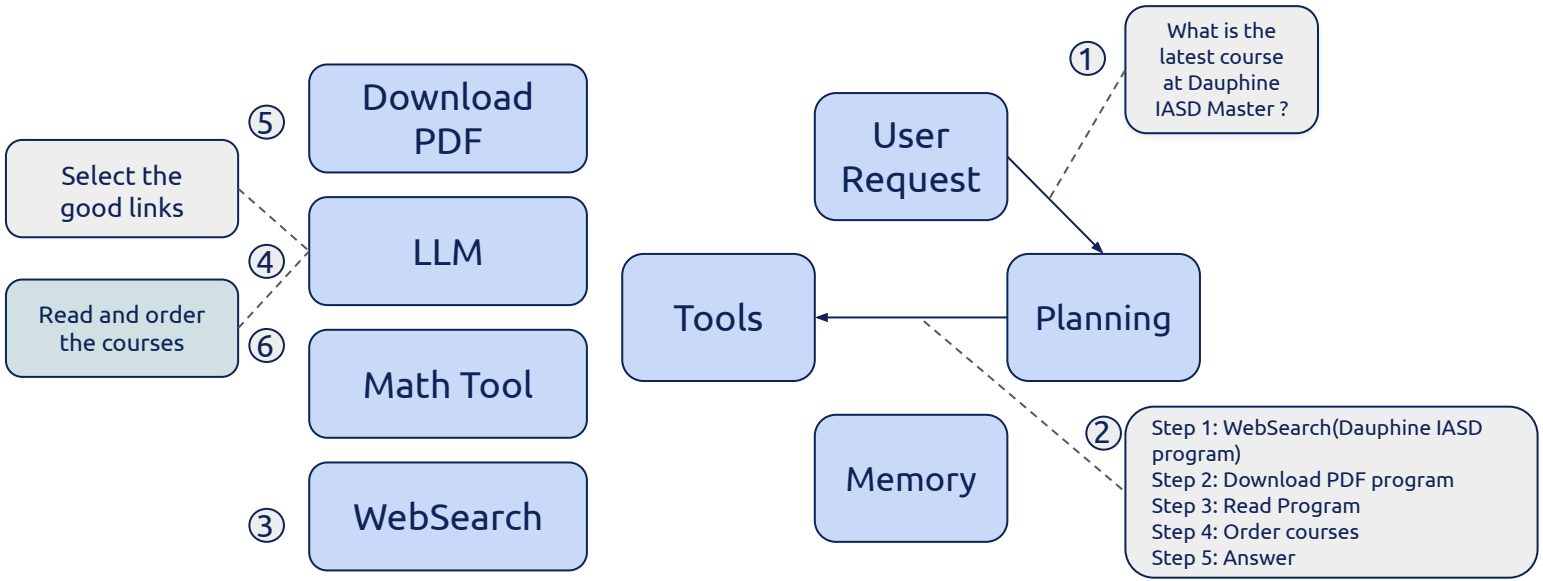
Agents example



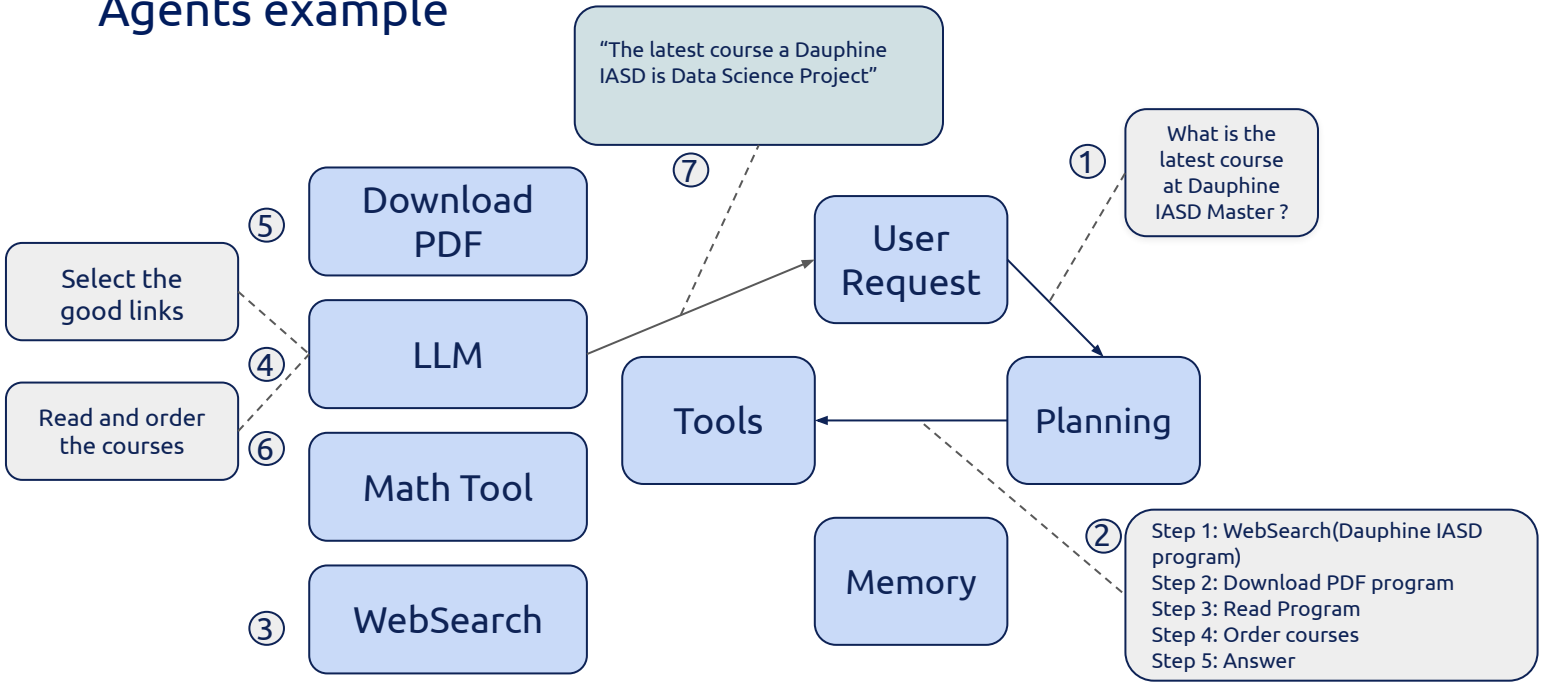
Agents example



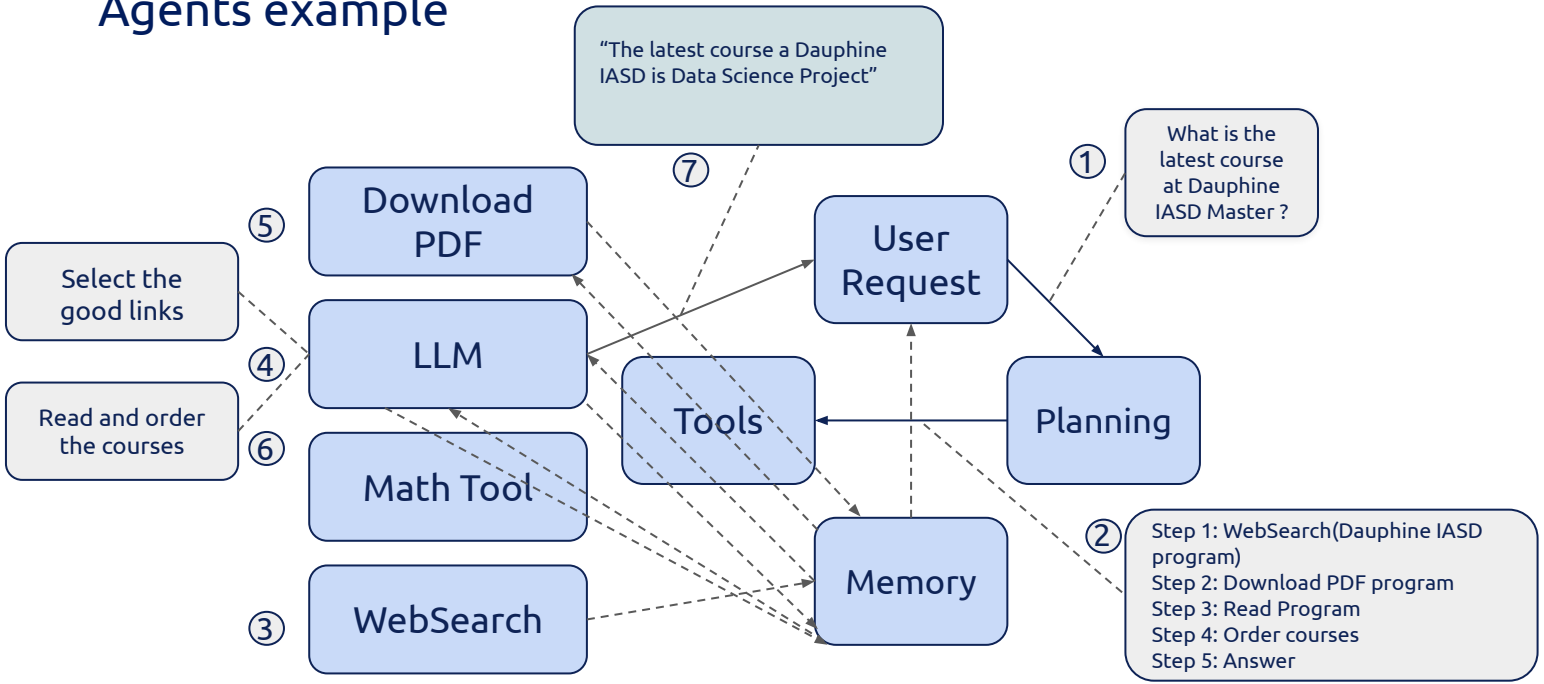
Agents example



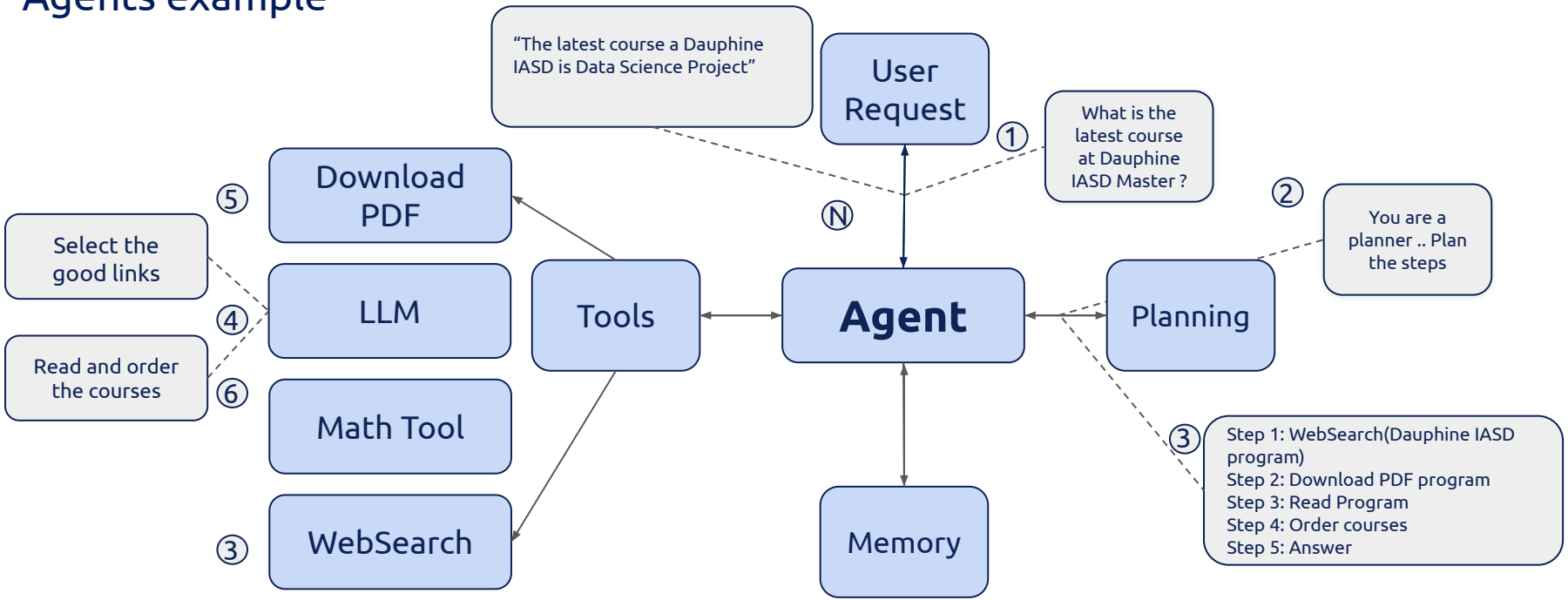
Agents example



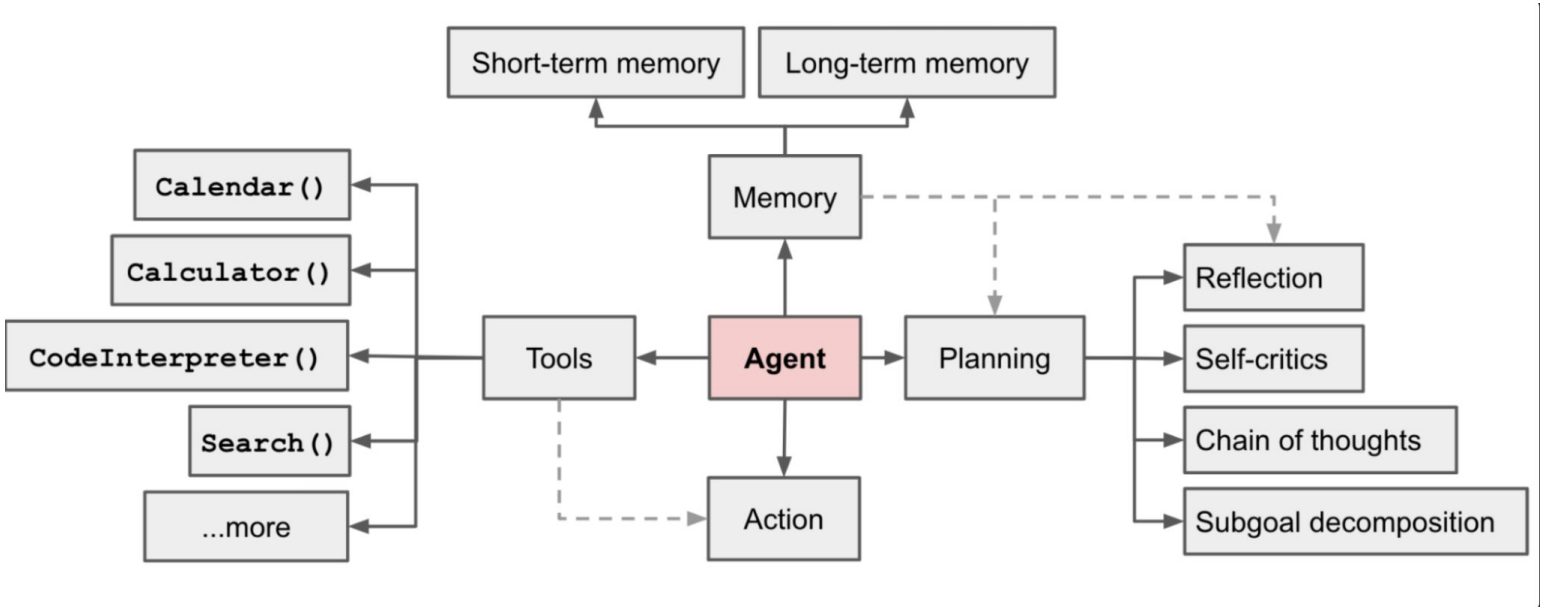
Agents example



Agents example



Agents



Human like AI Agents

Building agents with LLM (large language model) as its core controller is a end of 2023 concept.

Several proof-of-concepts demos, such as:

- AutoGPT, 167K ⭐
- GPT-Engineer, 52K ⭐
- ChatDev, 25K ⭐
- BabyAGI, 20K ⭐

are Agent example.

Chat Dev

ChatDev, 25K ⭐

Human like Agents

1. **Capable of using human-designed interfaces**: Able to work with existing interfaces meant for humans, transcending typical programmatic limitations.
2. **Acts as a digital extension**: Functions as an extension of the user, taking actions on their behalf.
3. **Fewer restrictions**: Can manage logins, payments, and interact with services without being limited by API constraints.
4. **Simple actions required**: Operates using basic action primitives, such as clicking and typing.
5. **Learns and improves**: Continuously learns from user interactions and enhances its performance over time.

Artificial general intelligence (AGI)

Definition: type of artificial intelligence (AI) that matches or surpasses human cognitive capabilities across a wide range of cognitive tasks.

To create an AGI, AI systems have to perform better than human.

Q. How to make performance comparison ?

Leaderboard

Submission made by our team are labelled "GAIA authors". While we report average scores over different runs when possible in our paper, we only report the best run in the leaderboard.

See below for submissions.

Citation

Results: Test | Results: Validation

Agent name	Average score (%)	Level 1 score (%)	Level 2 score (%)	Level 3 score (%)	organisation	Model family	Submission date
Multi Agent	38.87	53.76	37.74	14.29			2024-10-10
das_agent v0.4	38.21	51.61	36.48	18.37	NA	GPT-4o	2024-10-04
Trase Agent v0.1	35.55	50.54	33.33	14.29	Trase Systems	Fine-tuned GPT-4o	2024-09-04
Sibyl System v0.2	34.55	47.31	32.7	16.33	Baichuan Inc.	GPT-4o	2023-11-03

Hugging Face, General AI Assistant Benchmark

General AI Assistant benchmark (GAIA)

We **evaluate** the systems by their capacity of solving **tasks that requires reasoning**
We **don't evaluate** if they reason **like** humans

GAIA:
A Benchmark for General AI Assistants

Grégoire Mialon[1], **Clémentine Fourrier**[2], **Craig Swift**[3], **Thomas Wolf**[2], **Yann LeCun**[1], **Thomas Scialom**[4]

[1]FAIR, Meta, [2]HuggingFace, [3]AutoGPT, [4]GenAI, Meta

We introduce GAIA, a benchmark for General AI Assistants that, if solved, would represent a milestone in AI research. GAIA proposes real-world questions that require a set of fundamental abilities such as reasoning, multi-modality handling, web browsing, and generally tool-use proficiency. GAIA questions are conceptually simple for humans yet challenging for most advanced AIs: we show that human respondents obtain 92% vs. 15% for GPT-4 equipped with plugins. This notable performance disparity contrasts with the recent trend of LLMs outperforming humans on tasks requiring professional skills in *e.g.* law or chemistry. GAIA's philosophy departs from the current trend in AI benchmarks suggesting to target tasks that are ever more difficult for humans. We posit that the advent of Artificial General Intelligence (AGI) hinges on a system's capability to exhibit similar robustness as the average human does on such questions. Using GAIA's methodology, we devise 466 questions and their answer. We release our questions while retaining answers to 300 of them to power a leader-board hereby accessible.

Date: November 23, 2023
Correspondence: {gmialon,tscialom}@meta.com, clementine@huggingface.co
Code: https://huggingface.co/gaia-benchmark

Level 1

Question: What was the actual enrollment count of the clinical trial on H. pylori in acne vulgaris patients from Jan-May 2018 as listed on the NIH website?
Ground truth: 90

Level 2

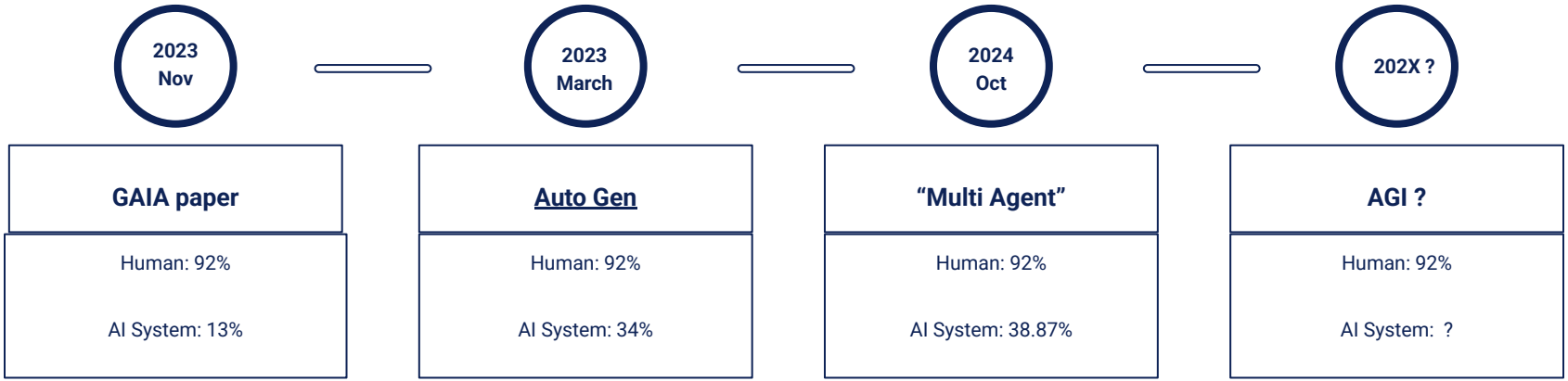
Question: If this whole pint is made up of ice cream, how many percent above or below the US federal standards for butterfat content is it when using the standards as reported by Wikipedia in 2020? Answer as + or - a number rounded to one decimal place.
Ground truth: +4.6

Level 3

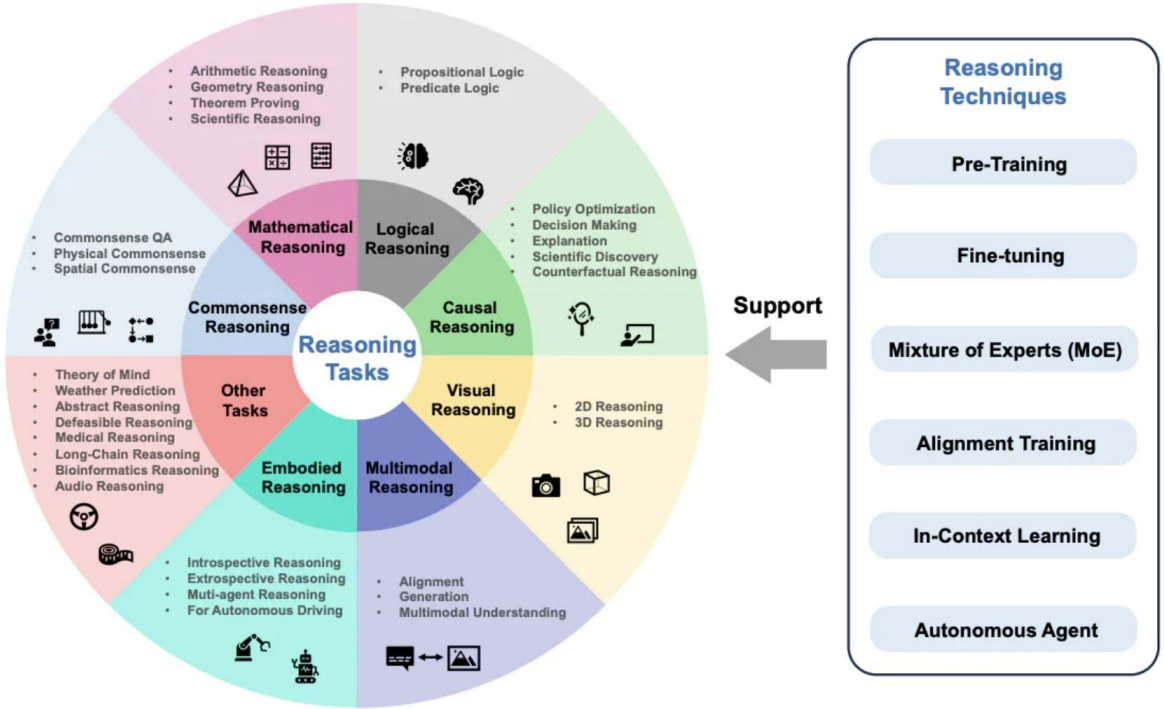
Question: In NASA's Astronomy Picture of the Day on 2006 January 21, two astronauts are visible, with one appearing much smaller than the other. As of August 2023, out of the astronauts in the NASA Astronaut Group that the smaller astronaut was a member of, which one spent the least time in space, and how many minutes did he spend in space, rounded to the nearest minute? Exclude any astronauts who did not spend any time in space. Give the last name of the astronaut, separated from the number of minutes by a semicolon. Use commas as thousands separators in the number of minutes.
Ground truth: White; 5876

LeCun, Wolf & Al, 2023, GAIA: A Benchmark for General AI Assistants

Will AI Systems become AGI ?

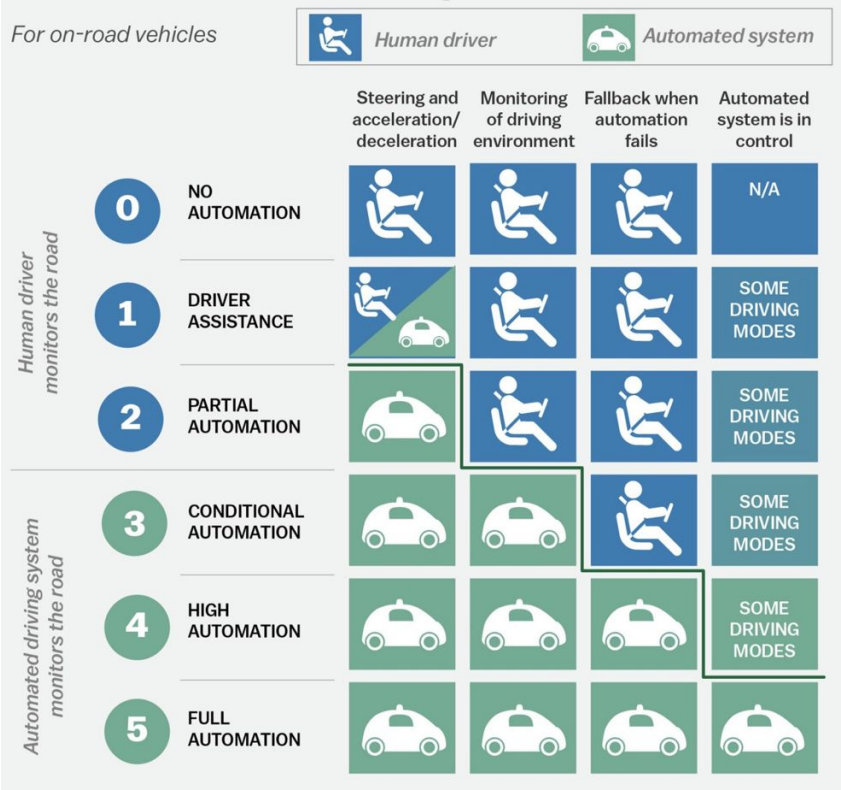


Reasoning tasks challenges



Sun & Al, 2023, Reasoning with Language Model Prompting: A Survey

Automation Levels



The Rise of Autonomous Vehicles

Automation Levels

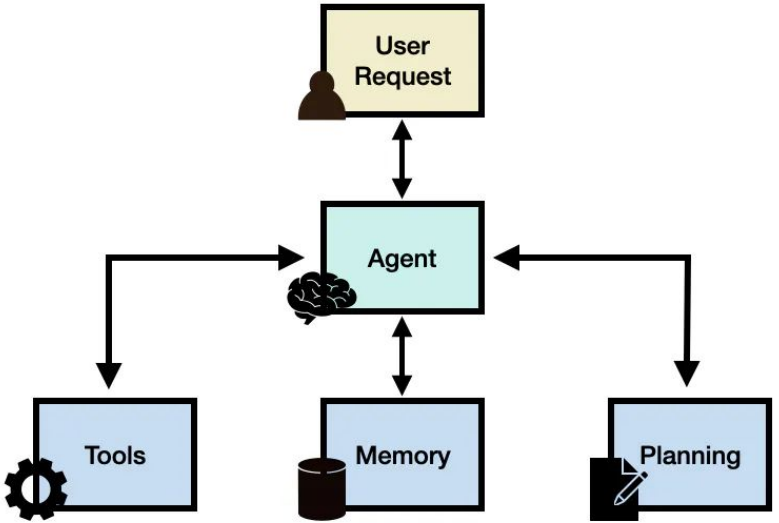
Performance (rows) x **Generality** (columns)	**Narrow** *clearly scoped task or set of tasks*	**General** *wide range of non-physical tasks, including metacognitive tasks like learning new skills*
Level 0: No AI	**Narrow Non-AI** calculator software; compiler	**General Non-AI** human-in-the-loop computing, e.g., Amazon Mechanical Turk
Level 1: Emerging *equal to or somewhat better than an unskilled human*	**Emerging Narrow AI** GOFAI (Boden, 2014); simple rule-based systems, e.g., SHRDLU (Winograd, 1971)	**Emerging AGI** ChatGPT (OpenAI, 2023), Bard (Anil et al., 2023), Llama 2 (Touvron et al., 2023), Gemini (Pichai & Hassabis, 2023)
Level 2: Competent *at least 50th percentile of skilled adults*	**Competent Narrow AI** toxicity detectors such as Jigsaw (Das et al., 2022); Smart Speakers such as Siri (Apple), Alexa (Amazon), or Google Assistant (Google); VQA systems such as PaLI (Chen et al., 2023); Watson (IBM); SOTA LLMs for a subset of tasks (e.g., short essay writing, simple coding)	**Competent AGI** not yet achieved
Level 3: Expert *at least 90th percentile of skilled adults*	**Expert Narrow AI** spelling & grammar checkers such as Grammarly (Grammarly, 2023); generative image models such as Imagen (Saharia et al., 2022) or Dall-E 2 (Ramesh et al., 2022)	**Expert AGI** not yet achieved
Level 4: Virtuoso *at least 99th percentile of skilled adults*	**Virtuoso Narrow AI** Deep Blue (Campbell et al., 2002), AlphaGo (Silver et al., 2016; 2017)	**Virtuoso AGI** not yet achieved
Level 5: Superhuman *outperforms 100% of humans*	**Superhuman Narrow AI** AlphaFold (Jumper et al., 2021; Varadi et al., 2021), AlphaZero (Silver et al., 2018), StockFish (Stockfish, 2023)	**Artificial Superintelligence (ASI)** not yet achieved

Morris & Al, 2024. Position: Levels of AGI for Operationalizing Progress on the Path to AGI

Agent components

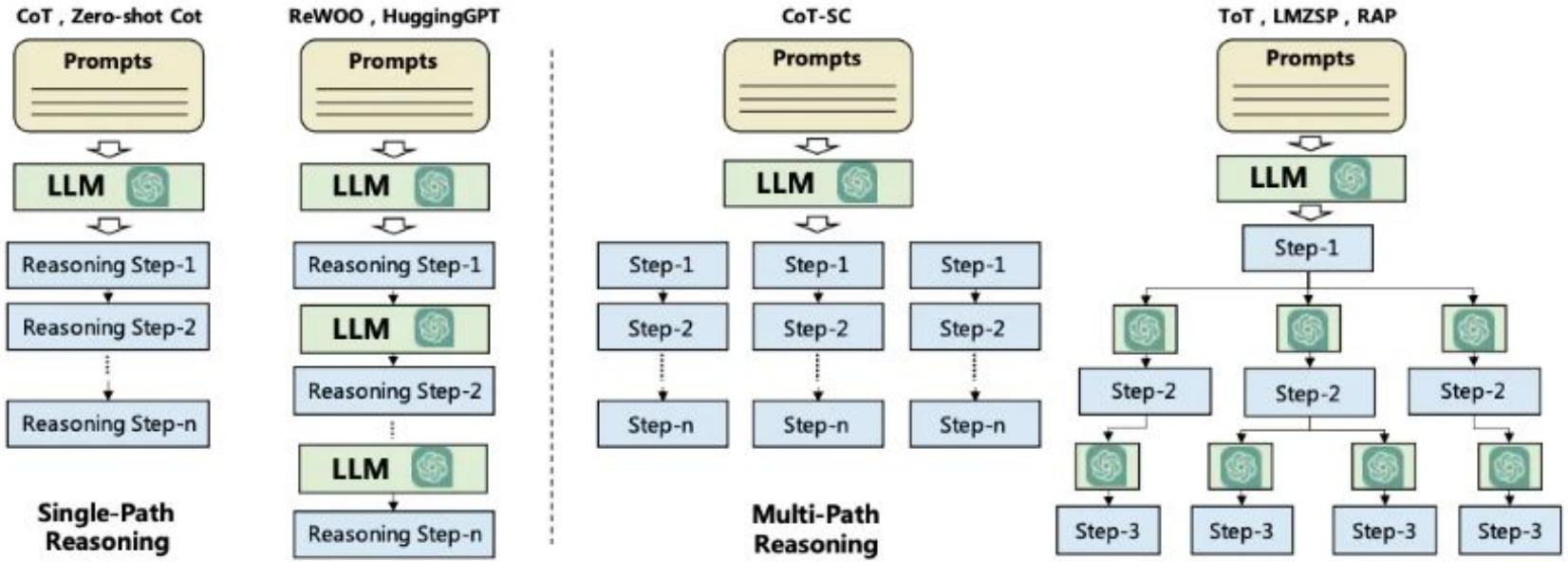
Generally speaking, an LLM agent framework can consist of the following core components:

User Request: a user question or request
Agent/Brain: the agent core acting as coordinator
Planning: assists the agent in planning future actions
Memory: manages the agent's past behaviors



Wang & Al, 2024, A Survey on Large Language Model based Autonomous Agents

Planning strategies

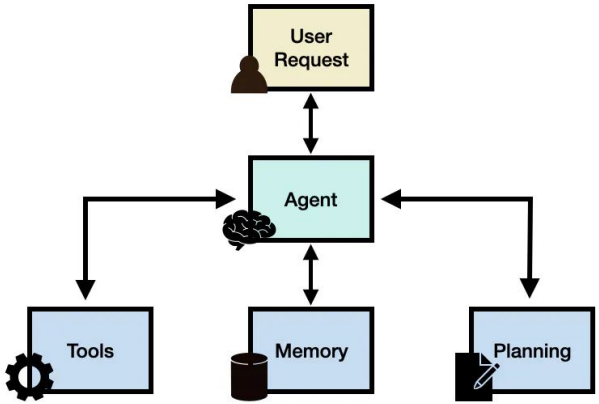


Wang & Al, 2024, A Survey on Large Language Model based Autonomous Agents

Planning feedback

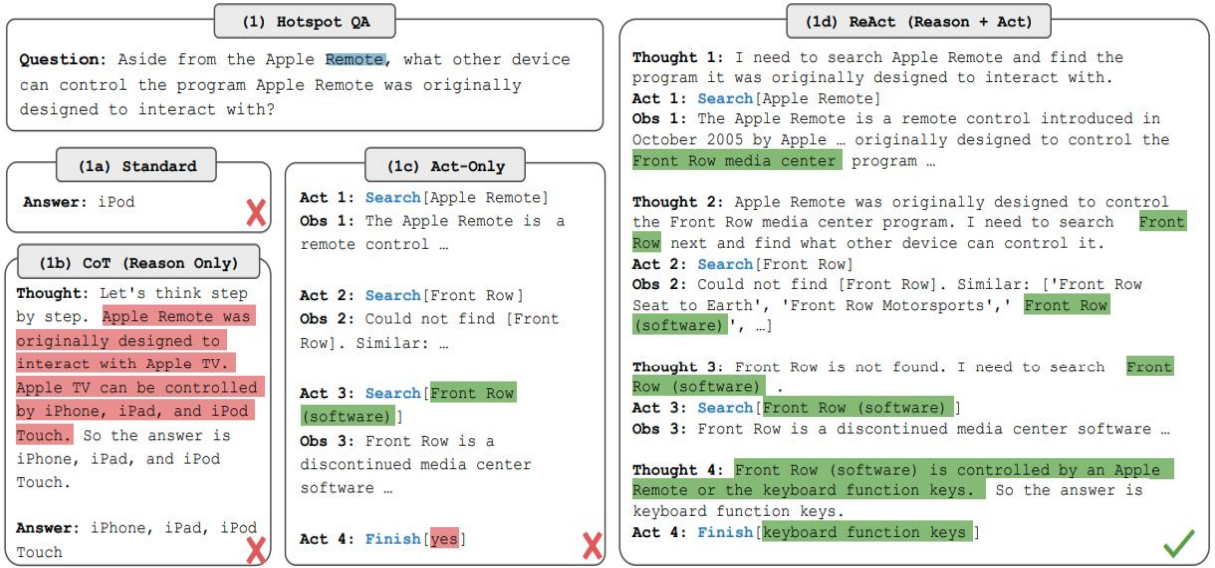
Planning module have no feedback
The plan is provided at the beginning of the execution and will fail if any of the step fails

Q. How to address this challenge ?



Wang & Al, 2024, A Survey on Large Language Model based Autonomous Agents

Synergizing Reasoning and Acting in Language Models (ReAct)



Prompt Method[a]	HotpotQA (EM)	Fever (Acc)
Standard	28.7	57.1
CoT (Wei et al., 2022)	29.4	56.3
CoT-SC (Wang et al., 2022a)	33.4	60.4
Act	25.7	58.9
ReAct	27.4	60.9
CoT-SC → ReAct	34.2	**64.6**
ReAct → CoT-SC	**35.1**	62.0
Supervised SoTA[b]	67.5	89.5

Yao & Al. 2023. Synergizing Reasoning and Acting in Language Models (ReAct)

Memory

Definition: The **memory** module is essential for storing an agent's internal logs, including previous thoughts, actions, and observations from interactions with the environment and users. In the literature on LLM agents, two primary types of memory have been identified:

- **Short-term memory**: This contains context about the agent's current situation, often implemented through in-context learning. It is limited and short-lived due to the constraints of the context window.
- **Long-term memory**: This stores the agent's past behaviors and thoughts that must be retained and accessed over longer periods. It typically relies on an external vector store, enabling fast and scalable retrieval of relevant information when needed.

A **hybrid memory** system combines both short-term and long-term memory to enhance the agent's capacity for long-range reasoning and accumulating experience.

Both the planning and memory modules allow the agent to operate in a dynamic environment and enable it to effectively recall past behaviors and plan future actions.

Tools

Definition: Tools refer to a set of utilities that enable the LLM agent to interact with external environments, such as the Wikipedia Search API, Code Interpreter, or Math Engine.

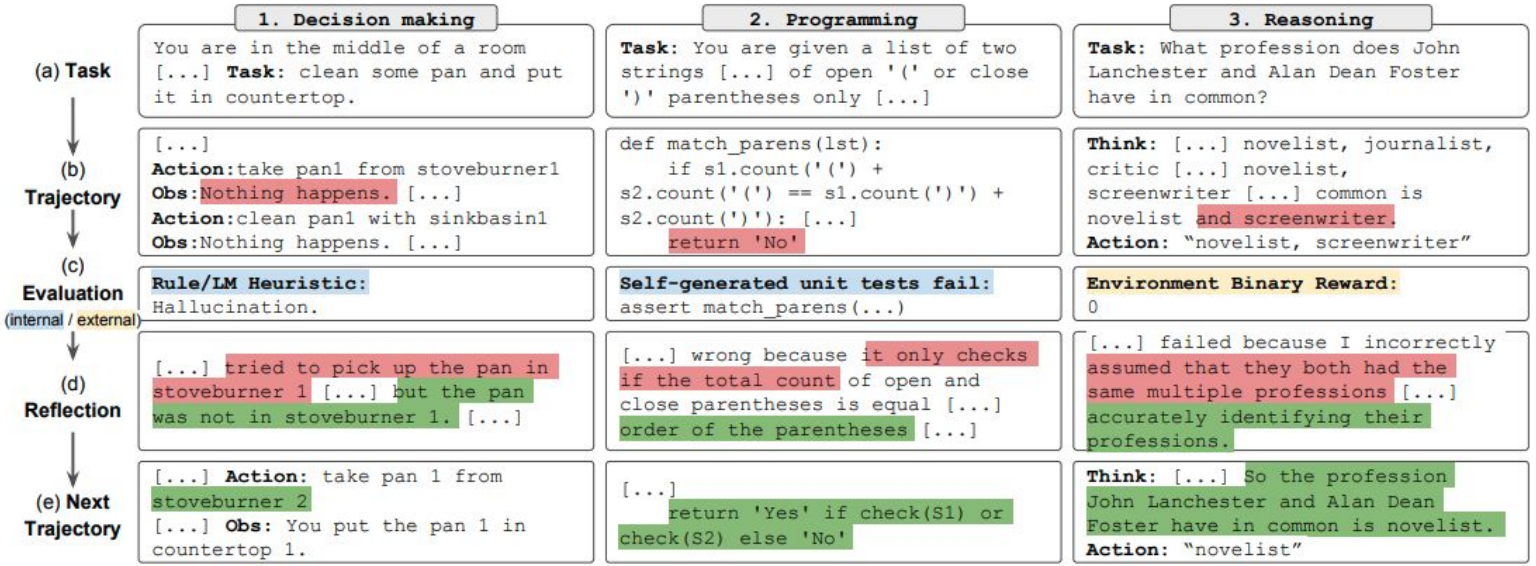
These tools can also include:

- databases
- knowledge bases
- external models.

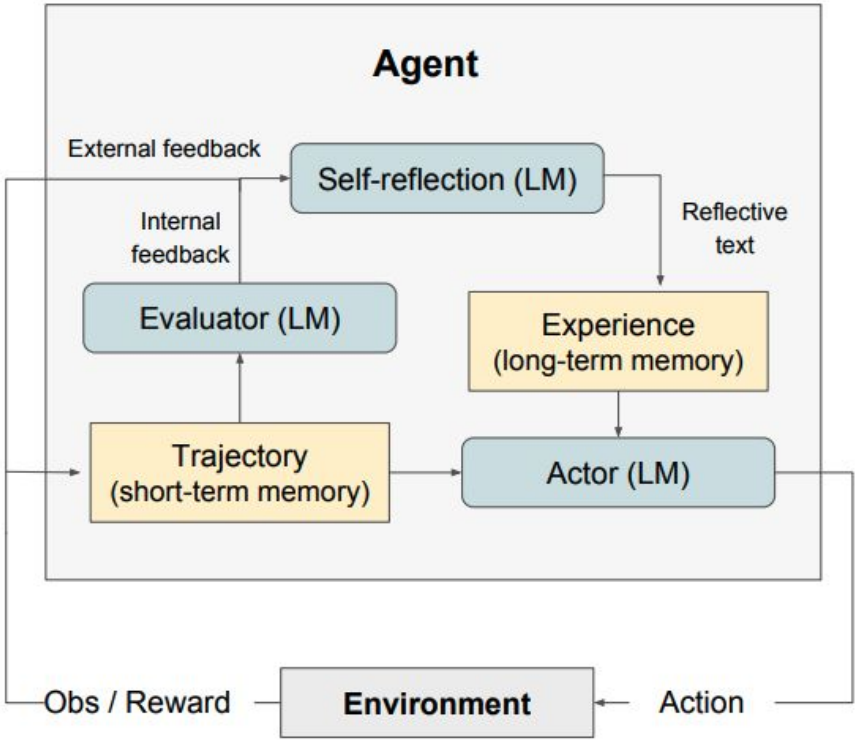
When the agent uses these external tools, it carries out tasks through workflows designed to help the agent gather observations or relevant information needed to complete subtasks and fulfill user requests.

Example:
In a health-related query, a code interpreter serves as a tool that executes code and generates the chart information requested by the user.

Reflexion: Language Agents with Verbal Reinforcement Learning



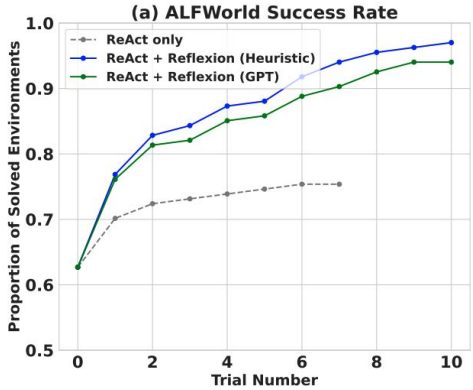
Reflexion: Language Agents with Verbal Reinforcement Learning

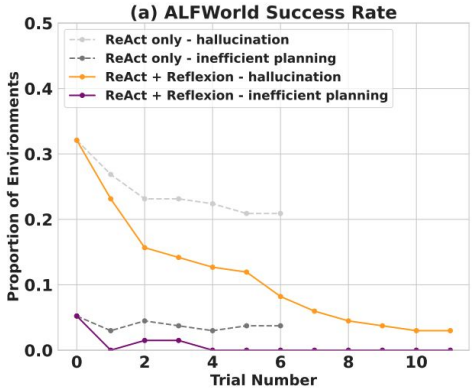


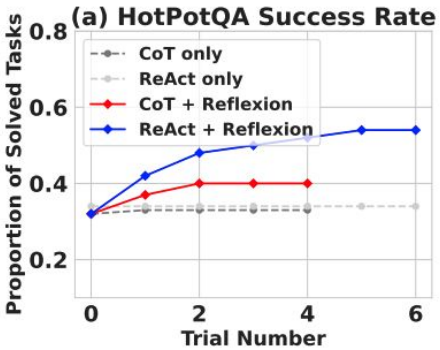
Shinn & Al, 2023, Reflexion: Language Agents with Verbal Reinforcement Learning

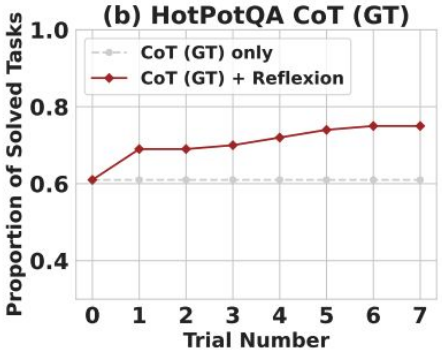
Reflexion: Language Agents with Verbal Reinforcement Learning

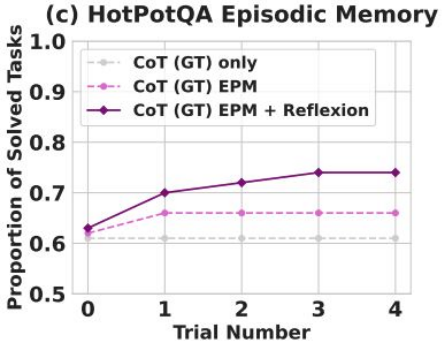
Performance









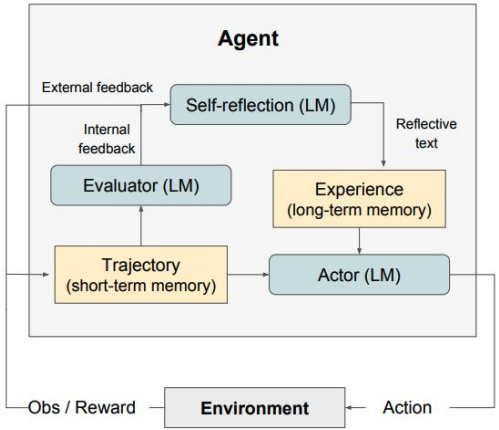


Shinn & Al, 2023, Reflexion: Language Agents with Verbal Reinforcement Learning

Reflexion: Language Agents with Verbal Reinforcement Learning

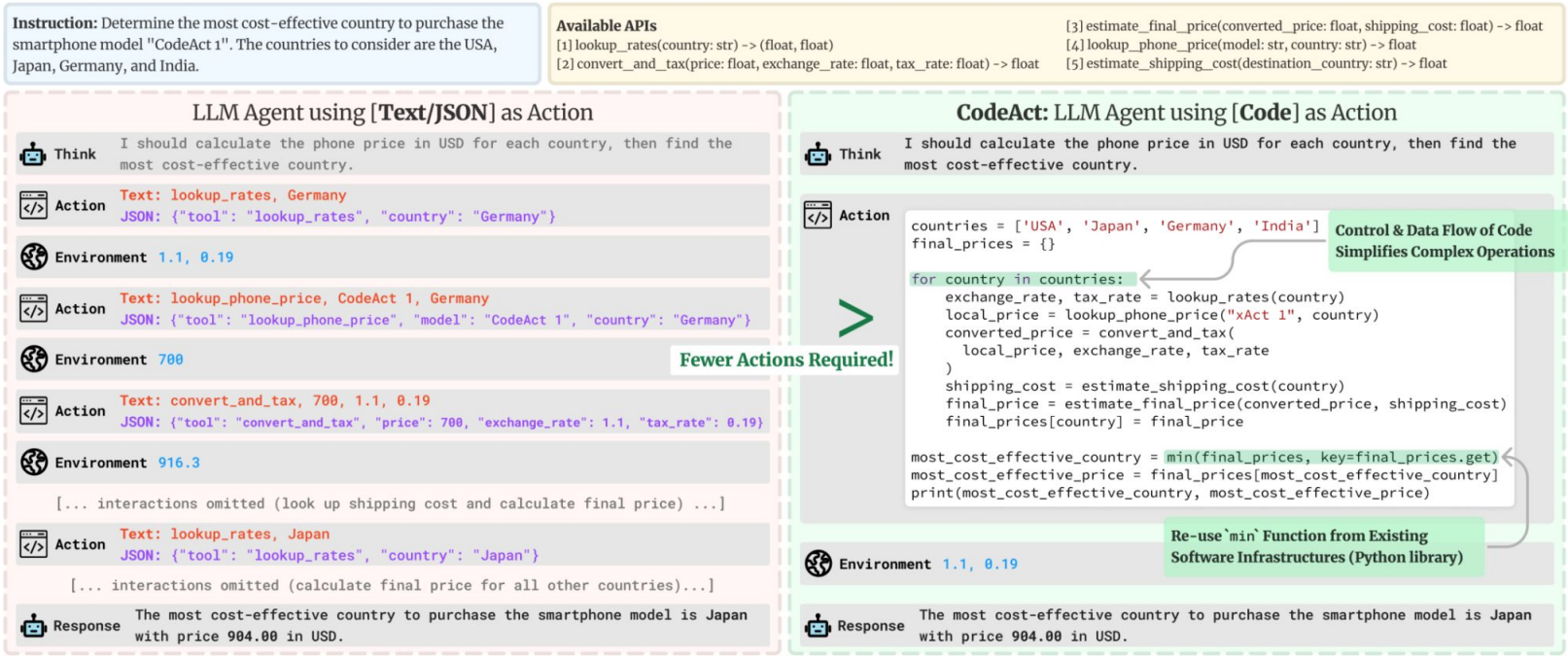
Definition: Reflexion is a framework designed to enhance language-based agents by using linguistic feedback.
Reflexion introduces a novel approach to verbal reinforcement, where the policy is defined by the agent's memory encoding combined with selected LLM parameters.

- At its core, Reflexion transforms feedback whether in the form of free language or numeric scores from the environment into linguistic feedback, also known as self-reflection.
- This feedback is then incorporated as contextual information for the LLM agent in subsequent interactions.
- This process enables the agent to quickly learn from past errors, leading to improved performance across various complex tasks.



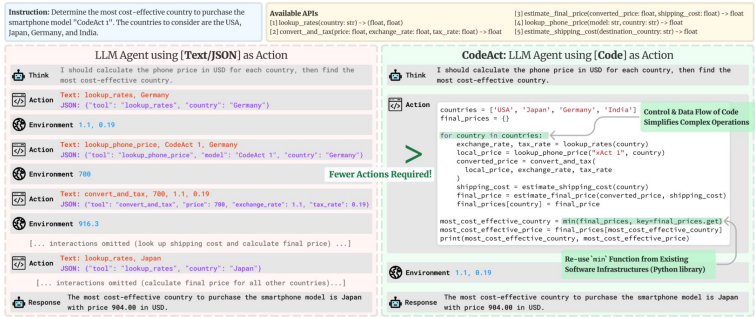
Shinn & Al, 2023, Reflexion: Language Agents with Verbal Reinforcement Learning

Code Actions Elicit Better LLM Agents



Wang & Al, 2024, Executable Code Actions Elicit Better LLM Agents

Code Actions Elicit Better LLM Agents



Code actions are more concise than JSON actions.

Efficiency in Running Parallel Actions:
To run 4 parallel streams of 5 consecutive actions:
- *In JSON*: Requires generating 20 separate JSON blobs in multiple steps.
- *In Code*: Achieved in a single step.

Reduced Steps and Cost:
- Code actions require 30% fewer steps than JSON actions on average.
- Fewer steps mean 30% fewer tokens generated, making LLM calls and thus agent system runs approximately 30% cheaper.

Reusability with Common Libraries:
- Code allows the reuse of tools from existing libraries, enhancing functionality and efficiency.

Improved Benchmark Performance with Code:
- **Intuitive Expression**: Code is a more natural way to express actions.
- **LLM Proficiency:** LLMs are extensively trained on code, making them more fluent in writing code than JSON.

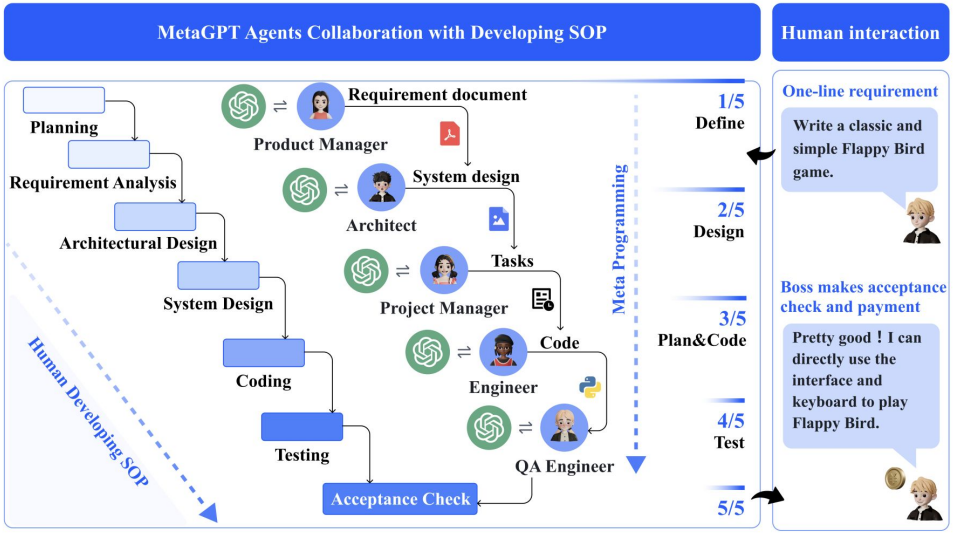
Wang & Al, 2024, Executable Code Actions Elicit Better LLM Agents

Challenges

- Collecting user data & preferences:
  - Actively asking for preferences
  - Passive learning from interactions
- Learning from user preferences: Supervised fine-tuning vs human-feedback
- On-fly adaptation
- Privacy

IV.C. Multi Agents Systems

A. **Beyond LLMs**
B. **Agents & Tools**
C. **Multi Agents Systems**



Hong & Al, 2023.11, Meta GPT

Dauphine | PSL TUNIS

Multi Agent Systems

Definition: LLM-based multi-agent systems refer to frameworks where multiple autonomous agents, each potentially powered by a Large Language Model, interact within a shared environment. These agents can cooperate, compete, or operate independently, aiming to achieve individual or shared goals.

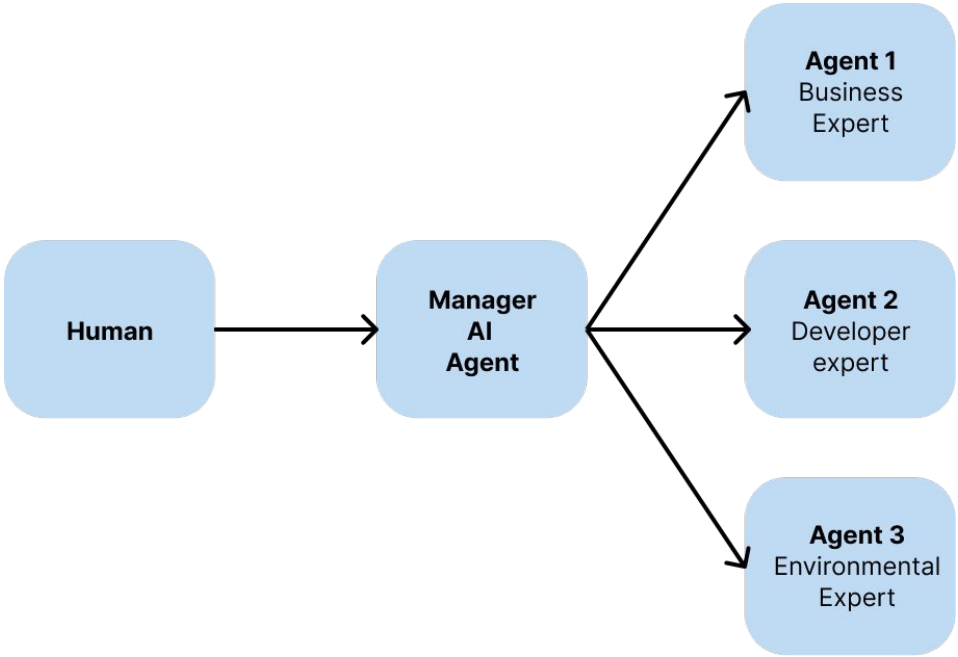
- **Autonomy**: Each agent operates independently and can make decisions based on its objectives, knowledge, and the perceived state of the environment.
- **Scalability**: LLMs enable these agents to handle complex tasks and large amounts of data, making the system scalable to various applications and scenarios.
- **Interoperability**: Agents can communicate and negotiate with each other, using natural language or other forms of structured data facilitated by their underlying LLM capabilities.
- **Adaptability**: These systems are capable of adapting to new information or changes in the environment, learning from interactions and refining their strategies over time.

Why Multi Agent Systems

- **Parallelization** is possible for Multi Agents systems
- **Task specialization**: A agent can be the intermediate between the user and services (spreadsheet, slides, internet expert, coder)
- **Challenges**:
  - Agent to Agent Communication
- **How to exchange informations (natural languages) ?**
  - Like human, there are misunderstanding

Agent to agent communication

- Agents can communicate between each other
- Hierarchy notion like manager
- Worker can run on different GPU
- Create failover mechanism



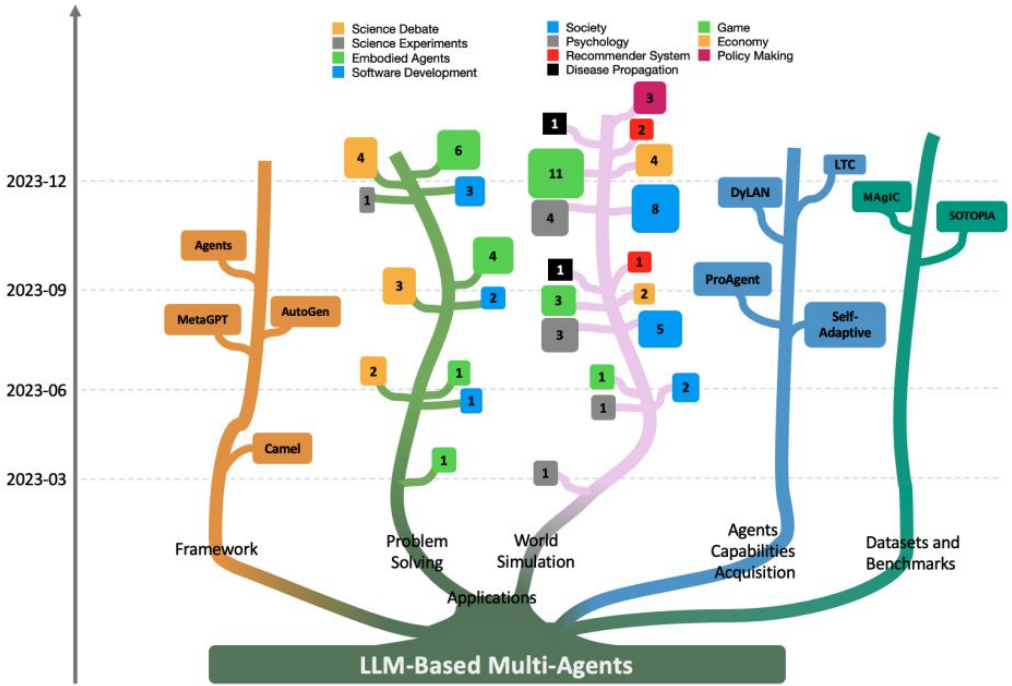
Limitations

- **Reliability**: Ensuring that the autonomous agent consistently performs its tasks without failure or errors.
- **Looping and Plan Divergence**: Addressing scenarios where the agent's decision-making process leads to infinite loops or deviates from the original plan.
- **Testing and Benchmarking:** Developing robust methods to evaluate and benchmark the performance of autonomous agents across different tasks and environments.
- **Real-World Deployment and Observability:**
  - Trust: How do we ensure trust in fully autonomous AI systems, especially in high-stakes environments?
  - Human Overrides: How can we integrate mechanisms for human intervention or overrides in the event of unforeseen circumstances or errors ?

Limitations

Auto GPT used $1000 of Open AI API calls to order a $15 pizza

Survey of progresses and challenges



Guo & al, 2024, Large Language Model based Multi-Agents: A Survey of Progress and Challenges

Multi Agent Systems

Y Combinator, 2024, October, Why Vertical LLM Agents Are The New $1 Billion SaaS Opportunities

Go to : https://kahoot.com/
Click Play
Enter code: XXX XXX
Enter your name

I. Fine tuning & optimizations techniques

A. Pretraining a Large Language Model

1. Introduction
2. Cross entropy loss
3. Tokenization
4. Evaluation
5. Data preprocessing
6. Scaling laws
7. Training process
8. Cost and optimization

B. Fine tuning a Large Language Model

1. Supervised Fine Tuning
2. RLHF
3. Reward model
4. PPO & DPO
5. Evaluation & Challenges

Dauphine | PSL
TUNIS

https://www.cloudskillsboost.google/focuses/104686?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&parent=catalog&search_id=40177754

Dauphine | PSL
TUNIS

Module Overview:

1. Rappels
   a. méthodes de similarités, Cos, MMR, RRF,
   b. HNSW, SCANN
   c. RNN LSTM
2. L'histoire des Transformers
   a. Transformer & Foundations models: Alexandre Allauzen, Anna Pappa
   b. LLM
      i. RLHF & alignment
      ii. PPO
3. Utilisation des LLMs → LANGCHAIN
   a. Context & Tokens
   b. Prompt Engineering
   c. Approches RAG, CRAG, GraphRAG
   d. HyDE, CrossEncoder, Reranker,
   e. Agents
   f. Les processus d'évaluations
4. Approches de fine tuning, LORA, Quantization, HF Hub
5. Ouverture modèles de diffusions

TGI has many SOTA techniques for decoding: Paged Attention, KV Caching and Flash Attention…
https://lnkd.in/e7BpXDqW

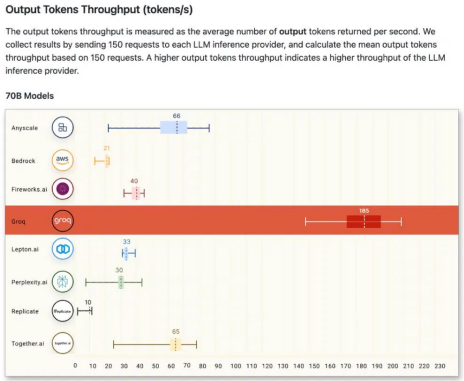
Projet chatbot, vidéo summarizer, agent, …

Dauphine | PSL
TUNIS

- Session 1: Understanding Large Language Models
  - What are Large Language Models (LLMs)
  - Overview of applications: natural language processing tasks, code generation, proof generation
  - Challenges and limitations of LLMs
- Session 2: Working with Text Data
  - Data preprocessing techniques for text data
  - Tokenization and vocabulary building
- Session 3: Attention Mechanisms
  - Types of attention mechanisms (self-attention, multi-head attention)
  - Efficient implementations
- Session 4: Implementing a GPT Model from scratch
  - Overview of Generative Pre-trained Transformers (GPT)
  - Architecture of GPT models (normalisation layers, residual connections, positional embedding)
  - Pretraining on Unlabeled Data: scaling laws, evaluation metrics
- Session 5: Fine Tuning
  - Fine tuning for text classification
  - Fine tuning with human feedback and reward models
- Session 6: Retrieval Augmented Generation
  - Using retrieval mechanisms to enhance generation tasks
  - Language agents
- Session 7: LLMs for code generation
  - Code generation tasks (code completion, code summarization, code repair)
  - Learning from code-related feedback
  - Inference algorithms for code
- Session 8: LLMs and formal code
  - LLMs for mathematics
  - Neural theorem proving
  - Verified code synthesis

Transformers

uphine | PSL TUNIS



https://groq.com/groq-lpu-inference-engine-crushes-first-public-llm-benchmark/

Dauphine | PSL
TUNIS

Si vous utilisez des
hashtag
#LLM Open-Source en production, ce poste vous intéressera probablement.

Il y a 2 semaines est sorti le papier présentant **SageAttention**, une implémentation du mécanisme d'attention optimisée pour l'inférence des transformers.

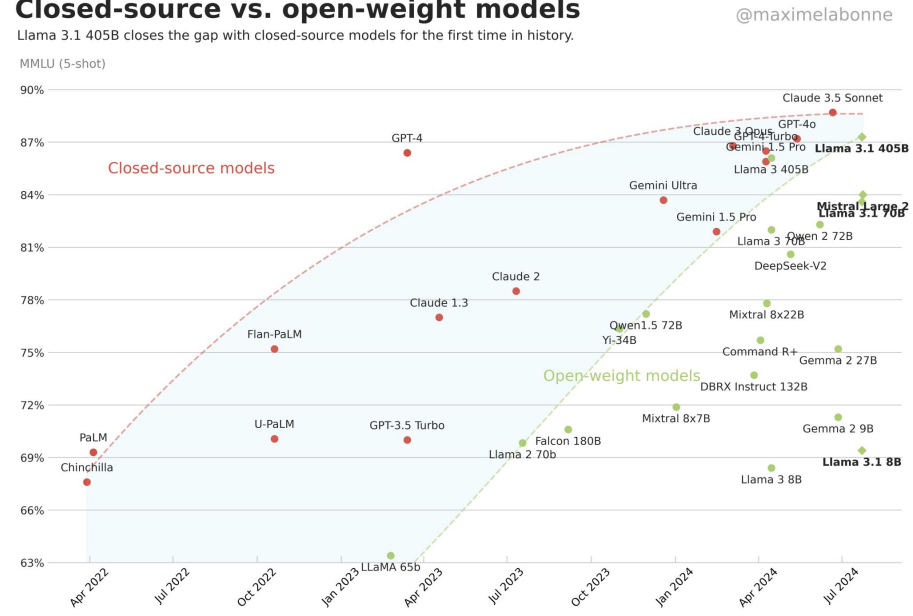
➡️ Cette approche est très prometteuse pour plusieurs raisons:
• Elle déclare accélérer grandement l'inférence (x1.7-2.1 par rapport à *Flash Attention* 2 sur une RTX4090)
• Elle est totalement plug-and-play, ce qui la rends très facile à incorporer dans vos modèles (cf l'image ci-dessous)

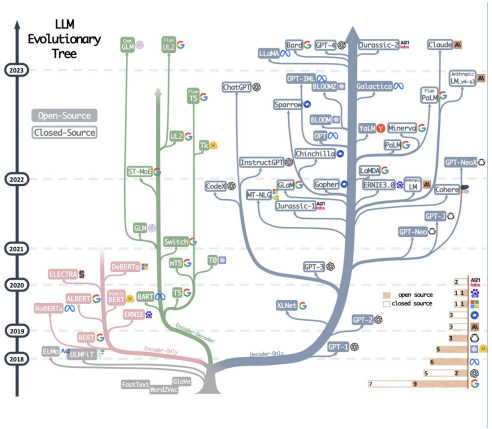
L'ayant testé, je peux vous faire quelques retours:

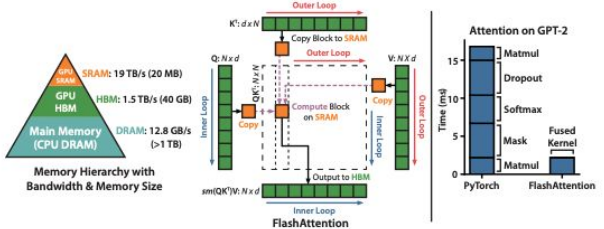
• L'accélération d'inférence n'est pas vérifiée sur n'importe quel type de GPU. Avec un GPU L4 dans Google Cloud (https://lnkd.in/gWphsXhr), je n'ai pas noté d'amélioration démesurée (7% plus rapide sur l'exemple donné dans leur repository Github)
• Et pour cause, les auteurs spécifient dans le package que leur algo est actuellement supporté par les cartes graphiques RTX4090 and RTX3090: pour les autres architectures GPU, nada.

💡 Malgré tout, je pense qu'il est intéressant de garder ce projet dans le viseur; si davantage

- Flash Attention
- Sage Attention https://arxiv.org/pdf/2410.02367
-







https://arxiv.org/pdf/2205.14135

Dauphine | PSL TUNIS